SWiSH Max

User's Guide

SWiSH Max User's Guide

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

CONTENTS

CHAPTER 5 Basic Working Procedures

CHAPTER 6 User Interface

CHAPTER 7 Help / Frequently asked Questions

CHAPTER 8 Tutorials

Welcome to SWiSH Max

Chapter 1

1 Welcome to SWiSH Max

SWiSH Max is a complete Flash™ animation authoring application. Create stunning and powerful Flash™ animations without using Adobe Flash™.

SWiSH Max (currently version 2) is easy to use and produces complex animations with text, images, graphics, video and sound. SWiSH Max has tools for creating lines, rectangles, ellipses, vector and freehand curves, motion paths, movie clips, rollover buttons, and input forms all in an intuitive easy-to-use interface.SWiSH Max also includes a large range of preset multimedia effects, components and vector art.

Earlier versions were called SWiSH Lite, SWiSH2 then SWiSH Max. SWiSH Max version 2 is the latest addition to the SWiSHzone.com family of Flash™ authoring tools and is an upgrade from the first version of SWiSH Max. Read about the new features in the chapter on "What is New in Version 2".
SWiSH Max exports the SWF file format used by Adobe Flash™, so the animation will play on any machine that has the Flash™ Player installed.

SWiSH Max animations can be incorporated into any web page or imported into Flash™. They can also be sent in an email, embedded in a Microsoft PowerPoint presentation or included in a Microsoft Word document.

See the Purchasing SWiSH Max section for details on how to buy and unlock SWiSH Max.

What is New in Version 2

Chapter 2

2 What is New in Version 2

With the release of version 2, SWiSH Max has undergone yet another transformation. SWiSH Max has been restructured from the ground up to add a new user interface, improved drawing tools, support for components, an expanded content library and many, many, other changes which will make working with SWiSH Max more effective, more efficient and more intuitive.

This section outlines many of the new features and improvements introduced in SWiSH Max version 2:

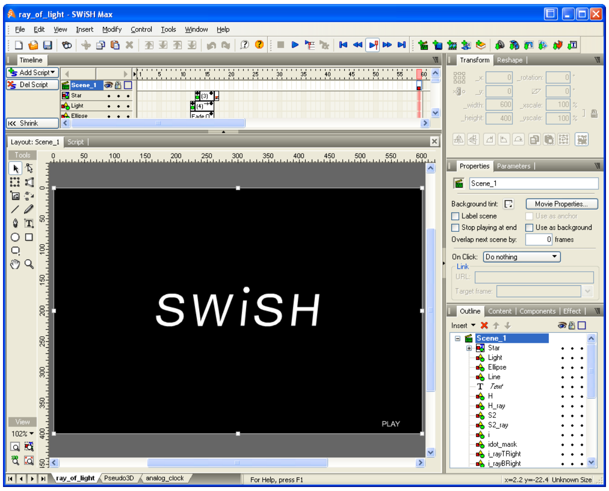
- User Interface
- Text Placement and Editing
- Effect Browser
- Internal Flash Player
- Improved Selection, Editing and Drawing Tools
- Video, Image and Sound Effects
- Preloader Speed Tester
- Improved Script Editor
- Content Library and Assets
- Components
- HTML Templates

2.1 User Interface

SWiSH Max version 2 includes some great improvements to the user interface.

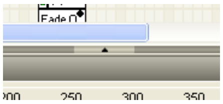
These are the most significant changes ...

New streamlined user interface
New dockable panels replace many dialogues. Drag panels by the left hand side handle to re-dock the panel. A right hand Panel Menu offers panel functions. Panel positions can be saved ('Windows | Layouts ...'). See User Interface.

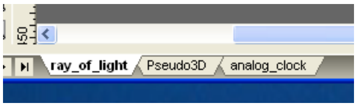


New Start Up Menu
Gives access to forums, updates, help, and for creating and editing recent SWI files. See Start Up Menu.

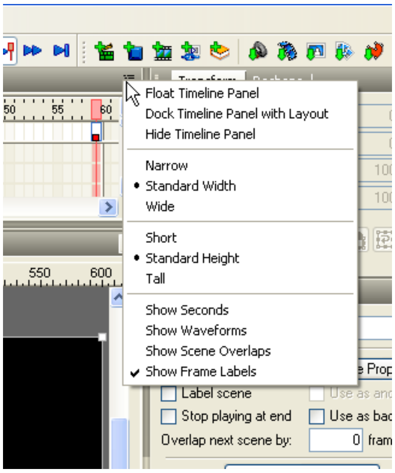
Quick hide/show panels
Panel docking areas can be quickly hidden with a click on the divider icon. See Managing Windows and Panels.



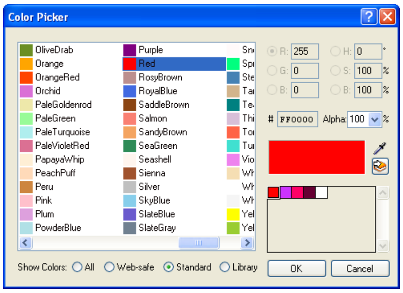
Multiple document support
Multiple Movies can be opened and worked on concurrently using the tabbed interface. See Managing Windows and Panels.



Panel menus
Panel menus include common operations and extra panel-specific commands (e.g. timeline panel settings, library operations). See Managing Windows and Panels.

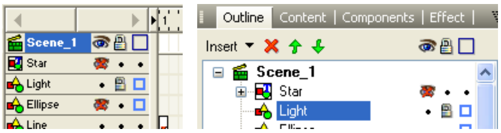


Color picker
Now supports standard colors, library colors, web safe colors and other improvements. See Color Picker.



Effects settings
Effect settings is now a dockable panel. A new Effects browser Panel gives a preview of all effects.

Timeline and Outline Object Attributes
Hide/lock/outline object attributes have been moved from the Outline to the Timeline with separate controls.

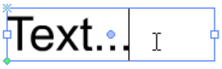


2.2 Text Placement and Editing

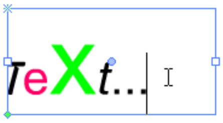
In SWiSH Max version 2 text can now be typed directly onto the stage.

These are the most significant changes to handling text ...

In-place Text editing
Text can be typed directly to the stage. Also click and edit existing text on the stage. See Text Tool.



Per character attributes
Each text character can have different attributes within a single text object. Different color, font, kerning etc. See Text Panel.

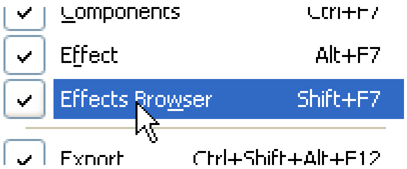


Library text and text assets
Text can be used as a common library resource and even shared across different Flash files (SWFs) in a project.

2.3 Effect Browser

SWiSH Max now includes a panel displaying the installed Effects. Use this feature to insert or replace effects in the timeline during authoring and playback.

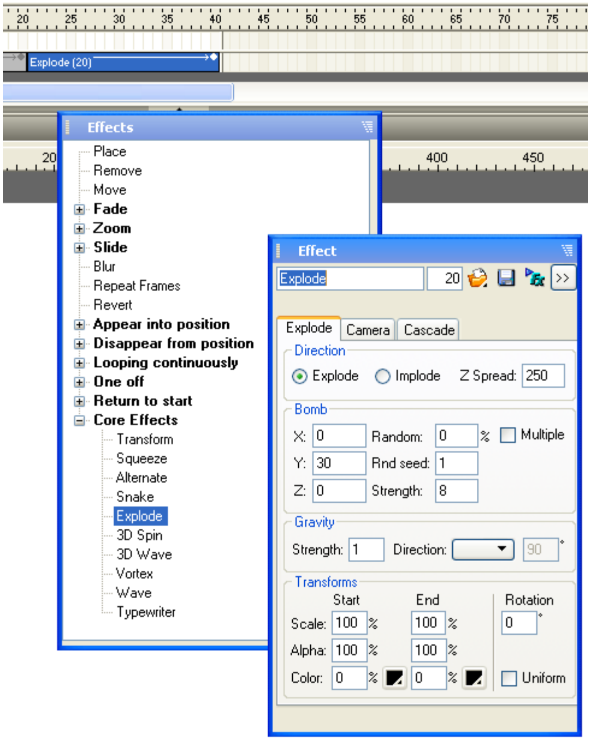
To use this feature check that the 'Effect Browser' is an active window.



When an Effect is applied to an Object the effect is highlighted in the Effect Browser (as shown below).

Click on any frame in the timeline and click on an Effect to insert or replace (an existing Effect) or Alt-Click to insert an Effect.

While a effect (in the timeline) is selected, if the movie to previewed and another effect selected in the Effect Browser the timeline effect will update live and shows the impact alternate effects have on the animation. Note when the preview is stopped use 'Edit | Undo' (Ctrl+Z) to restore the original effect and settings.



Explode 20

Direction

● Explode ○ Implode Z Spread: 250

Bomb

X: 0 Random: 0 % □ Multiple

Y: 30 Rnd seed: 1

Z: 0 Strength: 8

Gravity

Strength: 1 Direction: 90 °

Transforms

	Start		End		Rotation
Scale:	100	%	100	%	0 °
Alpha:	100	%	100	%	
Color:	0	%	0	%	□ Uniform

2.4 Internal Flash Player

SWiSH Max version 2 has a new internal player and will preview any SWF version.

SWiSH Max now uses the Flash Player OCX in the Preview window and consequently SWiSH Max now support internal preview of all SWF versions (assuming the latest Flash Player is installed). This means all scripts will play in the internal player correctly, with trace() statements redirected to the Debug Panel.

2.5 Improved Selection, Editing and Drawing Tools

In SWiSH Max version 2 the drawing tools have been overhauled to make them more intuitive and powerful, and now SWiSH Max follows many of the accepted commands of gold standard drawing applications.

Here are some of the more significant changes to handling drawing tools and object selection ....

Pen tool
The old "Bezier" tool has been rename as the "Pen" tool. The Pen tool now also edits shapes as well as drawing new ones and there a number of key combinations using Ctrl, Alt, Shift to improve drawing efficiency. See Pen Tool.

Keyboard short cuts
Keyboard short cuts have been added for all the drawing tools:
A - Subselection tool
E - Ellipse tool
F - Fill transform tool
H - Hand tool
I - Perspective tool
L - Line tool
M - Motion path tool
N - Pencil tool
P - Pen tool
Q - Transform tool
R - Rectangle tool

T - Text tool
V - Selection tool
Z - Zoom tool
Autoshapes: C - Cube, O - rounded rect., U - heart, W - arrow, D - notched arrow, 2 - two way arrow, S - star, Y - polygon, K - beveled button, B - rounded button

Temporary Select / Hand / Zoom
Selection, Hand and Zoom tools can be temporarily selected whilst using other tools.
Control - selection the Selection Tool.
Spacebar - selects the Hand Tool to drag Objects or Vertices whilst drawing
Control+Spacebar(+Alt) - zooms in (out) on a selected area

Subselection Tool
Multiple points (even from different objects) can be selected, moved, and transformed. See Subselection tool.

Transform and Select Tools
The Selection tool resizes where possible whilst a (new) Transform tool always scales. See Reshape vs. Transform.

Separate Transform and Reshape panel
There are now separate panels for Transforming and Reshaping (resizing). Includes buttons for common operations like flipping. See Reshape vs.Transform

Transformation point (*previously "Rotation" point*)
Each Object has a separate transformation point (for rotation, skew and scale). The Alt key toggles between rotate/skew/scale about the rotation point or opposite corner. See Reference and Transformation Points.

Improved Selection tool options
The modify mode (i.e. resize/scale vs rotate/skew) is determined by the mouse position. Hover directly over a handle to resize/scale, hover slightly outside the handle to rotate/skew. See Drag, Scale, Resize and Rotate/Skew, as well as Cursors and Anchors.

Two Multiple Selection Transform Modes
Every Object transforms about its Transformation point (as it does currently) OR selection transforms as a group (like most other applications). See Reshape/Transform as Group.

2.6 Video, Soundtracks and Images

SWiSH Max version 2 now supports multiple soundtracks, video and image handling.

Here are some of the more significant changes to these areas ...

Supports multiple overlapping soundtracks per timeline
SWiSH Max version 2 removes the previous restriction of one soundtrack per timeline. Sounds can be mixed and converted to a common export format. See Importing Sound.

Supports multiple sound effects per soundtrack
Fade up to level, fade out to level, fade left-to-right etc. See Sound Effects.

Export to AVI and GIF
Improved animation export including animated GIF support. See Export Settings and GIF Animation.

Waveforms

Option to show audio waveforms in the timeline. See Timeline Panel and Importing Sound.

Import common video formats
Video content can be saved in the Library, or imported directly to the stage, or even exported to FLV format. See Import Video... and Convert Video to FLV.

External FLV video
External video objects can be added that loads an external FLV file. It is also possible to add player control components. See External Media component.

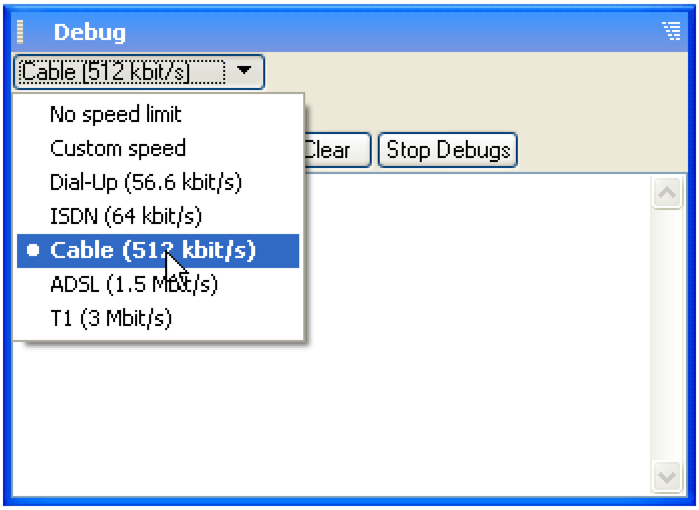
Embedded video in SWF movie with video effects
Implements a video container Object that supports brightness, contrast, saturation, hue, resolution visual effects. See Video effects tab
and Embedded Video Panel.

Bitmaps effects
Including gamma, cropping and effects like blur and smooth. See Appearance Image Properties.

Automatic image compression
Calculate the best compression settings for images. See Export Image Properties.

2.7 Preloader Speed Tester

SWiSH Max version 2 now includes a speed limit option for the internal preview. This is very useful to test preloaders. See Debug Panel.



2.8 Improved Script Editor

SWiSH Max version 2 has an improved Script editor. The Guided and Export modes have been replaced with 'Assist Pane'. The Editor now has many additional features such as syntax coloring, custom keywords and support for external script files. See Script Panel.

2.9 Library and Assets

SWiSH Max version 2 has an improved Content Panel which now includes a Library. The Library is used to accumulate and share common resources throughout the Movie. See Library.

SWiSH Max also now support sharing Objects as Assets. An asset is an Object (in a SWF) which can be shared between separate SWFs when playing on the Flash Player. See Assets.

2.10 Components

SWiSH Max version 2 now supports Components. A number of components are offered with the installer but users can make their own. The Component object is actually a Movie Clip which hides its child objects and script from the user to simplify use or secure the contents. i.e. When a component is added to a movie the child objects and script belonging to the component are not visible in the user-interface. The only way to control the properties of the component is via the Parameters Panel. Examples of components might be: checkboxes and radio buttons, scrollbars and sliders, popup menus and lists, site navigation menus, contact forms, preloaders or media controllers.

Note these new aspects of components ...

Components Panel
Select and place components by drag and drop from component panel. See Components Panel.

Parameters Panel
When a Component Object is selected the Parameters Panel will show a scrollable list of component properties. See Parameters Panel.

Component authoring
Support for components authoring with customizable attributes and scripting. See Authoring Components.

2.11 HTML templates

SWiSH Max now supports HTML templates. Exported HTML can use a custom template which might include Javascript and other information. See File | Export Settings | HTML Template.

Introduction

Chapter 3

3 Introduction

This section provide some basic information about SWiSH Max, what it is and what it can be used for.

How to get started

- Check out What's New in Version 2 for details on the new features introduced since the first release of SWiSH Max.

- See Getting Help for details on using this help and getting more information about SWiSH Max.

- Study the Fundamentals, User Interface and Basic Working Procedures sections for an overview of the program.

- Then work through the Tutorials to familiarize yourself with using SWiSH Max.

Conventions used in this manual

Throughout this document we refer to Menu selection options in the following format - Edit | Copy (Ctrl+C). This means to select the main titlebar 'Edit' Menu and from within it, the 'Copy' option. An equivalent quick key sequence is performed by pressing the Control key (Ctrl) and the C key together.

3.1 Getting Help

There is help for SWiSH Max in this help file, with context-sensitive help in the application, through samples and tutorials both in this document and on SWiSHzone.com and there is also a community forum which can be searched for tips or even to ask new questions.

- **Using this help file**
Use the contents, search and index. Available in a printable form from ***<not available yet>***.

- **Context Sensitive help**
Click on the icon '?' or '?' or use the hot key 'Shift + F1' to display information about a specific topic. Read more about this function in the section on Context-sensitive Help.

- **Tutorials**
Check out the online tutorials section in the manual or the online tutorials.

- **User community forums**
Search the forums or ask a question at forums.swishzone.com.

3.2 Sample Movies

The following samples are included in the Samples folder to illustrate some of the potential of SWiSH Max.

These samples can be viewed using the Menu commands File | Samples. Use the submenu to choose from either Beginner level files, Intermediate level files, Advanced level files, or the Scripting Tutorials.

Beginner

Beginner | effects_demo.swi
SWiSH Max has far too many effects to combine in to one single file, but this sample shows a bit of what the program can achieve without too much effort. It also includes Scenes that demonstrate a Radarscope Effect and a Spotlight Effect produced with a masked Movie Clips and three Vector Objects.

Beginner | analog_clock.swi
Demonstrates the basic usage of the Date object.

Beginner | menu_sample1.swi
This sample uses the Button Object for a basic navigation menu.

Intermediate

Intermediate | conversion_calculator.swi
Performs various calculations through scripted input text fields and variables.

Intermediate | menu_sample2.swi
This menu sample uses a popular method known as the transform slider effect. It demonstrates basic scripting methods used for mouse follower effects.

Intermediate | object_properties.swi
This sample displays the use of a draggable slider bar that controls various object properties, such as transparency, scale, and rotation.

Intermediate | filter_effects.swi
This samples demonstrates the use of scripted SWF8 filters.

Intermediate | form_quiz.swi
This sample demonstrates how to make a quiz with simple components.

Intermediate | ray_of_light.swi
This sample demonstrates how to make a quiz with simple components.

Intermediate | text_scroller.swi
This sample demonstrates the use of a scroller to move a block of text.

Advanced

Advanced | countdown_clock.swi
Demonstrates uses of the Date object. Unlike the analog clock sample that uses basic functions, this sample shows how to perform arithmetic with the date object to create a clock that counts down to a specific date.

Advanced | digital_clock.swi
Another clock, but this time it demonstrates switching between different modes.

Advanced | ballgame_physics.swi
A fun and playful sample of advanced scripting. This sample uses the SWiSH Max specific physics properties and demonstrates collision detection.

Advanced | audio_mixer.swi
A playSound mixer sample demonstrating the control of sounds.

Advanced | filter_effects_generator.swi
A code generator for filters and blends.

Advanced | cartoon_xmas.swi
This sample demonstrates a simple cartoon animation.

Tutorials

Tutorials | Sample_site.swi
This file demonstrates the end result of the "My First Site" screencast tutorial.

Tutorials | Scripting samples files ...
The following files are used in Scripting Tutorials: button.swi, button1.swi, calculator.swi, collide.swi, dragging.swi, droptarget.swi, flowcontrol.swi, game.swi, game1.swi, hellowworld.swi, maxarray.swi, mymoviecontrol.swi, properties.swi, txtmessage.swi, welcome.swi.

Many more samples are available at www.swishzone.com/movies.php.

3.3 Purchasing and Licensing

Most SWiSHzone.com products are protected from piracy by a software licensing system.

Depending on the product, a free trial period is normally allowed. The trial period (if enabled) will give you sufficient time to fully evaluate the terrific value of our products.

To activate your free trial, you will need to obtain a trial Site Key. See the Trial License section for more information.

You can purchase the product before or after the trial period has expired, but once the trial period has expired you will no longer be able to use this product on your computer on a trial basis.

To purchase the product, see the Registration section for more information.

SWiSHzone.com works hard to produce reasonably priced software. In addition to our reasonable prices, there are additional discount options for network-license versions and educational users. Look for details at www.swishzone.com or contact Sales and explain your requirements. There is sure to be a solution to suit your needs.

Please refer to the following sections for assistance:

- Register gives detailed instructions on how to access the Unlock web page. From that page it is possible to purchase and / or obtain Site Keys. An overview of the purchasing and licensing options is presented on our purchase pages.
- Licensing notice explains how to get information on your license and our licensing policy
- Network License can be used if you want to run your SWiSHzone.com product over a network (some products do not offer this facility).

Finally, we have a list of the most frequently asked questions on licensing and the protection system.

3.3.1 Frequently Asked Questions

How do I install on a NT / Windows 2000 or XP system so that the program is available to all users?
See the section Multi User use on XP / Win2000 / NT. Note some products may not support all these operating systems.

How do I update my software?

If an update for the program is available for download or on CD, follow the instructions supplied.
In all cases, please run the installer. Do not copy SZP .exe's directly between machines as this may cause incomplete copies or other errors.

I used to work on two computers (home and office). Can I install my software on both computers?
You are entitled to a License for home and office. Follow the instructions in the Register section to generate a second Site Key for your software. This policy does not apply when a multi user network license has been purchased.

Can I just copy my SWiSHzone program to another computer?
No. The copy protection software will prevent the second copy from working. You can however register two copies according to our home and office policy described above. Run the installation program on the second computer then follow the instructions in the Register section of this document.

Can I run my single user license copy on a network drive?
Yes. However each machine that references the installed program will require a local installation (possibly to the same network drive) and a local license.

What happens to the protection software when my computer HDD crashes and is replaced by a new one?
Follow the instructions in the Register section.

Is a Site key required during the free trial period?
Yes, Follow the instructions in the Trial License section.

What is my Site Code?
See Site Code.

What is my Site Key?
See Site Key.

How can I obtain a Site Key?
Go to the unlock page. This page can be accessed via the "Get Free Trial Key" or "Register" buttons of the 'Unlock' window.

I received the Site Key, but I can not see the difference between the number 0 and the letter O.
Because of the length of the Site Key, we recommend that cut and paste be used to transfer the site key.

Why can't I use the same Site Key multiple times?
The Site Code generated during the installation of the software depends on the computer. A new Site Code is generated if you install the software on another computer. Consequently, you cannot use the same Site Key for another computer. It is the Site Key that permits starting and using the application after the trial period.

For some reason I need to re-install my software. What happens with the Site Key?
If needed, go to the unlock page. This page can be accessed via the "Try" or "Unlock" buttons of the 'Unlock ' window.

I want to license a machine that is not connected to the internet?
Paste the Site Code displayed in the 'Unlock' window to a text file and transfer that file to a computer that is connected to the internet. Access the Unlock web page and use the Site Code from the text file. Paste the resulting Site Key back into the text file and transfer back to the original computer.

I still need help...
If your questions are not answered here, contact your dealer or connect to the internet and use the "Try" or " Unlock" button to access the unlock page. From this web page, follow the Help link.

Please include the following information if available.

- The email address that you used to register your SWiSHzone.com product.
- World pay reciept ID number.
- Operating system e.g. 95 / 98 / ME / NT / Win2k / XP / VISTA

Please also state:

- If you have previously un-installed your SWiSHzone.com product.
- Anything unusual about your system, huge disks, dual processors etc.

3.3.2 Trial License

A trial License allows the user to evaluate the SWiSHzone.com product for a limited time. Typically, the evaluation period will be for 15 days or 45 invocations (which ever occurs first).

Note: Some SWiSHzone products do not come with a Free Trial Period. See here for more information.

If you are already using a trial license and would like to find out about obtaining an unlimited license, see Registration for more information.

When first installed, the product will be unlicensed. (See No License).

To obtain a trial license, connect to the internet and press the "Try" button. This will activate a web page to obtain a trial Site Key that is matched to your Site Code.

Please read the Notes section for important information about when the trial commences, activation time and issues relating to multi user use on NT / Windows2000 / XP machines.

Pasting a valid Site Key into the edit box will cause the Site Key OK dialog to be displayed. Pressing Continue will activate the product.

While using a trial license, a 'Unlock' window will be displayed during startup. This window allows displays the current trial license status and gives opportunity to purchase and register the product. See Registration for more information.

Note: See the section Multi User use on XP / Win2000 / NT if you are installing on one of those operating systems and require multiple users to have access to the program. Although the trial can only be used by the person who installed it, it should be installed by an administrator account if you require multi user access in the future.

3.3.2.1 Notes

- Some SWiSHzone.com products are not available for trial use. See here for more information.

- Only 1 Trial Key will be issued to each computer. Support will NOT issue replacement trial keys.

- The trial period commences when the Site Key is first activated (pasted into the Site Key edit box).

- The available invocations will only decrement if the SZP is used. Pressing the cancel (x) button of the Unlock Window will allow viewing of license status without running the SZP and subsequent decrement of the available invocations.

- The Trial Site Key should be activated within 2 days. There is a timeout period, after which the site key will become invalid.

- Ensure that the date on your computer system is set to the correct date when activating the site key. Dates set in the future or in the past will invalidate the site key.

- When installed on a NT / Windows2000 / XP system, only the user that installs the SZP will be licensed to trial it. Other users attempting to run the installed product on the same computer using a different login will receive a status number 1011 indicating that the product is unlicensed. This WILL INVALIDATE the Trial Key for ALL users. (This limitation only applies to trial licenses, not Unlimited Site Keys.)

- See the first note above. Trial Keys will not be re-issued because of failure to follow the above suggestions.

3.3.2.2 Products Without Free Trial

Some SWiSHzone.com products do not come with a free trial period.

The Product Splash Image will indicate if a free trial period is available for the product you are using. If a trial is not available, the "Try" button will not be visible.

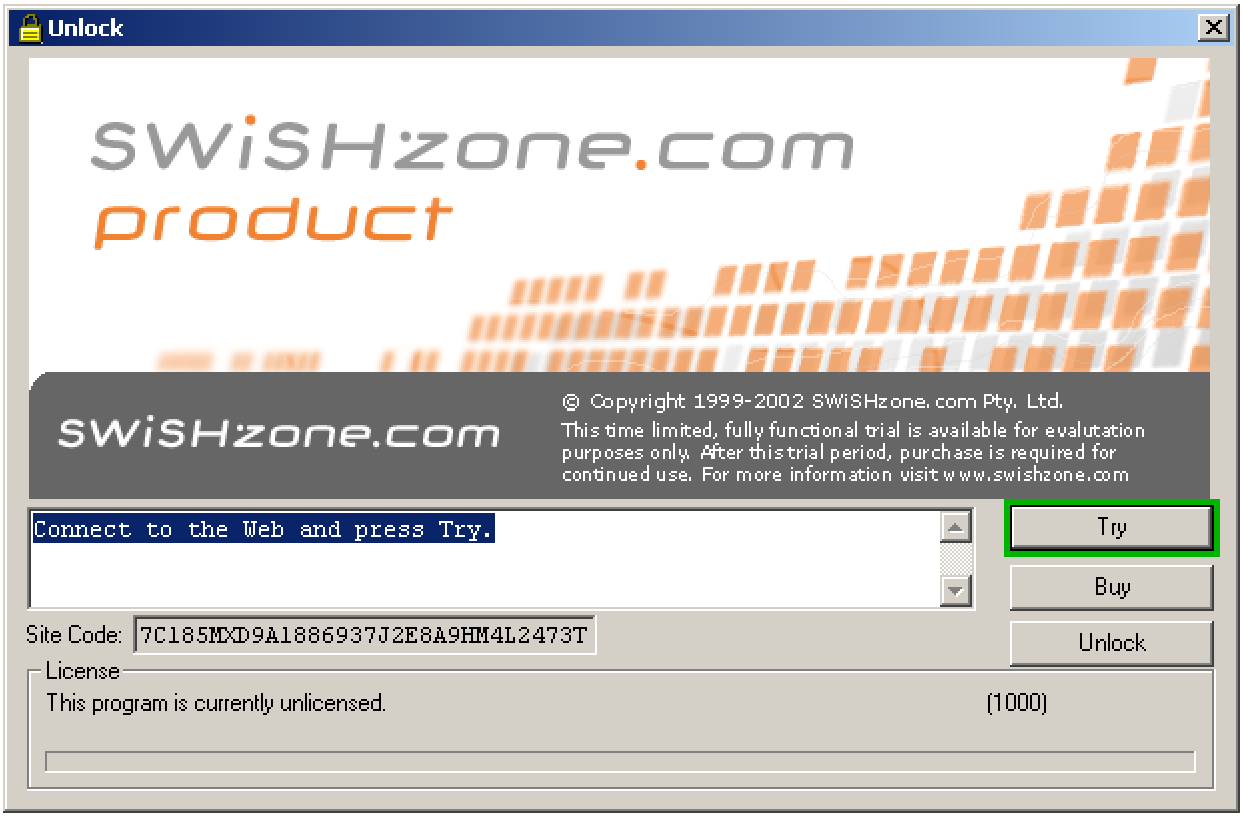
The following SWiSHzone products do not have a free trial period:

- SWiSH Max templates
- SWiSH Lite (Discontinued Product)
- SWiSH2 (Discontinued Product)
- SWiSH PowerFx (Discontinued Product)
- Non English versions of products.

Note that while discontinued products are still available for download to licensed customers, they are unavailable for purchase or trial.

3.3.2.3 No License

After installing the product, one of the following windows will be displayed if no previous license was found.



© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This time limited, fully functional trial is available for evalutation purposes only. After this trial period, purchase is required for continued use. For more information visit www.swishzone.com

```
Connect to the Web and press Try.
```

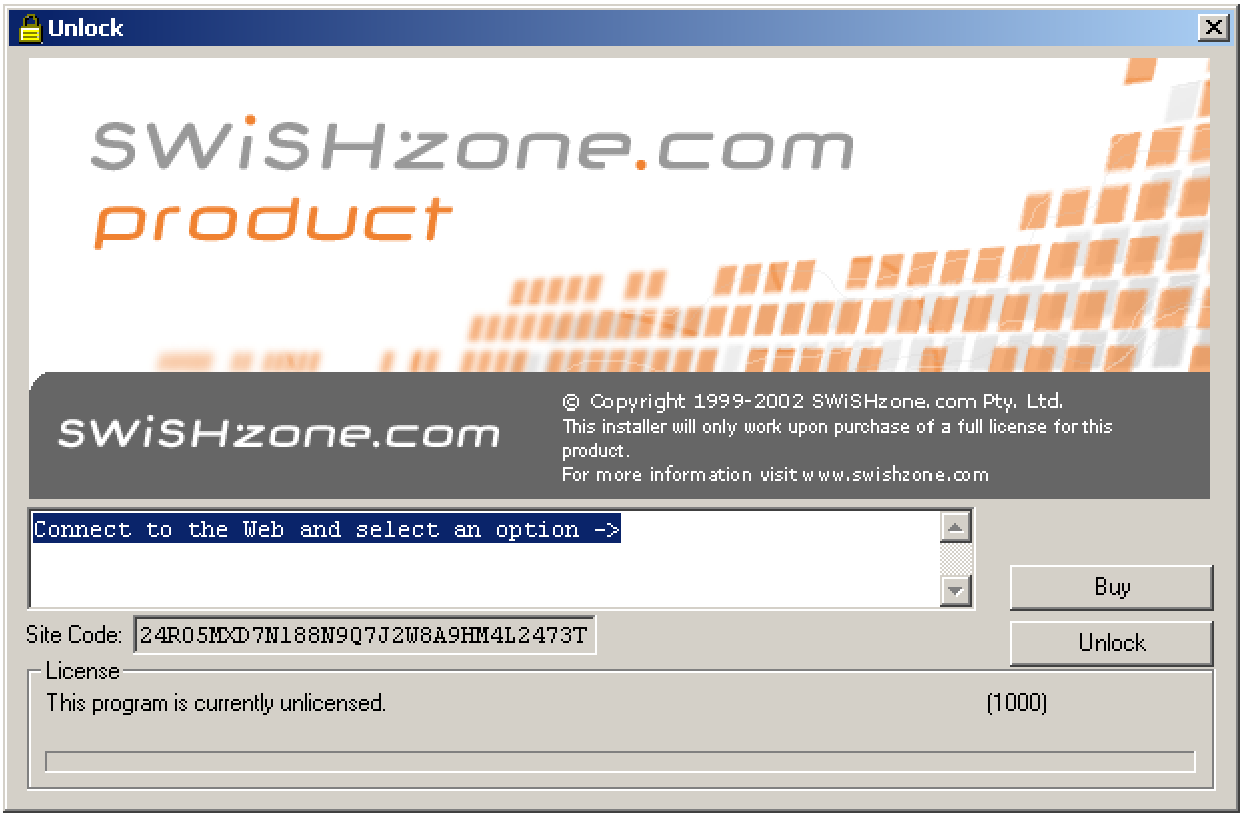
Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

License

This program is currently unlicensed. (1000)

This window above indicates that a trial license is allowed for the product.
To obtain your trial license you will need to:

- Connect to the internet.
- Activate the unlock web page via the "Try" button.
- Paste the Site Key that was obtained from the web page into the Site Key Edit Box.



© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This installer will only work upon purchase of a full license for this product.
For more information visit www.swishzone.com

Connect to the Web and select an option ->

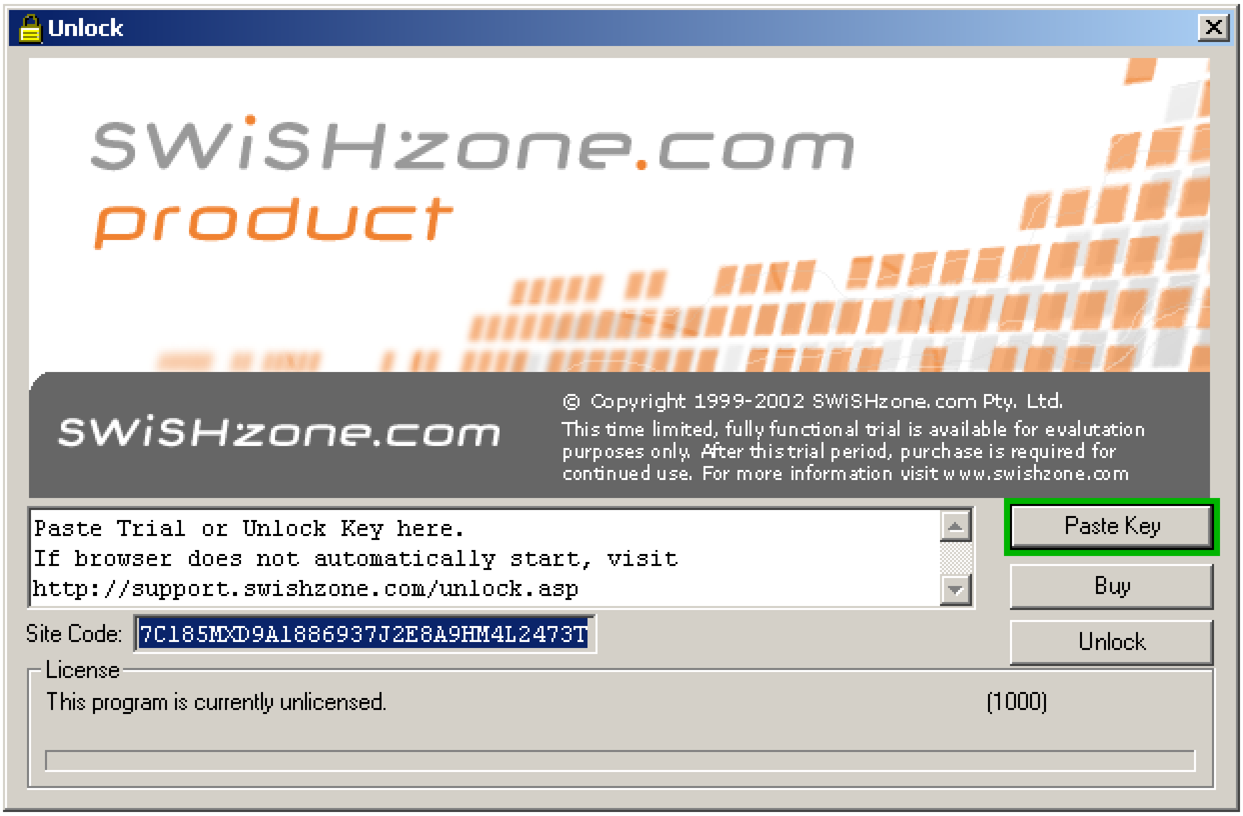
Site Code: 24R05MXD7N188N9Q7J2W8A9HM4L2473T

License
This program is currently unlicensed. (1000)

The window above indicates that a trial license is not available for the product.
If you have already purchased the product, you can unlock the product by pressing the "Unlock" button.
Once you have recieved your Site Key, use copy and paste commands to Paste it into the Site Key Edit Box.
If you have not purchased the product, you can access our web based purchase pages by pressing the "Buy" button.

3.3.2.4 Paste Unlock Key

Once a Site Key has been copied to the clipboard, Button B1 will change to a "Paste Key" button.



© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This time limited, fully functional trial is available for evalutation purposes only. After this trial period, purchase is required for continued use. For more information visit www.swishzone.com

```
Paste Trial or Unlock Key here.
If browser does not automatically start, visit
http://support.swishzone.com/unlock.asp
```

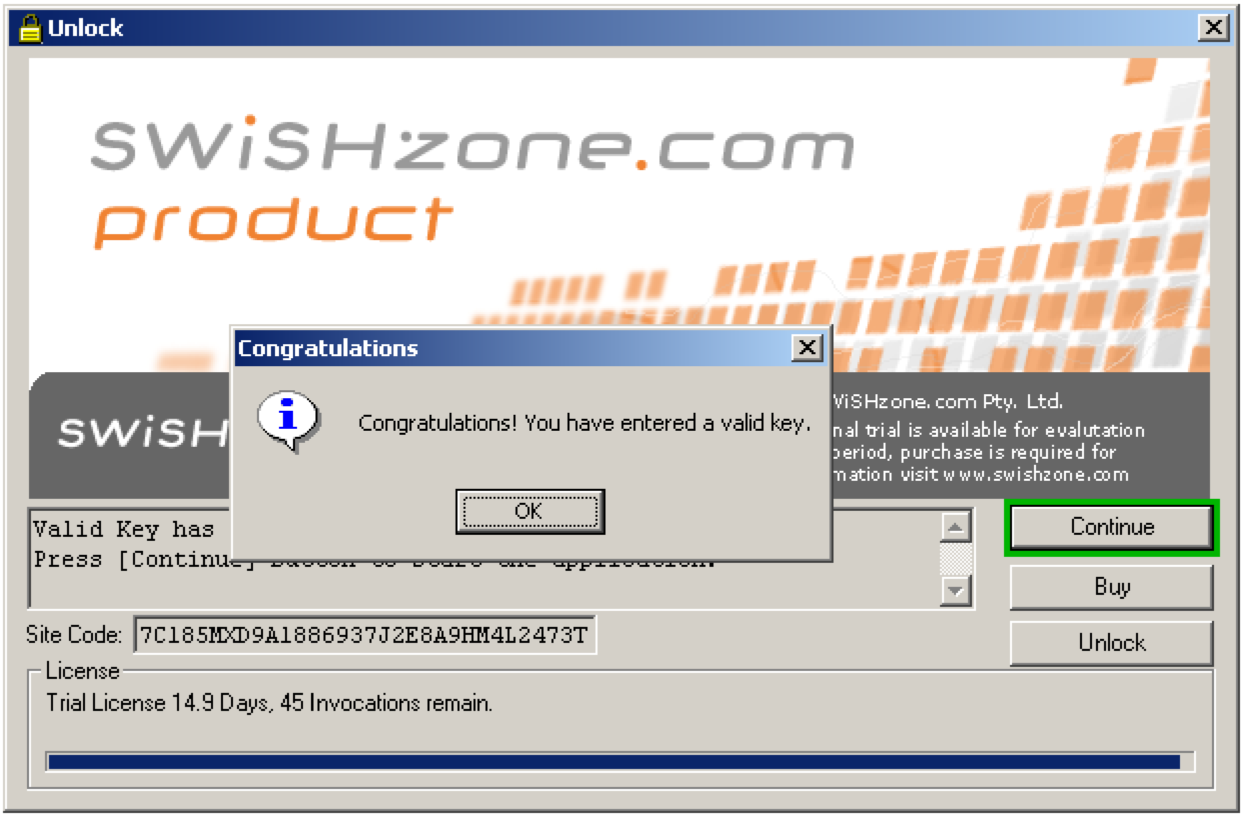
Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

License
This program is currently unlicensed. (1000)

The Site Key can be pasted by pressing the "Paste Key" button or using standard paste procedure so that the Site Key is pasted into the "Site Key Edit Box".

3.3.2.5 Site Key OK

If the Site Key is valid, the following window will be displayed:



Unlock
SWiSHzone.com
product
SWiSH
ViSHzone.com Pty. Ltd.
nal trial is available for evalutation
period, purchase is required for
mation visit www.swishzone.com
Continue
Buy
Unlock

Congratulations

Congratulations! You have entered a valid key.

OK

Valid Key has
Press [Continue] button to start the application.

Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

License

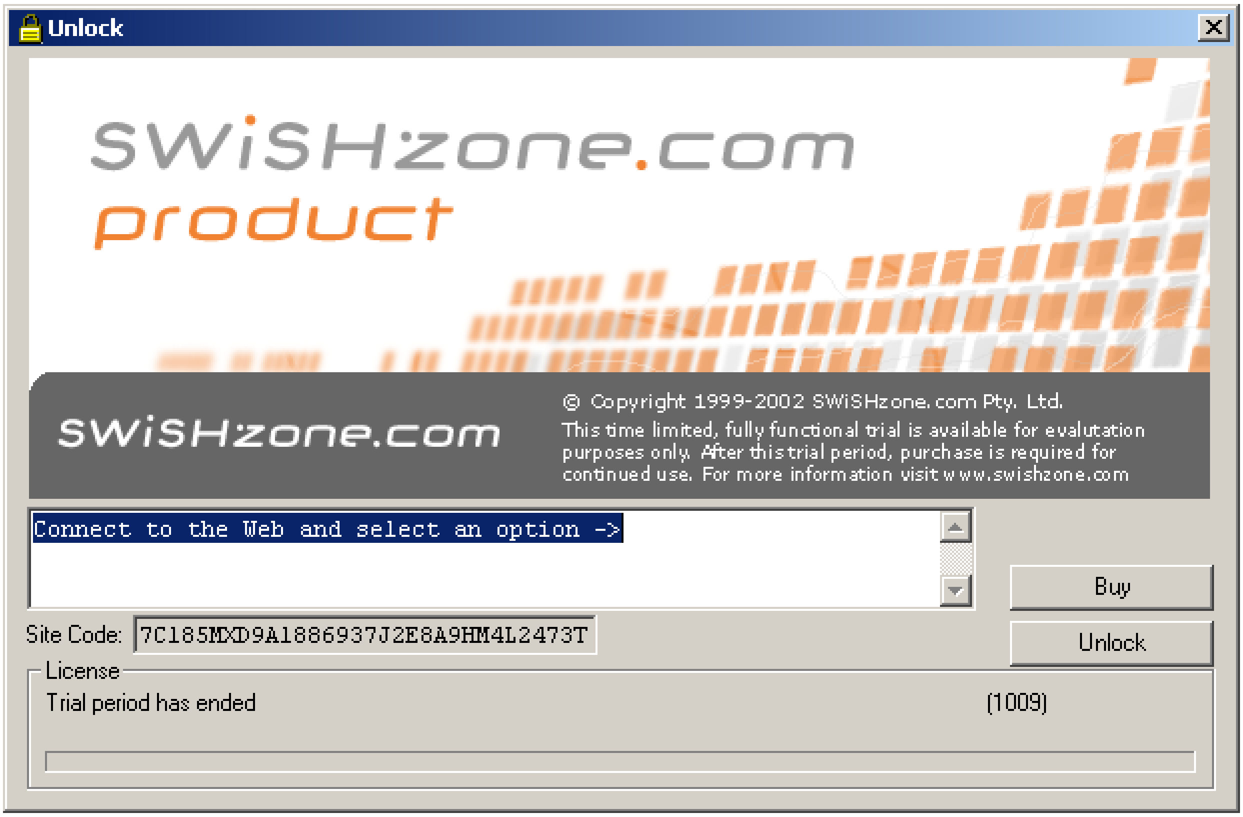
Trial License 14.9 Days, 45 Invocations remain.

Press OK to remove the Congratulations dialog. The appliation will then start.
On subsequent invociations, press the Continue button to use the application.

3.3.2.6 Trial has ended

After the trial has ended the Unlock Window will appear as shown below.
Note that the Get Free Trial / Continue button is missing (no longer entitled to free trial).

To continue using the SZP, use the "Buy" button to purchase then obtain an Unlimited Site Key using the " Unlock" button.



© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This time limited, fully functional trial is available for evalutation purposes only. After this trial period, purchase is required for continued use. For more information visit www.swishzone.com

Connect to the Web and select an option ->

Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

License
Trial period has ended (1009)

3.3.2.7 Licensing Notice

Unless registered, the Product will cease to function after the trial period indicated on the start-up splash screen. No data is lost or destroyed when the trial period expires. Licensed (Registered) users can unlock their software on a permanent basis anytime during or after the trial period. Detailed information on unlocking the product can be found in this manual.

The 'Unlock' window CANNOT be accessed while the Product is in use. The 'Unlock' window will be displayed each time the product starts if using a Trial License. The 'Unlock' window will also be displayed if the license has expired or been made invalid for some reason.

The CD-ROM or download installers DO NOT self-destruct after the application is installed. They can be used to install the Product on an infinite number of PCs. However each PC will require either a trial or unlimited license to run. We encourage potential customers to install the product on multiple PCs for evaluation purposes. Keep in mind, however, that you must purchase a license for each PC you wish to run the Product on a permanent basis.

Variations from the aforementioned policy will be at the sole discretion of SWiSHzone.com Pty. Ltd.

3.3.3 Registration

To remove the time or runs limitation of the trial license, or to re-activate the product once the trial period has expired, it is necessary to purchase an unlimited license and install an Unlimited Site Key.

When the product runs using a trial license, the dialog in the Start up with trial license section is displayed. This dialog allows the user to see the current license status.

You can access the purchase pages using the "Buy" button. Once you have purchased a license, that license will be associated with your email address. Using your email address, you can then generate an Unlimited Site Key which will permanently enable the product.

To purchase the product, use the "Buy" button to access our web based shopping pages.

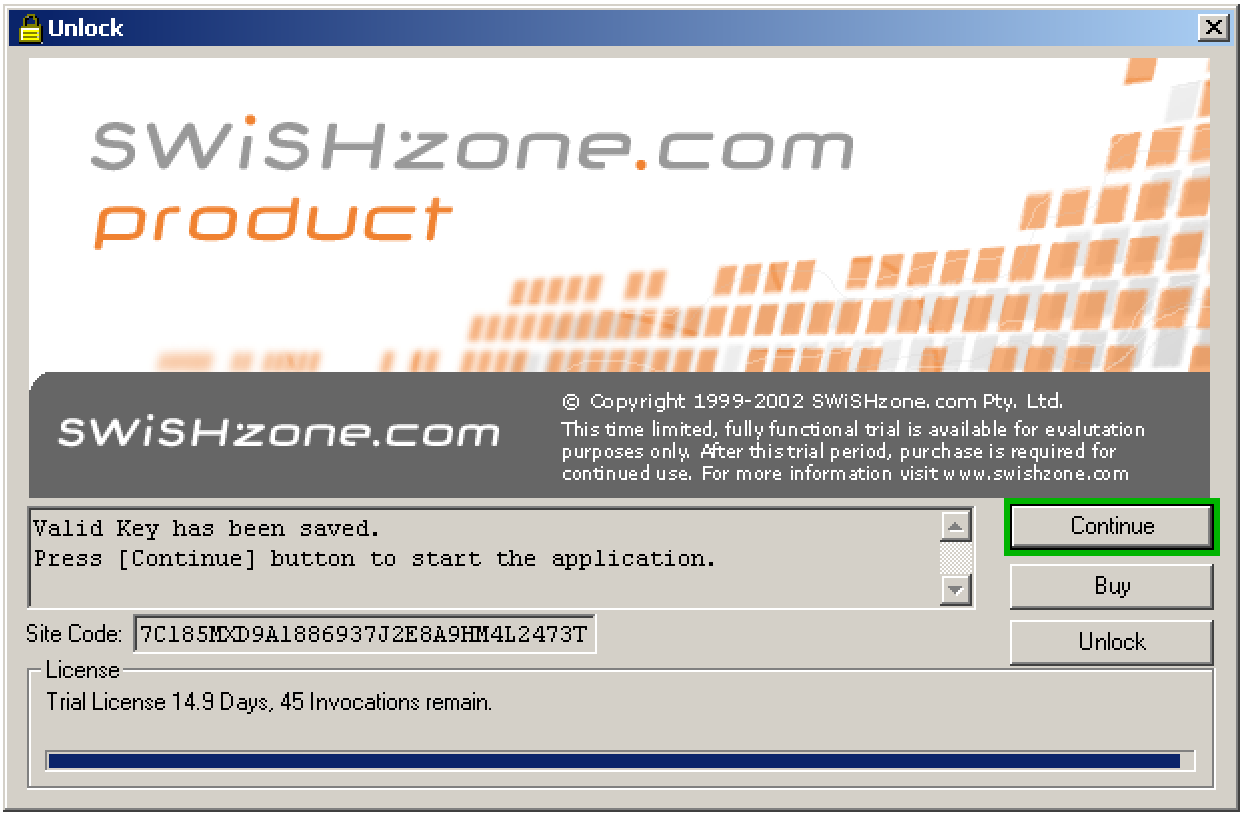
To Unlock the product, use the "Unlock" button to obtain an Unlimited Site Key. This can also be used to re-license the product after some error.

See the Register section for more information.

Note: See the section Multi User use on XP / Win2000 / NT / Vista if you are installing on one of those operating systems and require multiple users to have access to the program.

3.3.3.1 Start up with trial license

While running with a trial key, the following dialog will be displayed each time the product starts. Press the Continue button to use the application.


Unlock
SWiSHzone.com
product
SWiSHzone.com
© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This time limited, fully functional trial is available for evalutation purposes only. After this trial period, purchase is required for continued use. For more information visit www.swishzone.com
Continue
Buy
Unlock

```
Valid Key has been saved.
Press [Continue] button to start the application.
```

Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

License
Trial License 14.9 Days, 45 Invocations remain.

The dialog also shows the time and runs remaining on the trial license.
To purchase an unlimited license, connect to the web and press the Register button.

3.3.3.2 Unlock

Press the "Unlock" button of the 'Unlock' window to open a browser window that attempts to connect to the Unlock web page. This will also re-enable the Site Key Edit Box allowing a new site key to be pasted.

If connection is successful, the page will request the Site Code and the registered email address.

If you accessed the unlock page via the "Unlock" button, your current site code should have been automatically entered into the unlock page. This can be ammended if necessary.

If the registered user owns a valid license, a site key will be created and emailed to the registered email address.

3.3.3.3 Notes

When using NT / Windows2000 / XP, installation of the SZP and subsequent activation of the Unlimited Site Key by an administrator account will allow all uses of the system to use the SZP on that computer.

Installation and activation of the SZP by non adminsitrator accounts is possible. However, the license will only be useable by that user.

Upgrading the operating system, changing hardware or reformatting disk drives may cause a previously licensed system to become unlicensed. Follow the procedure in Register to obtain a new key.

3.3.4 Problems

If you encounter problems during the licensing / unlock process, connect to the web and press the "Try" or " Unlock" buttons to access the unlock page. From this web page, follow the Help link.

This is the preferred method of accessing the help page as it will transfer important information about your Site code and error number to our support staff.

Alternatively, you can contact our support staff via http://support.swishzone.com/support.asp

3.3.5 Network License

SWiSHloc supports network licenses.
This type of licensing requires a network server which should be running NT, Windows2000 or XP.
Client machines can run 95, 98, NT, Windows2000 or XP.
The network must support TCP/IP.

This type of setup is normally very attractive for large office or teaching institutions as the site key used by each of the client machines is identical.

The cost of a network license is identical to the equivalent number of single user licenses. Volume discounts still apply.

Detailed installation instructions and various network tools are included with the network installation package.

Because of the general complexity in setting up network solutions, the network installation package is not available automatically from our web site. Please contact Support with your requirements.

After analysing your requirements, If a network solution sounds like a useful solution, we will send you the network installation package.

Network licenses will not be supplied for sites with less than 5 seats (licensed users).

3.3.6 Multi User use on XP / Win2000 / NT

SWiSHzone.com products can be installed on a NT/Win2000/XP/VISTA so that they are available to all logins. Note this only applies to full (non trial) unlock keys, and some products do not install on (some of) these operating systems. A trial key is only available to the user who first unlocked the program.

To allow all users access to the program, the following steps must be followed:
1. The original installation and unlock must be done by an administrator account.

2. The install directory must be shared so that other users can read the files. (Write permission is not required). Individual users must not refer to their own installed area as this will cause problems when installing updates.

If a user that did not have administrator privileges originally installed the application, please contact support for more information.

3.3.7 Definitions

3.3.7.1 SWiSHloc

SWiSHloc is the copy protection system used to protect a number of SWiSHzone Products.

3.3.7.2 SWiSHzone Product

SWiSHzone.com produces a number of products. For the purposes of this document, a SWiSHzone Product (sometimes referred to as **SZP**) is any product that uses the SWiSHloc method of copy protection.

3.3.7.3 Registered Email Address

The **Registered Email Address** is the email address that was used when a Registered User purchases the product.
The registered email address is used to track registered users on our database. The registered email address is also used to determine license ownership and consequently associated Site Key generation permissions.

3.3.7.4 Registered User

A **Registered User** is a user that has a valid license for the SZP. Licenses can be obtained by purchasing the product.

A registered user with a valid license can visit the Unlock web page to obtain an Unlimited Site Key that will allow them to unlock their product on a permanent basis.

3.3.7.5 Site Code

A **Site Code** is used to identify a computer.

The Site Code is a character string that is displayed in the 'Unlock' window.

An example site code is

```
{XXHDA7QK6NMVUK9BSC5YN4HT49DQKGBRGAGERV5D3NNY6QPR8X
7XBJKVQSB9LJJK772WUTYMF9YAG7XUQKQRSZ8KGKHLEK7QXN574
TKA44R3T4VWDNCE-7AJH3WTAJGA3LDN5B2WWM58GXFVH5LBU77}
```

A corresponding Site Key available from our Unlock web page will unlock the application.

3.3.7.6 Site Key

A **Site Key** is an encrypted string that is created to match the Site Code.
The Site Key is often referred to as the "Unlock Key". It contains information about the particular type of license a user owns.
A Trial Site Key is a special key that allows timed use of the product.
Once license ownership has been verified, the program is unlocked according to the license constraints.

The Site Key is a long character string enclosed by {}.
An example Site key is

{SH44VGGR9R66TFK2WETCAWCSK7SBKZQTUS44UUWEYBNY7GK9MTMG6DJWHP
WCFF72BJMWBR3PZLMQ6C8KJ55UQN9B7A872WWS82WCMN4VJCBLBJNMTJ4X}

When pasting the Site key, include the opening and closing {}.

3.3.7.7 Site Key Activation

The process of pasting a valid Site Key into the Site Key Edit box that is displayed in the Unlock window.

3.3.7.8 Trial Site Key

A **Trial Site Key** is a key that is free issued to computers that are not shown as having previously been issued with a trial key.
The Trial Site Key limits program use according to both time period and invocations. (Typically 15 days / 45 invocations, which ever occurs first). Otherwise, the program is fully functional.

See the notes section for other special requirements associated with trial keys.

3.3.7.9 Try Button

This is button B1 of the Unlock Window. Note that this button is only visible after a fresh installation of the product.
Use this button to open the Unlock Web Page and obtain a Trial Site Key.

When this button is used to open the Unlock Web Page, your Site Code will be sent to the website eliminating the need to re-type the Site Code.

In some cases, the button may fail to function correctly because it is unable to determine the location of a default web browser.
The computer should be connected to the Web prior to pressing this button.

Note: Some SWiSHzone products do not come with a Free Trial Period. See here for more information.

3.3.7.10 Unlock Button

This is button B3 of the Unlock Window. Use this button to open the Unlock Web Page and obtain a Unlimited Site Key.

When this button is used to open the Unlock Web Page, your Site Code will be sent to the website eliminating the need to re-type the Site Code.

In some cases, the button may fail to function correctly because it is unable to determine the location of a default web browser.
The computer should be connected to the Web prior to pressing this button.

3.3.7.11 Unlimited Site Key

An **Unlimited Site Key** is a Site Key that indicates that the user owns an unlimited license for the product. When this Key is entered, the trial key time / invocations limitations are removed.

Once an unlimited Site Key has been entered, the Unlock dialog will no longer appear unless some error or machine re-configuration invalidates the license.

3.3.7.12 Unlock web page

The web page http://support.swishzone.com/unlock.asp can be used to obtain Trial and Unlimited Site Keys.

This page can be accessed from the Unlock Window via the "Try" or "Unlock" buttons.

If possible, access the page using the "Try" or "Unlock" buttons as this will automatically transfer your site code to the web page.

Note: Unlimited Site Keys are only available to Registered Users and the Unlock web page will require your Registered Email Address to produce one of those Keys.

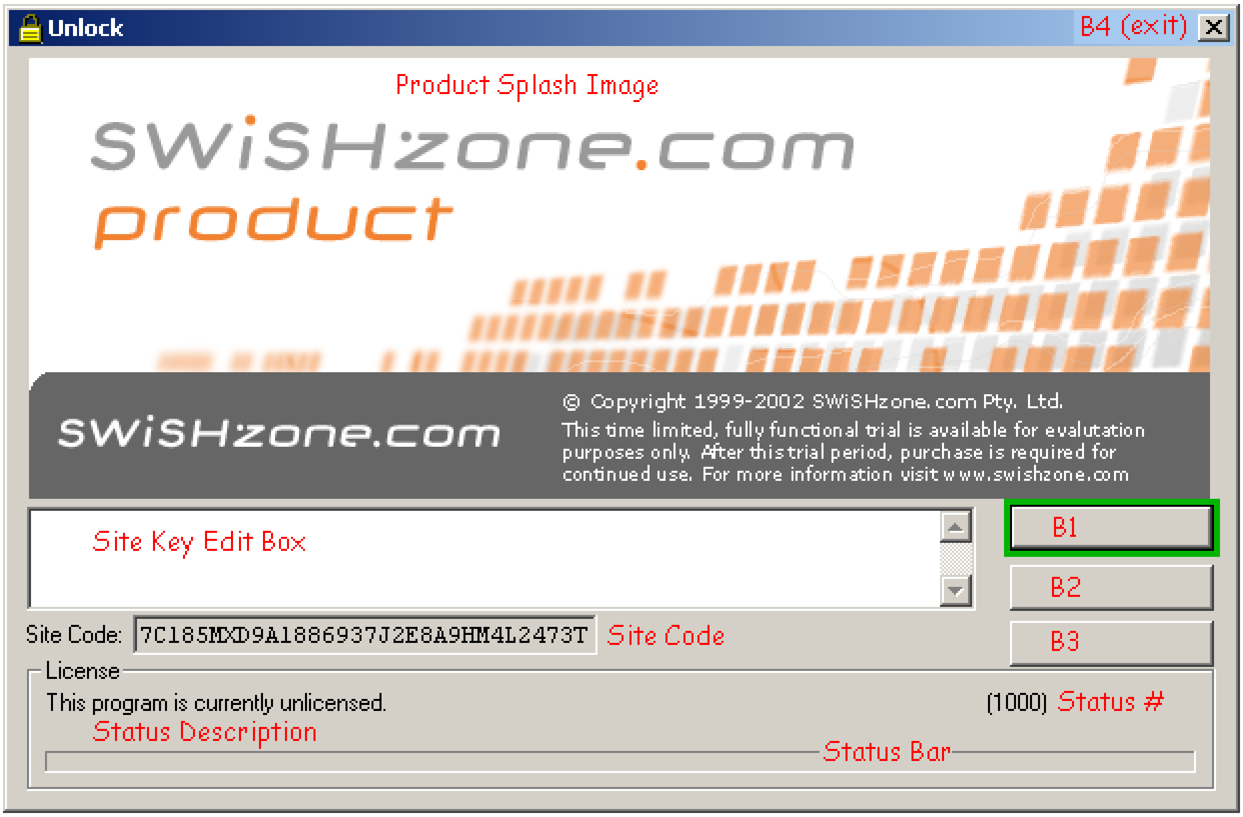
3.3.7.13 Unlock Window

This is the main SWiSHloc dialog window.

This window provides allows you to:
Unlock your SZP by pasting an appropriate Site Key into the Site Key Edit Box.

- View the current license status (If license is valid, press "Continue", button (B1) to start the SZP.)
- Obtain a Trial Site Key via the "Try" button, (B1).
- Purchase the application via the "Buy" button, (B2).
- Obtain an Unlimited Site Key via the "Unlock" button, (B3).

If the product is using a trial license or is currently unlicensed, the Unlock window will appear each time the SZP is started.


Unlock
B4 (exit)
Product Splash Image
SWiSHzone.com
product
SWiSHzone.com
© Copyright 1999-2002 SWiSHzone.com Pty. Ltd.
This time limited, fully functional trial is available for evalutation purposes only. After this trial period, purchase is required for continued use. For more information visit www.swishzone.com
Site Key Edit Box
B1
B2
B3
Site Code:
Site Code
License
Status #
Status Description
Status Bar

Site Code: 7C185MXD9A1886937J2E8A9HM4L2473T

This program is currently unlicensed.

(1000)

Product Splash Image
This image identifies the product. The text indicates if a trial period is normally available for the product.

Site Key Edit Box
This Edit Box allows the pasting of Site Keys. After a valid site key has been entered, the edit box will become disabled.

Site Code
The site code for the product and your computer is displayed here.

Computers that are not connected to the internet can be licensed by transferring the Site Code (use cut and paste) to a text file which is then transferred to another computer that has internet access. From the internet connected computer, access the Unlock web page and obtain a Site Key using the Site Code from the text file. Paste that Site key to a text file and transfer back to the original computer.

Status Description
The current license status:
Unlicensed, Trial License, Unlimited etc.

Status #
A code indicating the current license status. Support staff will find this number useful.

Status bar
A graphical bar that indicates the % of time / runs available with a trial license.

B1 (Try / Paste / Continue)
If the product is unlicensed and this is the first installation, the button will have the caption "Try". Press this button to access the Unlock web page to obtain your free trial key.

Note: Some SWiSHzone products do not come with a Free Trial Period. See here for more information. In this case the "Try" Button will not be visible.

If a Unlock Key has been pasted to the clipboard, the button will change to a "Paste" button, allowing the pasting of the Unlock key.

If the product is licensed, the button will have the caption "Continue". Pressing this button will start the SZP.

B2 (Buy)
Use this button to contact access our electronic store. The user should be connected to the Web before using this button.

B3 (Unlock)
Press this button to obtain an Unlimited Site Key.
Pressing this button will re-enable the Site Key Edit Box allowing a new key to be pasted.

B4 (x)
This button will allow exit without starting the SZP. If you are using a trial license, your invocation count will not be affected. Note that the time period limitation commences once you have obtained your trial site key.

3.3.7.14 Unlock Key

This is another name for Site Key.

3.4 Requirements and Credits

Minimum system requirements

- Windows 2000/XP/VISTA
- 1024 x 768, 16-bit display (32-bit recommended)
- SWiSH Max requires the latest Adobe Flash Player installed on the system for internal preview.

SWiSH Max is a Adobe Flash-enabled application. SWiSH Max incorporates the Flash Player which enables it to preview .swf animations without launching an external player or browser.

Credits

Many thanks to the hard working developers on the SWiSH Max project: Damien Skracic, Hung-Hsin Chang, Roger Onslow, Gus Nalwan, John McKee, Cameron Browne, Olivier Debon and David Michie.

Copyrights and Trademarks

Flash® is a registered trademark of Adobe Inc.
SWISH, SWiSH Max and SWiSHzone.com are trademarks of SWiSHzone.com Pty. Ltd.

Fundamentals

Chapter 4

4 Fundamentals

This section presents an explanation of many of the main ideas and features within SWiSH Max. It is worthwhile reviewing all these sections to become acquainted with the terminology and technology used to make SWiSH Max's Flash Movies.

➡ Read Flash Movies to get an explanation of the structure of the Flash animation and how the parts (frames, scenes, objects etc.) all fit together.

➡ Read Objects to learn about the types of objects (shapes, text etc.) are used to create the Movie and how these are used.

➡ Read Tools to get an understanding of Tools which can be used to manipulate the objects within SWiSH Max.

➡ Effects describes the types of effects which can be applied to the Objects in SWiSH Max and how to make custom effects.

➡ Components are reusable, packaged modules that adds a particular capability to a Flash movie like a check box or list box. This section gives an overview of how to use, and make components in SWiSH Max.

4.1 Flash Movies

This section describes the basic elements of SWiSH Max animation.

The animation created is called a Movie. Within each Movie is a collection of Scenes. Each Scene has a Timeline consisting of multiple Frames.

During the Timeline of each Scene, place Objects (e.g. text, images, etc.) to which Effects can be applied. These Effects will start and stop at particular Frames and can be controlled by inserting Actions and Events.

In Scenes, Events occur when the Movie reaches a certain Frame. For objects, Events occur when the user interacts with an object using the mouse, such as rolling the mouse cursor over the Object or clicking on it.

An Event can trigger more than one Action. For example, when the mouse rolls over an Object, the Movie can be stopped (with a Stop Action) and the browser can be told to load an URL into another Frame (with the getURL Action).

Actions are operations that are triggered by Events. Actions can alter the playing of the Movie, start or stop sounds, load other Movies or web pages, or communicate with the host browser or player.

Motion picture analogy

To gain a better understanding of the meaning of terms like Movie, Frame, Framerate, Timeline, Objects, Actions and Events think of a motion picture film, for example 'The Godfather'. This movie has many different scenes, for example the wedding scene and the horse head in the bed scene. These Scenes may overlap slightly but are generally separate and as one scene ends another starts.

A movie projector is a device that continuously moves film along a path so that each frame of the film is stopped for a fraction of a second in front of a light source. In SWiSH Max we also refer frames and the speed of the film is the frame rate (measured in frames per second). If the film it laid out on the table each frame can be seen side by side - this is represented by the Timeline.

Now imagine the film scene where a character shoots a gun. This scene runs across a number of frames. In the scene the gun (for example) is an object which can perform an action like firing a bullet.

Unfortunately the motion picture is not interactive but a Flash animation is. So imagine the gun can be clicked with the mouse. This click causes an event which creates an action, again like firing a bullet.

4.1.1 Movies

A Movie is a sequence of Scenes. A Movie contains all the Scenes, Objects, Effects, Events, and Actions that comprise the final animation.

Properties of a Movie (including size, Frame Rate and background color) can be edited in the Movie Properties.

A new Movie can be created by pressing the 'New' button on the Standard Toolbar, or by selecting New from the File Menu. Scenes can be added to a Movie by pressing the 'Insert Scene' button on the Insert Toolbar, or by selecting Scene from the Insert Menu.

Movies can be saved to disk (as a .swi file) by pressing the 'Save' button on the Standard Toolbar, or by selecting Save from the File Menu. Existing Movies can be read from disk by pressing the 'Open' button on the Standard Toolbar, or by selecting Open from the File Menu.

Movies can be exported in the formats: Shockwave Flash (.swf), video (.avi), Projector EXE (.exe), GIF animation (.gif), PNG file (.png). Select an export format choice by selecting Export from the File Menu.

Note: SWiSH Max cannot read or write .fla files because .fla is a proprietary format owned by Adobe.

4.1.2 Scenes

A Scene is a collection of Objects that are animated over a number of Frames. When a Scene is complete, all the Objects are removed from the display and the Movie automatically moves to the next Scene.

A new Scene can be created by pressing the 'Insert Scene' button on the Insert Toolbar, or by selecting Scene from the Insert Menu. Objects can be added to the Scene by pressing one of the buttons on the Insert Toolbar, by selecting options from the Insert Menu, or by using one of the drawing tools in the Toolbox on the Layout Panel.

If no Object in the Scene is selected, the Object Panel displays the Scene Panel. Use the 'Scene' Panel to set the Scene's name, background color, as well as an URL to link to when the Scene is clicked.

Each Scene has its own Timeline which can be edited using the Timeline Panel. Use the Timeline to coordinate and combine animations of different Objects.

4.1.3 Timeline and Frames

Each Scene is made up of a series of Frames, in the same way that a motion picture is made up of Frames. The Timeline Panel shows a visual representation of the Frames, with the first Frame at the left and last Frame at the right.

The top row shows the Frame Actions for the Scene. These Actions are executed when the Movie reaches the Frame where the Action is located. Frame Actions always have a duration of one Frame, but it is possible to execute more than one Action in a single Frame.

The rows below the Scene row represent the Objects in the Scene. The rows are displayed in stacking order, with the Object that is in front of all other Objects displayed at the top (just below the Scene row), and the Object that is behind all other Objects displayed at the bottom.

Each Object row shows the Effects that are applied to that Object. Effects may have a duration of one or more Frames, but only one Effect can be applied to an Object at any given Frame.

The amount of time a single Frame is displayed is controlled by the 'Frame Rate' field on the Movie Properties dialog box. The Frame Rate is measured in Frames-per-second. A high Frame Rate value (e.g.

20) will produce a fast-running Movie. A low Frame Rate value (e.g. 1) will produce a slow-running Movie. The Frame Rate can be set to variable or constant. A variable Frame Rate means that the value entered is a maximum rate only. The speed of the Movie is limited by the power of the CPU playing the Movie and may vary from Frame to Frame depending on the complexity of the animation. A constant Frame Rate means that the Movie will maintain the value entered constantly throughout the Movie. To keep a constant Frame Rate the Movie may need to skip Frames. A constant Frame Rate is useful when keeping a Movie running in synchrony with a sound file.

Note: A Scene contains one Frame until an Effect or Action is added

4.2 Objects

SWiSH Max supports two types of Objects within a Scene:

- **Simple Objects**
- **Complex Objects**

The Object type defines how the Object behaves when various Effects are applied to it. Some Effects behave differently depending on the type of Object (Simple vs Complex).

A **Simple Object** is a single indivisible entity and is animated as a single object, unlike Complex Objects, in which the individual shape or objects contained within it are animated separately.

The following Objects are Simple Objects:

- Shape Objects
- Button Objects
- Movie Clip Objects
- Text Objects (input)
- Text Objects (dynamic)
- Text Objects (static with 'Target' checkbox ticked)
- Group Objects (named with 'Target' checkbox ticked).
- Sound Object (Soundtrack)

- Embedded Video Object
- External Media Object

A **Complex Object** is made up of many component Objects, unlike a Simple Object which is treated as an indivisible Object.

Some of the Authored Effects when applied, will apply to each of the component Objects individually. For example, if the Explode Effect is applied to a Complex Object, each of the component Objects will follow a different trajectory. If the same Effect is applied to a Simple Object, the Object will follow a single path.

The following Objects are Complex Objects:

- Text Objects (static text with 'Target' checkbox unchecked)
- Group Objects ('Target' checkbox unchecked).

4.2.1 Shape Objects

A Shape Object is a Simple Object defined by a list of curved or straight edges. It can contain multiple closed or open figures. The figures can have lines around their edges and can be filled with solid colors, gradients or images.

Also see Shape Object Properties Panel.

Shape Objects can be created using the following drawing tools:

Line

Pencil

Pen

Rectangle

Ellipse

AutoShapes.

You can also produce shapes by using Group as Shape or Convert to Shapes, or by importing vector or raster images.

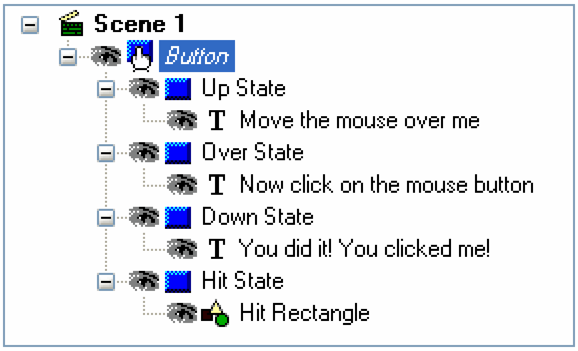
Use the Subselection, Perspective and Fill Transform tools to modify a shape. Edit the settings for a Shape Object, including line style and fill style, using the Shape Object Properties Panel.

Note:

- The Image Object in SWiSH v1.5x is now represented in SWiSH Max by a rectangular-shaped Object filled with the image
- AutoShapes have an additional set of handles for editing. These additional handles modify properties specific to the individual AutoShape (e.g. the number of vertices in a polygon shape). AutoShapes can be converted into Shapes by using Convert to Shapes . Once converted into Shapes, the AutoShape looses its additional editing handles

4.2.2 Button Objects

A Button Object is a Simple Object that is made up of four states: Up, Over, Down and Hit, as shown in the illustration below.



- Scene 1
  - Button
    - Up State
      - Move the mouse over me
    - Over State
      - Now click on the mouse button
    - Down State
      - You did it! You clicked me!
    - Hit State
      - Hit Rectangle

Also see Button Object Properties Panel.

Unless the Has Separate Over State, Has Separate Down State or Has Separate Hit State option is selected in the Button Panel, only the button's Up State will be displayed in the Outline Panel. A button will always have the four states but they will be identical unless modified.

Select the Has Separate Over State, Has Separate Down State or Has Separate Hit State options by right-clicking the button and selecting the appropriate option.

The **Up State** is the default appearance of the button. It is displayed when the Movie starts playing, and whenever the mouse is outside the button.

The **Over State** is displayed when the mouse is moved inside the button. Use it to create 'Rollover' and 'Hover' buttons in a Movie.

The **Down State** is the 'clicked' appearance of the button. It is displayed when the mouse is clicked inside the button.

The **Hit State** defines the active area of the button that reacts to mouse clicks. It is an invisible state and is never displayed

The behavior of the button as the mouse is dragged away from the hit area depends on the Tracking Setting for the button.

The four button states act like Group Objects. Insert any kind of Object, except another button, into the Up, Over and Down states, but only Shape Objects and Text Objects are allowed to define the hit area.

Create an empty Button Object using Insert Button or create a button with the states already set up by using Convert to Button.

Edit the settings for a Button Object using the Button Panel.

4.2.3 Movie Clip Objects

A Movie Clip is a Simple Object that has its own Timeline and is a collection of Objects with Effects. They are the equivalent of 'Movie Clips' in Flash. By default, a Movie Clip is also a Scripting Object.

Note: In SWiSH2 and SWiSH Max version 1 a Movie Clip was referred to as a "Sprite".

Also see Movie Clip Object Properties Panel.

Since a Movie Clip's Timeline is played independently from that of the main Movie, think of it as a piece of reusable animation inside the main Movie.

A Movie Clip Object can contain any type of Object, including other Movie Clip Objects.

Create an animated button by using a Movie Clip Object in the Up, Over and Down button states.

Create an empty Movie Clip using Insert Movie Clip, convert an Object into a Movie Clip using Convert to Movie Clip or Group as Movie Clip, or create a Movie Clips by importing an animated .gif or a .swf Movie.

Change the content of a Movie Clip when the Movie is playing using the Load Movie Action.

Edit the settings for a Movie Clip Object using the Movie Clip Properties Panel.

A Movie Clip is a Scripting Object and can consequently be referenced by other Scripting Objects. However, Movie Clip that have Mouse Events or are inside a Button State cannot act as targets, even if they have a name. Also Movie Clips that are within an unnamed Movie Clip or a Group can only be referred to from within the Movie Clip itself (i.e. from one of the Objects inside the Movie Clip).

4.2.4 Text Objects

A Text Object can be Static, Dynamic or Input. In addition, it can have Scripting Object properties applied if the 'Target' checkbox is checked.

Also see Text Object Properties Panel.

Static Text
Static Text is used to display Static Text in a Movie. A Static Text Object without its 'Target' checkbox checked, is a Complex Object that is made up of a collection of letters. If the Static Text has its 'Target' checkbox checked, then it is considered to be a Scripting Object (this causes the text to lose its Complex Object characteristics). Each letter is called a element of the Text Object.

Dynamic Text
Dynamic Text is an Object that allows the text content to be altered by a script while a Movie is running. Dynamic Text Objects have Simple Object characteristics.

Note: Ticking the 'target' for dynamic text objects and assigning a variable when they are not used within the script can cause a lot of unnecessary overhead for the Flash player. This can cause high CPU usage and will result in the SWF slowing down tremendously.

Input Text
Input Text is an Object that allows the person viewing the Movie to enter text. This is typically used for user-input forms. Input Text Objects have Simple Object characteristics.

Note: For SWF4 export only: If you have an input text field that is a target and has an associated variable, and you want to initialise that variable to an empty string using script, then you must ensure that the text content of the text field is also empty (ie the edit box in the Text Panel where you enter text should be empty). If not, then the content your specified in the Text Panel will appear in the text field when played. For SWF5+, you can initialise the variable to an empty string without any problem.

Text Objects are created by importing a text file or pressing . Import Text.

You can modify the settings for the text using the Text Panel.

Input, Dynamic and Scripting Text Objects are treated as a Simple Object when an Effect is applied.

4.2.5 Group Objects

A Group Object that does not have its 'Target' checkbox ticked, is a Complex Object that is made up of a collection of Objects. The Objects that make up a group are called its elements.

Also see the Group Object Properties Panel.

The components may be Simple or Complex Objects, but cannot have their own Effects. This means that any object with Effects is converted to a Movie Clip before it is grouped.

Unlike a Movie Clip Object, a Group Object does not have its own Timeline.

Group Objects can be created by pressing Group, Convert to Shapes or Convert to Letters on the Grouping Toolbar or the Modify Menu.

Modify the settings for a group using the Group Object Properties Panel.

4.2.6 Sound Object

A Sound is a simple object created when a soundtrack is imported. A Sound Object is only visible in the Outline or Timeline panels, and does not have a visible presence on the stage.

See Importing Sound for a description of importing sound.

Modify the settings for a sound using the Sound Object Properties Panel.

4.2.7 Embedded Video Object

An Embedded Video is a Simple Object that is created when Video is imported into SWiSH Max.

Also see Embedded Video Object Properties Panel.

4.2.8 External Media Object

An External Media component is a Simple Object that is created when external media is imported into SWiSH Max. This creates an external media container which can be used to load and play an external file in the current Scene/Movie Clip/Group.

Note: At the moment only Flash Video (.flv) video can loaded.

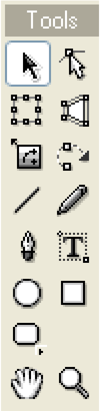
Once an external media container is placed the External Media Panel will be visible.See the section on External Media Properties Panel for more information.

4.3 Tools

Tools are modal commands that determine what SWiSH Max does when you click and drag the mouse on the workspace. You can select a tool by clicking one of the options in the Toolbox, which is located in the top-left of the Layout Panel. You can only select one tool at a time.

After use, the selected tool will remain active until another tool is chosen. The operation can be changed in Tools | Preferences... | Editing; you can opt for the Selection tool to automatically be re-selected and double-clicking a tool will keep it active until another tool is chosen.

The grab handles and cursors that appear around an Object and when the Object is altered will depend on the tool selected.



The available tools are:

- **Select (V)**
- **Subselection (A)**

- **Transform (Q)**
- **Perspective (I)**
- **Fill Transform (F)**
- **Motion Path (M)**
- **Line (L)**
- **Pencil (N)**
- **Pen (P)**
- **Text (T)**
- **Ellipse / Circle (E)**
- **Rectangle / Square (R)**
- **Auto Shapes**
- **Hand (H)**
- **Zoom (Z).**

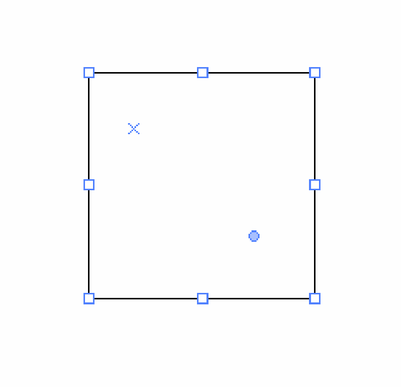
4.3.1 Definitions

This section offers descriptions for the different operations which can be applied with SWiSH Max tools:

- Reference and Transformation Points
- Cursors and Anchors
- Reshape and Transform operation
- Move
- Scale
- Resize
- Rotate / Skew

4.3.1.1 Reference and Transformation Points

Every Object has a Reference point 'x' and a Transformation point 'o'.



'x' is the Reference Point,
'o' is the Transformation Point

The Reference point 'x' is the origin of the object as referenced by effects, script etc. For example the script ...

```
onFrame (1) {
        box._rotation = 45;
```

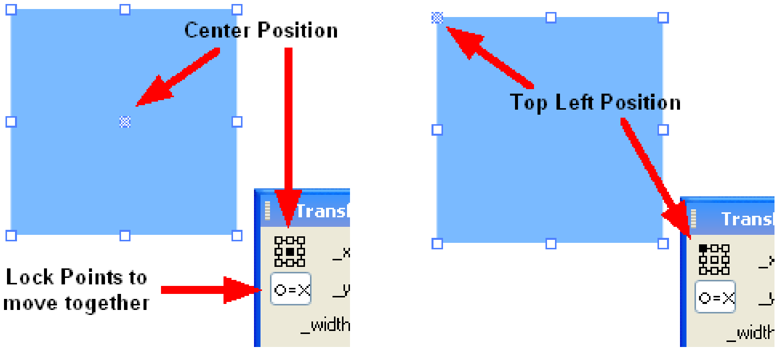
}

...rotates the Object 'box' about the 'x' point.

The Transformation point 'o' is the Object's anchor as referenced by tools, Reshape panel operations, Transform panel operations and any other object operations. Rotation, skew, resize (ALT+drag) and any positional changes applied to at the Reshape or Transform panels refer to the Transformation point.

To move the Transformation or Reference points

The Reference and Transformation points default to the last used positions when a new Object is created. Generally a user will want to keep these points together, for this purpose we have supplied a button on the Transform Panel and Reshape Panel which will lock these points together. A 9-point tool is also available which lets the user select the corners, the center of each edges or the center of the Object.

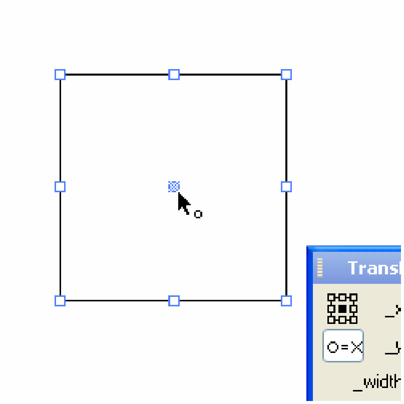


To move ***only*** the Transformation Point to a new location, ensure the 'Transform = Reference' icon button is turned **off** before dragging the Transformation Point to the new location.

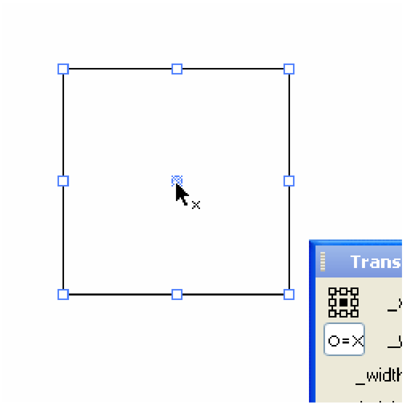
The Reference cannot be selected directly. To move the Reference Point to a new location, turn on the 'Transform = Reference' icon button, move both points to the new location for the Reference Point, turn off the "Transform = Reference" icon button, and then move the Transformation Point to it's location. Alternatively, hold the ***ALT*** key, move both points to the new location for the Reference Point, release the ALT key, and then move the Transformation Point to it's location.

Note:

- when the 'Transform = Reference' icon button is turned on (or ***ALT*** key held), the Transformation point will always reset to the Reference point position
- double clicking the Transformation Point will reset it to the Reference Point position
- Holding down the ***ALT*** key (when the 'Transform = Reference' icon button is turned off) will temporarily allow the two points to be dragged together
- To snap the Transformation Point to Object handles hold the ***Shift*** key when moving near the handles



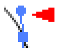
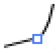
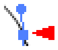
If the Reference and Transformation points are NOT locked the cursor displays the Transformation 'o' symbol.



If the Reference and Transformation points are locked the cursor displays the Reference 'x' symbol.

4.3.1.2 Cursors and Anchors

While working with the drawing tools and objects, various icons and cursors are used to define grab / drag points (anchors) and cursor context. The meaning of those shapes and icons is defined below. Anchors appear on and around the object. Anchors can be moved to alter the size / shape or properties of an Object.

Anchor	Tool	Use
⨯	All	Shows the Reference Point of the Object. The Reference Point can be moved by pressing and holding the Alt key (Alt) and the left mouse button and dragging this handle.
●	All	Shows the Transformation Point of the Object. The Transformation Point can be moved by pressing and holding the left mouse button and dragging this handle.
□	Selection, Perspective, Text	Only with Shape Object or Text Object. When this anchor is dragged, the object is Reshaped (resize, rotation, skew).
■	Selection, Subselection, Transform, Perspective, Fill Transform	All Objects. When this anchor is dragged, the object is Transformed (resize, rotation, skew).
◆	Selection, Subselection, Perspective, Fill Transform	This anchor appears in the text and autoshape objects. For autoshape objects, it changes the shape of the object. For text objects, it changes the margins.
	Subselection	This anchor indicates a control anchor for a curve.
 or 	Subselection	This anchor indicates a vertex anchor.

Cursors appear at the current mouse location. The cursor icon changes to indicate when the object and anchors can be selected by pressing the left mouse button.

Cursor	Tools	Meaning
↖⬚	All	No object / anchor is currently selected.
↖× ↖○	All	The object Reference or Transform Point can be selected
↖⊘	All	Object is not selectable. Shows when: 1. mouse hovering over a symbol at non-key-frame 2. mouser hovering over an empty movie clip which is expanded (Edit in Place). 3. Dragging a unexpanded symbol in the symbol editor
↔ ⤢ ↕ ⤡	All	A scale or resize anchor can be selected.
✥	All	The drawing object can be selected for moving.
↖✥	Selection, Subselection, Text	An autoshape anchor (auto shape object) or a margin anchor (text object) can be selected.
⤵	All	The object has been selected for rotation around the defined Transformation Point.
⇆	All	The object has been selected for Skewing around the defined Transformation Point.
⇹ ⇕	All	Used to drag and re-position the guides.
↖▪ ↖□ ↖⋰ ↖✥	Subselection	The item displayed by the cursor can be selected for reshaping.
✒× ✒+ ✒– ✒○ ✒⁄ ✒^	Pen	Used to create, add, remove, close, extend and reshape line Objects with the Pen Tool.
+⋰ +⁄ +T +○ +□ +☆.... +? '?' is an icon to indicate the chosen tool.	Motion Path, Line, Text, Ellipse, Rectangle, Autoshapes	Used for initial placement and sizing of drawing object.
✎×	Pencil	Used for initial placement and sizing of drawing object.
✋	Pan	Pans the workspace.

4.3.1.3 Reshape vs.Transform

Objects may be reshaped or transformed. Reshaping involves rearranging the object's constituent points, while transforming leaves the points unaltered and instead applies a kind of filter, or transformation, to the whole shape, such as rotation, scale and skew. Reshape and Transform have the same operations: move, scale / resize, rotate and skew.

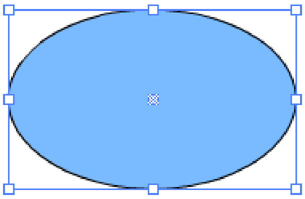
Reshaping:

- only applies to shape objects
- will not alter the line width
- will permanently alter the object
- changes the position and dimensions of the object without altering the scale

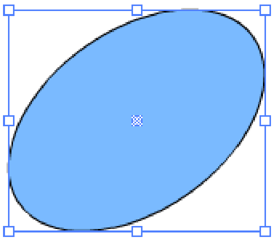
Note:

- If you need to reshape a group of objects, group these objects as a shape before applying the reshape transform (see Modify | Grouping | Group as Shape).
- Objects which have open square handles when selected (with the Select tool) can be reshaped; if an Object has closed square handles it can only be transformed.

The Transform panel will always display the characteristics of the transform. Once an object has been reshaped there is no information relating to the original object. The Selection Tool will transform an Object by moving the shape’s vertices rather than by modifying it’s transformation matrix. For example if an ellipse were rotated using the Select Tool the illustration below shows the handle positions before and after the rotation:



Object before reshaping



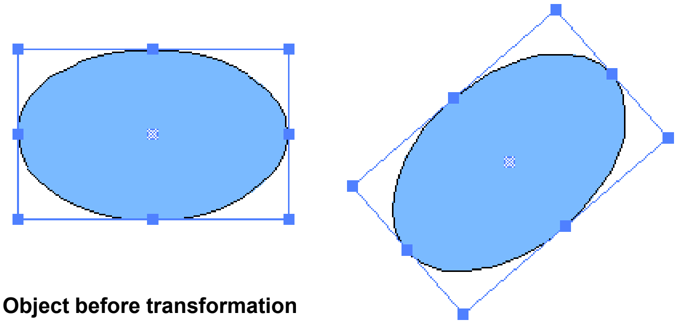
Object after reshaping

The Object’s handles remain un-rotated indicating that the object’s transformation matrix is unchanged. However, if the Object’s handles were rotated to begin with, they would remain rotated. Thus, the angle of the handles always reflects the object’s transformation matrix.

Transforming:

- applies to all objects, groups and Movie Clips
- will alter the line width
- can be removed to restore the original object. This is useful if you want to move a shape through a range of positions across a Scene.

Transforms graphic Objects by clicking and dragging. This selection tool is used for selecting and transforming Objects by modifying the object’s transformation matrix. For example if an ellipse were rotated using the Selection Tool this shows the handle positions before and after the rotate:



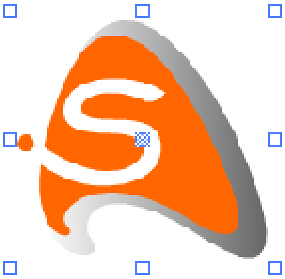
Object before transformation

Object after transformation

By default, simple shapes are Reshaped. You can force a transform using the Transform tool or via the Transform panel.

4.3.1.4 Select

To Select Objects
Click on the Object.



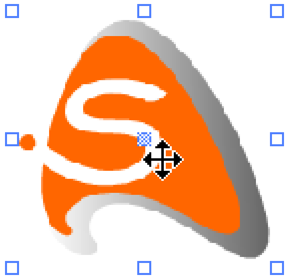
Note:

- To select multiple Objects, press and hold the ***Shift*** key while clicking, or drag the cursor to select surrounded Objects (marquee selection)
- Double-click on the Object to display its properties in the Object Panel

4.3.1.5 Drag

To drag move and drag copy Objects

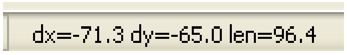
1. Position the mouse cursor over the Object and move over the Object until the cursor changes shape to the one shown below. See Cursors and Anchors for more information on the meaning of cursor shapes
2. Press and hold the left mouse button
3. Drag the Object to the new position and release the mouse



4. Hold down the ALT key and drag to copy an Object. This operation works with any tool which can drag an Object. This includes: Selection Tool, Transform Tool, Perspective Tool.

Note:

- To constrain the line direction to multiples of 45°, press and hold the ***Shift*** key while dragging.
- Pressing the ***Control*** key at any time will override selection to the Selection Tool, consequently it is possible to use the key combination ALT+CTRL-Drag to copy an Object in any mode.



dx=-71.3 dy=-65.0 len=96.4

Status bar shows dx dy and len in pixels

- **dx** - change in x position
- **dy** - change in y position
- **len** - length of the drag

Note: Status bar information only displays with mouse drag and not position changes with arrow keys.

4.3.1.6 Scale

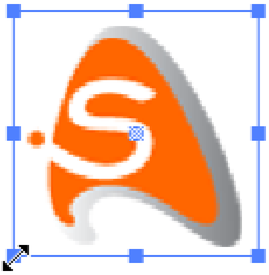
Scale

Dragging the handles provided changes the Scale of the Object. Scaling an Object enlarges or reduces the Object horizontally, vertically or both. If the Object contains lines or text elements, their widths will also be resized proportionally to the object's size.

The Transform Tool option causes the Object size to change via the Scale properties (_xscale, _yscale) of the Object. When applied to text (Static, Input and Dynamic) it will resize the text as well as the surrounding text box.

To Scale Objects

1. Press and hold the left mouse button on a scale handle
2. Drag the handle to the new position and release the mouse



Note:

- To maintain the aspect ratio of the Object, press and hold ***Shift*** key while dragging
- To Scale relative to the opposite vertex, press and hold the ***ALT*** key while dragging
- To Scale relative to the Transformation point, press and hold ***Control*** key while dragging

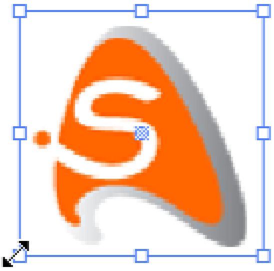
4.3.1.7 Resize

Dragging the handles provided resizes the Object. Resizing an Object enlarges or reduces the object horizontally, vertically or both. If the Object contains lines or text elements their widths will be unchanged.

This Tool option causes the Object size to change via the width and height properties (_width, _height) of the Object. This option cannot be applied to static text. When applied to Dynamic or Input text, it resizes the text box without changing the size of the text.

To Resize Objects

1. Press and hold the left mouse button on a resize handle
2. Drag the handle to the new position and release the mouse



Note:

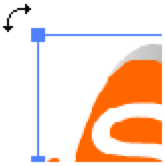
- To maintain the aspect ratio of the Object, press and hold ***Shift*** key while dragging
- To Resize relative to the opposite vertex, press and hold the ***ALT*** key while dragging
- To Resize relative to the Transformation point, press and hold ***Control*** key while dragging

4.3.1.8 Rotate / Skew

To Rotate an Object

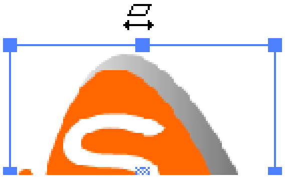
An Object can be rotated around its Transformation Point using the rotate cursor.

The rotate cursor ↶ appears when the mouse pointer is placed just to the outside of the corner handles. When the cursor appears, press the left mouse button and rotate the object to the desired angle.



To Skew an Object

The skew cursor appears when the mouse pointer is placed just to the outside of one of the middle handles.
When the cursor appears, press the left mouse button and skew the object to the desired angle.

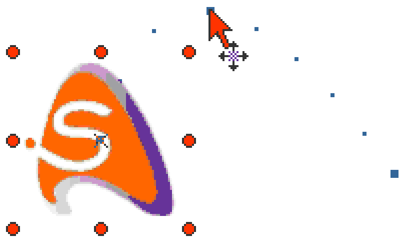


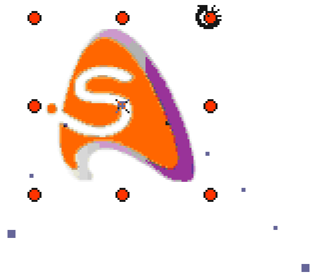
Note:

- To constrain the Rotation/Skew angle to multiples of 45°, press and hold ***Shift*** key while dragging
- To change the point about which the Object rotates, either select and drag the Transformation Point as indicated by a small circle ● in the 'Layout' Panel or change the alignment using the 9 Point icon in the Transform or Reshape Panels.
- To apply the change relative to opposite vertex (for rotation) or edge (for skew) and **not** the Transformation point, press and hold ***ALT*** key whilst dragging.

To Rotate/Skew an Object at a Keyframe

1. Click on the Object
2. Either click on the Timeline ruler or the 'Preview Frame' button to enter Preview Frame mode. The Motion Path of the Object will be displayed in 'Preview Frame' mode
3. Click on a Keyframe handle. Follow the same procedures mentioned above to Rotate/Skew the Object at the Keyframe



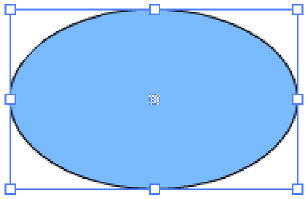


Note:

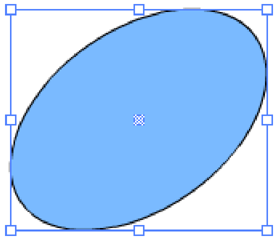
- In 'Preview Frame' mode, double-clicking on the Scene's background in the 'Layout' window will add a Move Effect to the selected object. A Motion Path is created from the object's current position to the position where the double-click occurred
- To apply the change relative to opposite vertex (for rotation) or edge (for skew) and **not** the Transformation point, press and hold ***ALT*** key whilst dragging

4.3.2 Selection Tool

Selection Tool (V)
The selection tool is used for selecting entire Objects and transforming graphic objects by moving or reshaping the object. The Selection Tool will transform an Object by moving the shape's vertices rather than by modifying it's transformation matrix. For example if an ellipse were rotated using the Select Tool the illustration below shows the handle positions before and after the rotation:



Object before reshaping



Object after reshaping

The Object's handles remain un-rotated indicating that the object's transformation matrix is unchanged. However, if the Object's handles were rotated to begin with, they would remain rotated. Thus, the angle of the handles always reflects the object's transformation matrix.

See also Reshape vs. Transformation and Transform Tool.

The Selection Tool can apply the following Reshape operations on Shapes:

- Select
- Drag
- Resize
- Rotate/Skew

Note:

- Pressing the ***Control*** key at any time (when NOT using the Selection tool) resets to the Select tool
- The Transformation Point (o) can always be moved by press-dragging. If it is locked to the Reference Point (x) then both points will be moved together. Pressing the Alt key while dragging will cause the points to become locked during the drag process.
- The transform point can also be moved by choosing a handle from the nine-handles control in the Reshape or Transform panel. Also see how to move the Transformation or Reference points.

4.3.3 Subselection Tool

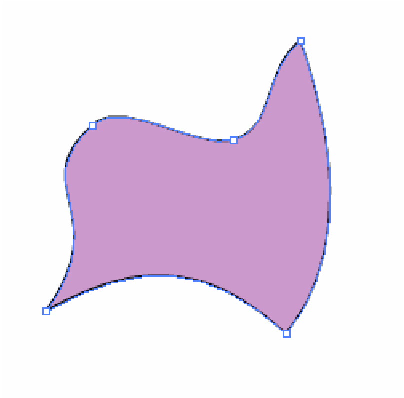
Subselection Tool (A)
Reshapes graphic Objects by clicking and dragging the vertices the object outline path.

When using the Subselection tool the Scale, Resize, Perspective and Rotate/Skew are available.

Note: When used on AutoShapes the Reshape tool activates the AutoShapes editing handles. See AutoShapes for more details

To Select an Object

Click on the Object. The vertices of the selected Object are indicated by white square anchors.

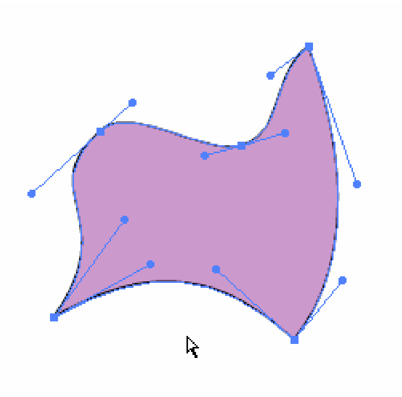


Object selected with Subselection Tool

To select an Anchor

Click on an anchor to edge of an Object to select an anchor or edge. Clicking on an edge selects the two anchors on either end of the edge. Use the ***Shift*** key to toggle the selection of an anchor or edge. Selected vertices are indicated by blue square anchors.

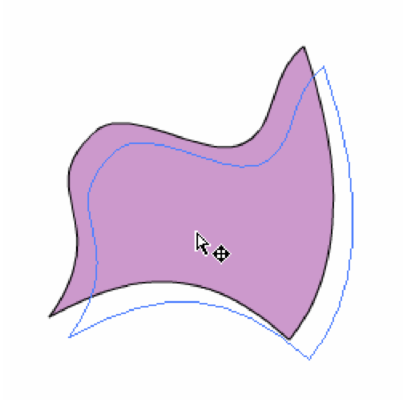
Selecting the anchor points will display the control points - blue circular anchors. Vertex anchors are always on the curve, whereas control-point anchors are usually off the curve. There are either zero, one or two control points associated with each edge, and these affect the curvature of the edge. The control points are shown joined to the adjacent vertices by a solid line.



Object showing selected vertex anchors and control point anchors

To move the whole Object

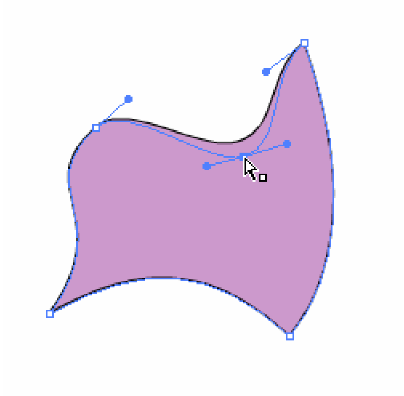
1. Press and hold the left mouse button inside the Object, but not on any vertex or control-point anchor
2. Drag the Object to the new position and release the mouse



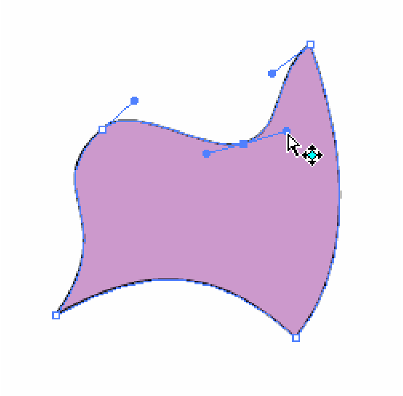
Moving Object

To Reshape Objects by dragging an anchor

1. Press and hold the left mouse button on a vertex or control-point anchor
2. Drag the anchor to the new position and release the mouse



Moving Vertex

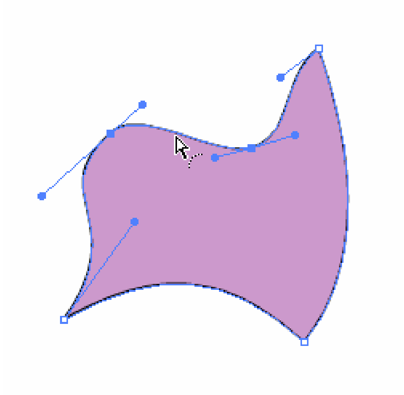


Moving Control Point Anchor

Note: To constrain the direction from the original position to the new position to multiples of 45°, press and hold the Shift key while dragging

To Reshape Objects by dragging an edge

1. Press and hold the left mouse button on an edge
2. Drag the edge to the new position and release the mouse

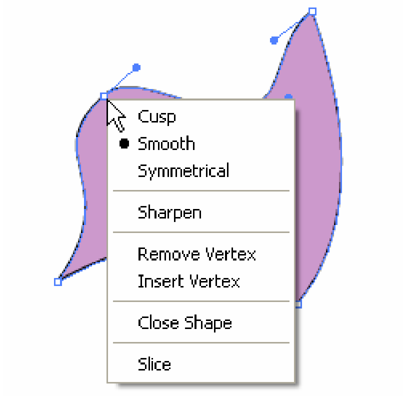


Reshape Edge

Note: To constrain the direction from the original position to the new position to multiples of 45°, press and hold Shift key while dragging

To delete or change the type of a vertex anchor

1. Move mouse pointer to the vertex anchor
2. Right-click and select the command from the pop-up Menu

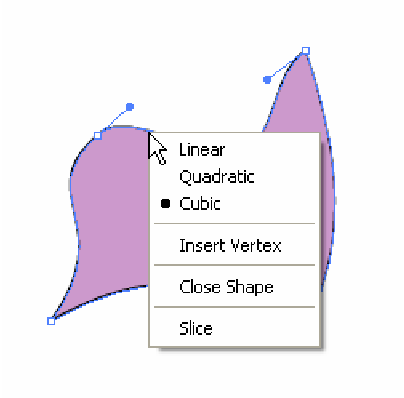


Vertex Anchor Context Menu

- **Cusp**: lets you adjust the adjacent control points independently. When the vertex is a cusp, the edges are still curved, but the corner is not smooth
- **Smooth**: the adjacent control points are always in the same direction, but can be different distances from the vertex. The corner at the vertex is smooth
- **Symmetrical**: the adjacent control points are always in the same direction and are both the same distance from the vertex
- **Sharpen**: the edges on either side of the vertex are made straight, thereby removing the associated control points. This makes the corner sharp
- **Remove Vertex**: removes the vertex anchor completely
- **Insert Vertex**: adds a new vertex anchor at the mouse position
- **Close Shape**: closes the shape
- **Slice**: Slices the shape (see below)

To change the type of an edge, or insert a vertex

1. Move mouse pointer to the edge
2.Right click and select the command from the pop-up Menu



Edge Context Menu

- **Linear**: the edge is straight and does not have any control points
- **Quadratic**: the edge is curved and has one control point

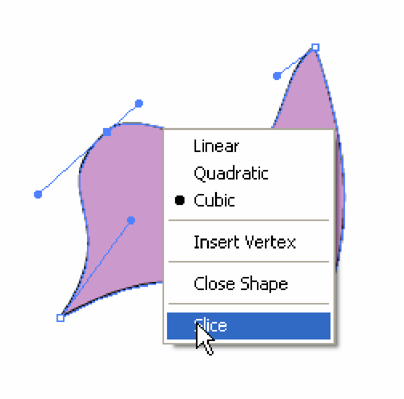
- **Cubic**: the edge is curved and has two control points
- **Insert Vertex**: adds a new vertex anchor at the mouse position
- **Close Shape**: closes the shape
- **Slice**: Slices the shape (see below)

To Reshape an Object in Preview Frame mode

In 'Preview Frame' mode, if you reshape an object at a Keyframe, the change is *not only* applied to the current Frame. The shape of the Object in all other Frames will be changed as well.

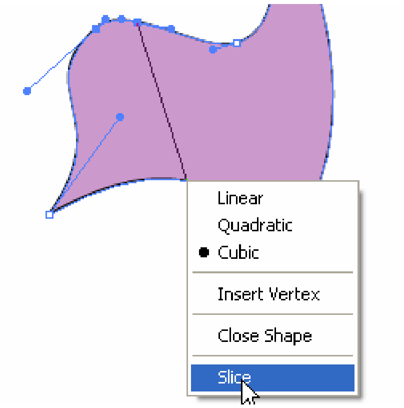
To Slice a Shape Object

1. Move mouse pointer to the edge of the Object
2. Right click and select the Slice command from the pop-up Menu



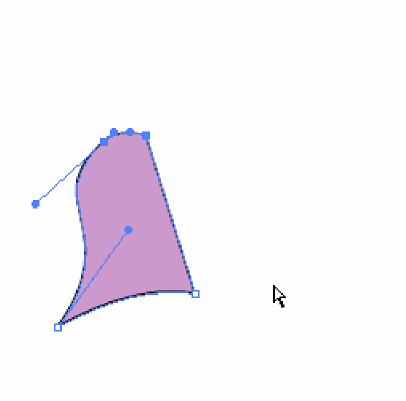
Slice Edge - Start

3. Move the mouse pointer to another point at the edge of the Object
4. Right click and select the Slice command from the pop-up Menu



Slice Edge - End

Your Shape should now be sliced as shown below.

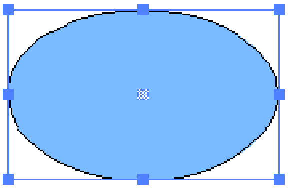


Slice Edge - Result

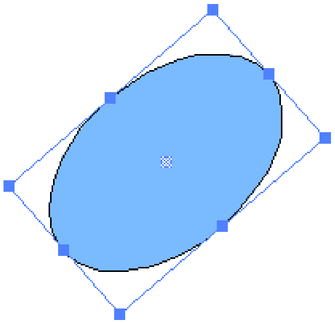
Note: Cannot be used for AutoShapes

4.3.4 Transform Tool

Transform Tool (Q)
Transforms graphic Objects by clicking and dragging. This selection tool is used for selecting and transforming Objects by modifying the object's transformation matrix. For example if an ellipse were rotated using the Selection Tool this shows the handle positions before and after the rotate:



Object before transformation



Object after transformation

The Object's handles are rotated indicating that the object's transformation matrix has changed. If the Object's handles were rotated to begin with, they would be rotated to a new position.

See also Reshape vs. Transformation and Reshape Tool.

The Transform Tool can apply the following Transform operations on all Objects:

- Select
- Drag
- Scale
- Rotate/Skew

Note:

- The Transform Tool cannot transform sub-path selections. It always transforms the entire object
- The Transform tool can be used to select and deselect objects like the Selection Tool
- Corner and side handles for the Transform Tool are drawn as solid squares indicating the object will be transformed (rather than reshaped)

- The Transformation Point (o) can always be moved by press-dragging. If it is locked to the Reference Point (x) then both points will be moved together. Pressing the Alt key while dragging will cause the points to become locked during the drag process.
- The transform point can also be moved by choosing a handle from the nine-handles control in the Reshape or Transform panel. Also see how to move the Transformation or Reference points.
- Text objects do not display margin handles when the Transform tool is selected

4.3.5 Perspective Tool

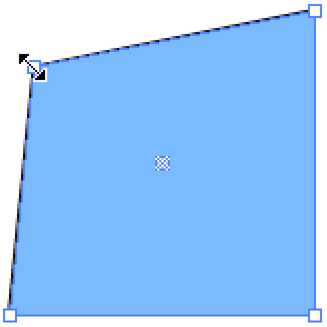
Perspective Tool (I)

Dragging the handles provided modifies the Perspective of the Object. Modifying the Perspective of an Object moves the Object's corner towards the point the perspective handle is moved to.

Note: This option ***ONLY*** applies to Shape Objects, if any other Object type is selected it will behave like the Selection Tool.

To Modify an Object's Perspective

1. Press and hold the left mouse button on a perspective handle
2. Drag the handle to the new position and release the mouse



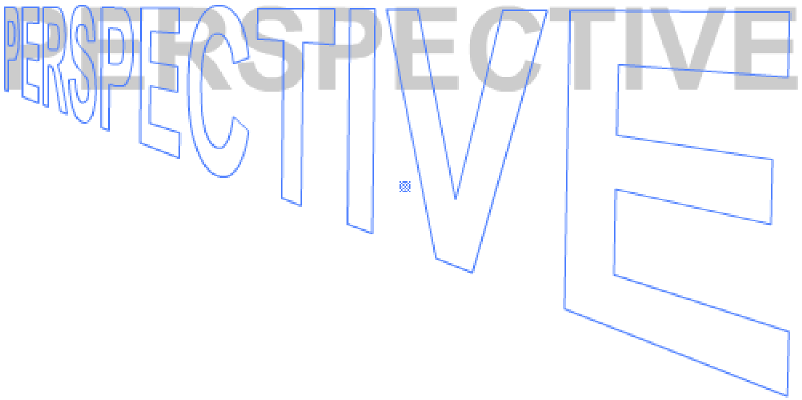
The Perspective Tool can apply the following Reshape operations on Shape Objects:

- Select
- Drag

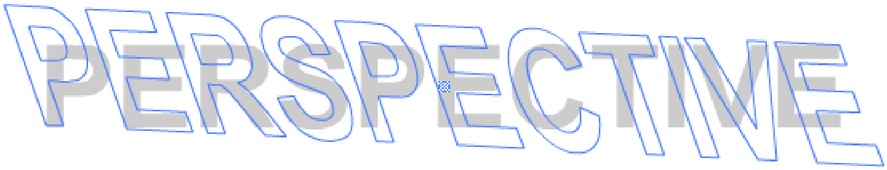
There are a few key combinations which will modify the operation of the perspective tool:

- **Shift-drag**: Moves the dragged handle along the existing edge leaving all other handles unchanged.
- **Alt-drag**: Also moves the opposite handle (to the dragged handle) in the same but opposite movement. The shape is always a parallelogram.
- **Shift+At-drag**: Similar action to Alt except it moves the adjacent edges. The shape is always a trapezium.

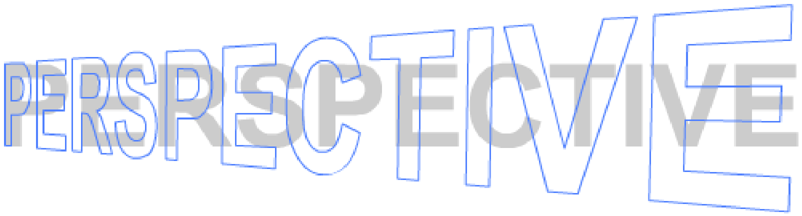
As an example consider the following Text Object "PERSPECTIVE" which has been grouped as a shape Object (Modify | Grouping | Group as Shape) and dragged using the hotkey combinations mentioned.



Perspective Tool using Shift-Drag of bottom right hand handle



Perspective Tool using Alt-Drag of bottom right hand handle



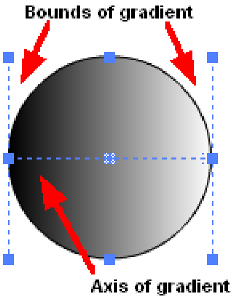
Perspective Tool using Shift+Alt-Drag of bottom right hand handle

4.3.6 Fill Transform Tool

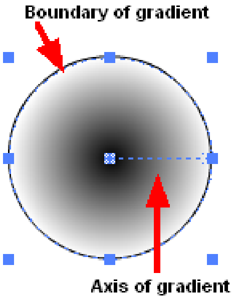
Fill Transform (F)
Transforms an Object's gradient or image fill without transforming the Object. The handles shown depend upon the tool option selected.

Note: This option ***CANNOT*** be applied to solid fill Objects. In this case the Fill Transform Tool will behave like the Selection Tool.

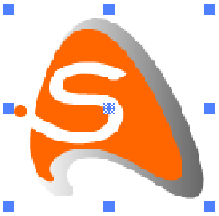
The following diagrams illustrate the handles used for the Fill transform.



Fill transform on linear gradient

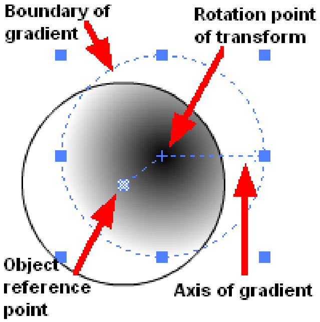


Fill transform on radial gradient

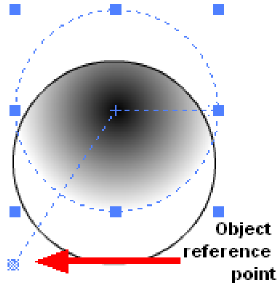


Fill transform on image

The following diagrams illustrate the handles shown when the transform is move off centre.



Linear gradient with transform centre dragged off centre (shifted reference point).

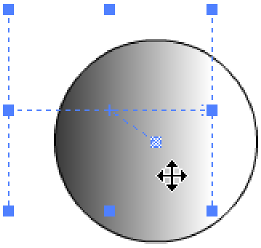


Linear gradient with reference at bottom left

Note: To see the extent of the image when it is dragged off the object check the option to 'View | Show Bounding Boxes'. It is also worthwhile adding a stroke to the image box, or object to see the bounds of the object.

To move the filled gradient or image

1. Select the Fill Transform tool
2. Select the Object you wish to transform
3. Follow the same procedures mentioned in Select tool to move the filled gradient or image

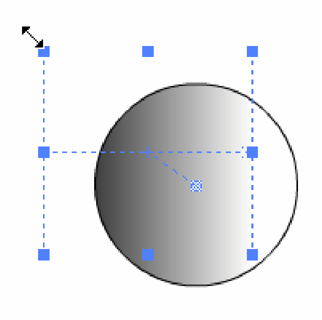


Move Transform

To Scale the filled gradient or image

1. Ensure the Fill Transform tool is selected
2. Select the Object, and move the cursor over the handles until you can select the Scale operation (i.e. ↔ ⤢ ↕ ⤡).

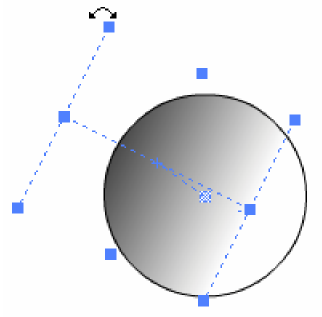
3. Click and drag to scale the filled gradient or image



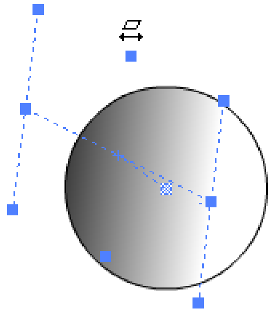
Scale Transform

To Rotate/Skew the filled gradient or image

1. Ensure the Fill Transform tool is selected
2. Select the object, and move the cursor over the handles until you can select the Rotate or Skew operation (i.e. ↷ or ⇔).
3. Click and drag to rotate (or Skew) the filled gradient or image



Rotate Transform



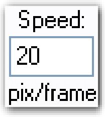
Skew Transform

4.3.7 Motion Path Tool

Motion Path (M)

Plots a Motion Path for the selected Object by clicking and dragging.

Speed is defined as the rate of movement. Increasing the pixels per second will decrease the number of Frames used while decreasing it's value will increase the number of Frames used.



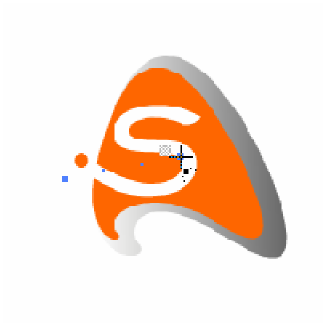
Speed:
20
pix/frame

Note: The Speed Tool option is only available when the Motion Path Tool is selected

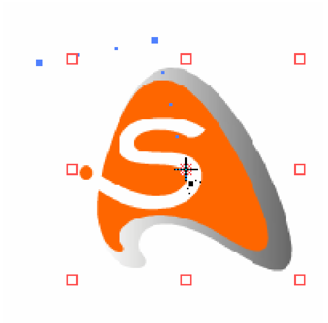
To draw a Motion Path for an Object

1. Select the Object

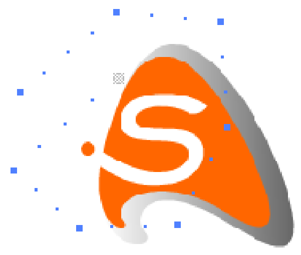
2. Click at the new position where the Object will be displayed for the next Keyframe. In this step, a new Move Effect will be added into the Object's Timeline automatically



Note: Hold the ***Control*** key whilst using the Motion Path Tool to show the Object handles



3. Repeat step 2 to add more Move Effects



Note:

- To constrain the direction from the current position to the new position to multiples of 45°, press and hold the ***Shift*** key while clicking
- To draw a sharp corner path for the next Keyframe, press and hold the ***Alt*** key while clicking
- Frame Preview mode is automatically selected whenever you click on the 'Layout' window
- Motion Path now creates Move Effects with absolute Transforms, so changing a Move Effect in the middle of a Motion Path won't affect subsequent Effects

4.3.8 Line Tool

Line Tool (L)
Draws a line.

To draw a Line
1. Position the mouse pointer to where you want the line to begin
2. Press and hold the left mouse button
3. Drag the mouse pointer to where you want to end the line, and release the mouse

Note:

- To constrain the line direction to multiples of 45°, press and hold the ***Shift*** key while dragging.
- To move the whole Line drawing before it is completed, place the first point, press and hold the ***Spacebar*** down while dragging.

4.3.9 Pencil Tool

Pencil Tool (N)

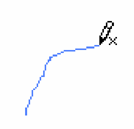
Draws a freehand line object.

To draw a freehand line

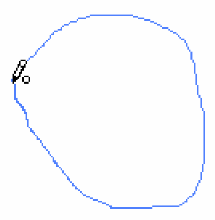
1. Position the mouse pointer to where you want the line to begin
2. Press and hold the left mouse button
3. Drag the mouse pointer on the path you want the line to take
4. Release the mouse button to end the drawing

Note:

- To move the whole Pencil drawing before it is completed, place the first point, press and hold the ***Spacebar*** down while dragging
- To draw a closed shape, release the mouse button at the starting point when the 'o' cursor appears, or hold the ***ALT*** key and release.



Pencil Start Drawing



Pencil Close Shape

4.3.10 Pen Tool

Pen Tool (P)

The Pen tool draws a set of connected curves or line segments.

The Pen Tool will remain selected until another tool is selected. The Pen Tool cursor has the same graphic as the toolbar icon with supplementary symbols offset from the bottom right which these include: "x', +', '-', 'o', '^', '/'.

Overview

To draw a set of connected curves

1. Position the mouse pointer to where you want the curve to begin
2. Press and hold the left mouse button to place the on-curve anchor
3. Drag the mouse pointer in the tangent direction of the curve to be drawn
4. Release the mouse button to place the off-curve anchors

5. Repeat steps 1-4 to draw additional curves
6. Press 'Enter' key to end the drawing

Note:

- To constrain the tangent direction to multiples of 45°, press and hold the Shift key while dragging

To draw a set of connected line segments

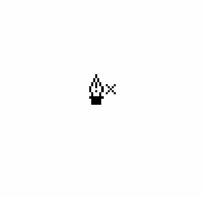
1. Press and release the left mouse button at the position you want the line segment to begin
2. Continue pressing and releasing the left mouse button at the position you want to end the current line segment and begin the next line segment
3. Press 'Enter' key to end the drawing

Note:

- To constrain the direction of the line segment to multiples of 15°, press and hold the Shift key while moving the mouse pointer
- To draw a polygon, double-click the left mouse button at the starting point on completion of the drawing

Description of Pen Tool Functions

When first entering this tool, pen cursor will show an 'x' to indicate a new shape will be started. When the first point is plotted, any previously selected object(s) will become unselected (ie, their handles will disappear). The point will be drawn and the pen cursor will lose the 'x'.

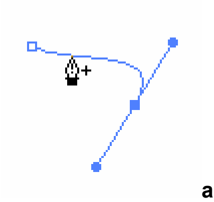


create

Click-Release to draw a straight point/line segment.
Click-Drag to draw a curved point/line segment with tangent vectors.

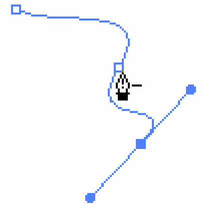
Note: Shift+click will constrain the horizontal or vertical line from the previous point.

Hover-Outline, cursor will add a '+'. If clicked, a new control point will be added. Its type based on preceding point type (straight or curved).



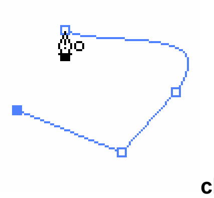
add point

Hover-Point, cursor will add a '-'. If clicked, the point will be deleted.



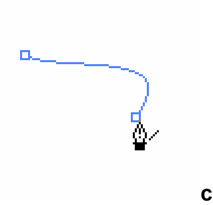
delete point

Hover-StartPoint, cursor will add a 'o' to indicate shape closure. If clicked, shape will be drawn with outline and control points displayed. A shape should be able to remain open, so that when a stroke is placed around it, it does not go all the way around. Pressing Enter, ESC or selecting something else finishes the shape but does not close it.



close shape

Shape creation can continue if clicking on one of the endpoints (cursor shows a '/').

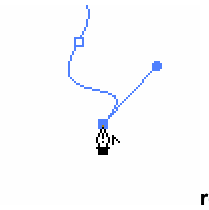


continue existing shape

DEL or Backspace (BS) key will delete the most recently plotted anchor point (while creating new shape).

Convert Anchor Point Sub-tool (^)

This tool is entered by holding down the **Alt** key. The cursor, when over an anchor point, changes to have a '^'. The tool is canceled when the Alt key is released.



reshape point (Alt+Pen)

Note:
Alt+Click-Drag on an anchor point will make it a curved point and edit the curve while dragging the point
Alt+Click-Release on a control point (ie, anchor point) will make it a straight point
Alt+Click+Spacebar-Drag on a anchor point will move the anchor point

4.3.11 Text Tool

Text Tool (T)
Draws a Text Object.

To draw a Text Object

1. Select the Text tool
2. Position the mouse cursor where you wish to place the text
3. Press the left mouse button to create a new Text object and type to create the text. The new Text Object type (and attributes) will be the same as the previous Text Object. A new Static Text Object will automatically resize itself to contain the text that is typed

4. Select the text and alter the font size, color, Object type, etc. in the Text Panel
5. If necessary format the Text Object using Margins and Indents

Note: It is recommended 'Show Outline' and 'Show Bounding Boxes' are enabled to make empty text boxes visible.

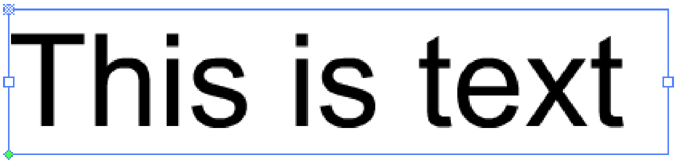
In Static Text Objects individual characters, words or paragraphs can be formatted for font, size, color and attribute. This is not possible if 'Render text as HTML' or 'Use Device Fonts' is set.

The following mouse select operations work on Text Objects:

- Single click - place the cursor
- Double click - select a word
- Triple click - select a paragraph
- Control+A - select all text in the Object
- Shift-Drag - selects a range of text

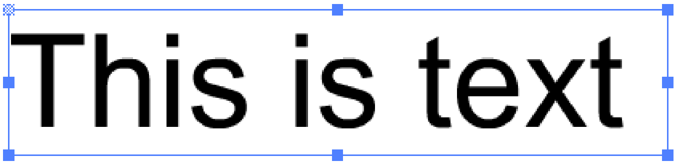
Text Objects can be Reshaped or Transformed. Reshaping Text Objects only resizes or skews the text box (no rotation) and has no effect on the text itself. Transforming Text Objects scales, rotates or skews the Object.

Resize: Click the Selection Tool (or hold down the Ctrl key) and drag one of the side or corner handles to resize the text box. Resizing has no effect on the size of the text.
Scale: Click the Transform Tool and drag one of the side or corner handles to scale the text box. Scaling will change the size of the text.
Rotate/Skew: Click the Transform Tool and drag one of the side or corner handles to rotate or skew the text and the text box.



This is text

Text Object showing Reshape handles. Reshape only changes the Text box size not the Text.



This is text

Text Object showing Reshape handles. Transform changes the Text, and Text box's, size and orientation.

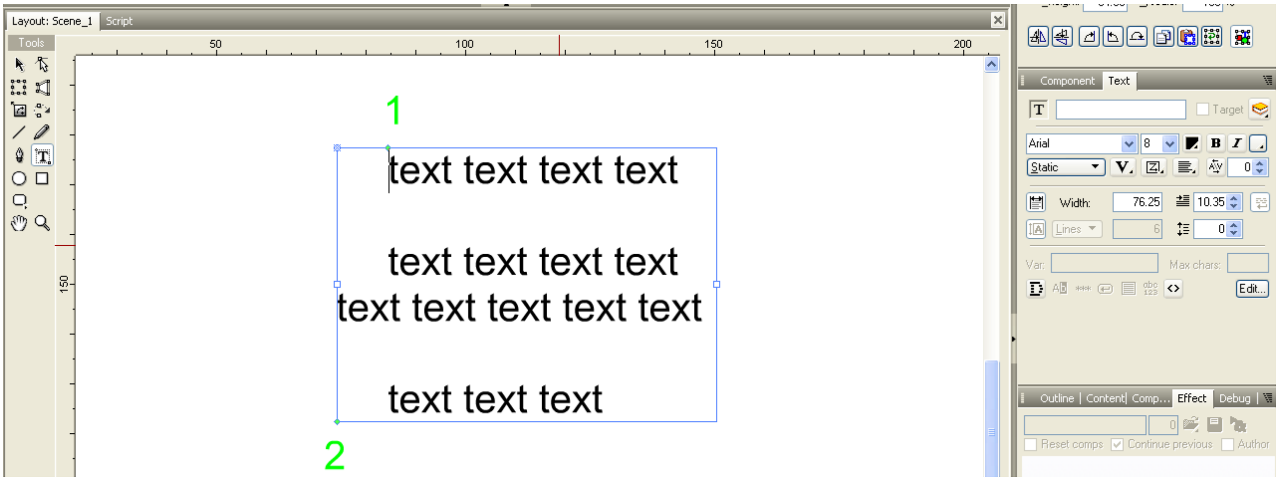
Note:

- If 'Enable Margins and Indent' is ***NOT*** enabled for Static Text Objects, this Object ***cannot*** be Reshaped
- A Static Text box can be rotated if you the Transformation point is set to the center of the object and skew is used on the end handles

4.3.11.1 Margin and Indent Handles

These handles apply to Text Objects that have margins. The 'Enable Margins and Indent' button in the Text Panel is used to enable margins.
When the text box is selected, additional drag handles are shown as green dots. The diagram below shows

the handles. The numbers 1 and 2 are used to identify the handles and do not appear in SWiSH Max.



Editing Text Object margins

1. Press and hold the left mouse button on one of the green text margin handles
2. Drag the selected text margin handle to the desired position
3. Release the mouse button

For Static Text Objects:
Handle 1 is used to adjust the left margin of the first line of a paragraph.
Handle 2 is used to adjust the left margin of the remainder of the paragraph.

For Dynamic and Input Text Objects:
Handle 1 is used to adjust the left margin of the first line of a paragraph.
To obtain word wrapping within the text box, it is necessary to select 'Wrap lines at Word-Breaks' in the Text Object Panel.

Note: Margin handles are intended for left justified text. They have limited use when applied to right or center justified text

4.3.12 Ellipse / Circle Tool

Ellipse Tool (E)
Draws an ellipse or circle.

To draw an Ellipse

1. Press and hold the left mouse button where you wish to position the ellipse
2. Drag the mouse pointer diagonally to adjusting the size of the ellipse
3. Release the mouse button to end the drawing

Note:

- To draw a circle, press and hold the ***Shift*** key while drawing with the Ellipse tool
- To move the whole shape before it is completed, place the first point, press and hold the ***Spacebar*** down while dragging.

4.3.13 Rectangle & Square Tool

Rectangle Tool (R)
Draws a rectangle or square.

To draw a Rectangle

1. Press and hold the left mouse button down (or click and release) at the position you wish to draw a corner of the rectangle
2. Drag the pointer diagonally to adjust the size of the rectangle
3. Release the mouse button (or click and release) to end the drawing

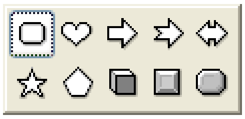
Note:

- To draw a square, press and hold the ***Shift*** key while dragging
- To move the shape before it is completed, place the first point, press and hold the ***Spacebar*** down while dragging.

4.3.14 AutoShapes

AutoShapes
Allows you to draw a number of predetermined shapes. AutoShapes have an additional set of handles for editing. The additional handles allow you to modify properties specific to the individual AutoShape (e.g. the number of vertices in a polygon shape).

To access the different shapes, press and hold the left mouse button over the AutoShape tool to display the Menu below.



Note:

- Once an AutoShape is selected, the AutoShape tool will retain that symbol until a different shape is selected
- To move the whole shape before it is completed, place the first point, press and hold the ***Spacebar*** down while dragging.

The following AutoShapes are available:

- Round rectangle (O)
- Heart (U)
- Arrow (W)
- Notched arrow (D)
- Two-way arrow (2)
- Star (S)
- Polygon (Y)
- 3D Cube (C)
- Beveled button (K)

- Rounded button (B).

Selecting an AutoShape

1. Click and hold the AutoShape tool to open the submenu of shapes
2. Select the desired AutoShape

To draw an AutoShape

1. Press and hold the left mouse button where you wish to position the AutoShape
2. Drag the mouse pointer diagonally to adjusting the size of the AutoShape
3. Release the mouse button to end the drawing

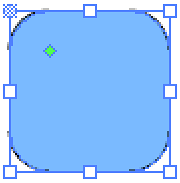
Note: To constrain the AutoShape to have equal width and height, press and hold the Shift key while dragging

To Reshape AutoShapes

Drag the moveable handles, or right/left click on the fixed handles to change the feature of an AutoShape.

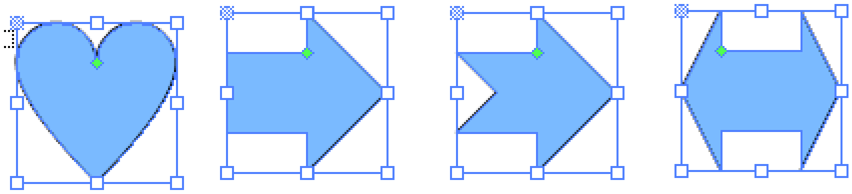
Rounded rectangle

- Press and drag to adjust the size of rounded corners



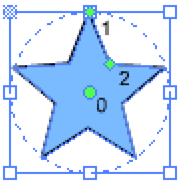
Heart/Arrow/Notched arrow/Two-way arrow

- Press and drag to adjust the shape



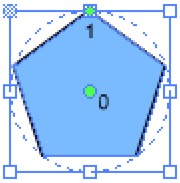
Star

- Left/right click handle 0 to increase/decrease the number of points
- Press and drag handle 1 on the convex vertex to adjust the convex vertices' orientation
- Press and drag handle 2 on the concave vertex to adjust the concave vertices' position



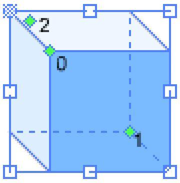
Regular polygon

- Left/right click handle 0 to increase/decrease the number of sides
- Press and drag handle 1 to adjust the orientation



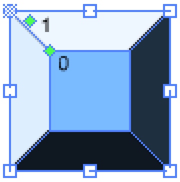
Cube

- Press and drag handles 0 and 1 to adjust the perspective 'look' of the cube
- Press and drag handle 2 to adjust the lighting direction and contrast of the shade



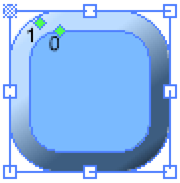
Beveled Button

- Press and drag handle 0 to adjust the widths of beveled edges
- Press and drag handle 1 to adjust the lighting direction and contrast of the shade



Rounded Button

- Press and drag handle 0 to adjust the size of rounded corners and the size of the bevel.
- Press and drag the handle 1 to adjust the lighting direction and contrast of the shade.



Converting to Shapes

AutoShapes can be converted into simple shapes by using the Break into Shapes function. Once converted into shapes the AutoShape looses its additional editing handles.

4.3.15 Hand Tool

Hand Tool (H)

Pans around the workspace in the Layout Panel by clicking and dragging.

To pan the 'Layout' Panel

1. Press and hold the left mouse button on the workspace
2. Drag the mouse to pan the 'Layout' window

3. Release the mouse button to finish panning

4.3.16 Zoom Tool

Zoom Tool (Z)
Zooms in/out of the workspace in the Layout Panel using the mouse.

To Zoom in/out
Click the left mouse button at the position you want to zoom in on.

Note:
- To zoom out, press and hold the ***Ctrl+'+'*** ("+" on the numberpad)
- To zoom in, press and hold the ***Ctrl+'-'*** ("-" on the numberpad)

To Zoom to an area
1. Press and hold the left mouse button at one corner of the area
2. Drag the mouse pointer diagonally to position the other corner of the area
3. Release the mouse button to zoom in

Note:
- To switch to the area zoom out area, press and hold the ***Ctrl+Spacebar***
- To switch to the area zoom in area, press and hold the ***Ctrl+Alt+Spacebar***

See also Zoom In, Zoom Out, View at 100%, Fit to Scene in Window, Fit Objects in Window, Adjust Zoom on Layout Change.

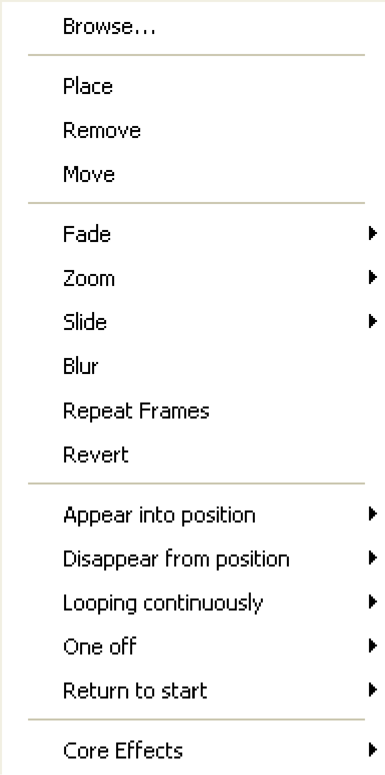
4.4 Effects

Effects are animations that change the appearance of an Object over time. You can add, modify and coordinate Effects using the Timeline Panel.

Note: Before proceeding it is advisable to review the section on 'Effect Settings Panel' .

Simple Effects vs Complex Effects
All Effects can be categorized as either Simple Effects or Complex Effects. A Simple Effect animates the entire Object in unison. A Complex Effect animates the elements that make up the Object independently.

Effects Menu



The Effects Menu (Insert | Effects) is divided up into the following four Effects groups:

- Place Effects - simple Effects used to add, remove and move Objects in the Movie (Place, Remove, Move)
- Basic Effects - commonly used Basic Effects (Fade, Zoom, Slide, Blur, Repeat Frames, Revert)
- Authored Effects - Effects pre-installed with SWiSH Max and any Effects created and saved in the Effects Library
- Core Effects - create new Effects based on one of the Core Effect types that can be used or saved in the Effect Library (Transform, Squeeze, Alternate, Snake, Explode, 3D Spin, 3D Wave, Vortex, Wave, Typewriter).

Effects Settings
The properties of an Effect are edited using the Effect Settings Panel, and many Effects also have Common Settings, such as Motion, Easing, Transform and Camera.

Effects Authoring
SWiSH Max allows the user to to author and distribute their own Effects. Read the Effects Authoring guide to find out more.

Note: If multiple Objects are selected when an Effect is added, the Effect will be applied to each of the selected Objects

4.4.1 Place Effects

Place Effects are simple Effects used to add, remove and move objects in your Movie.

The following Place Effects are provided:

- Place

- Remove
- Move
- Play

4.4.1.1 Place

The Place Effect displays an Object. This Effect is one Frame long and is indicated by a 🟩 Frame in the Timeline.

You can use the Place Effect to move Objects that are already displayed by altering the Object's properties in the Place Effect's setting (in the Effect Settings panel). To do this, double-click the Place Effect in the Timeline to display the 'Effect' Panel. The Effect Settings panel enables you to set the position, scale, rotation, alpha and color of the Object as it is revealed.

Refer to the Start At tab Effect Setting for more information.

4.4.1.2 Remove

The Remove Effect hides a visible Object. This Effect is one Frame long and is indicated by a ☒ Frame in the Timeline.

4.4.1.3 Move

The Move Effect changes an Object's position, scale, rotation, alpha, color or any combination of these variables over time.

The settings for the Move Effect can be displayed by double-clicking the Move Effect in the Timeline (to open the Effect Settings Panel). The Effect Settings Panel allows you to set the position, scale, rotation, alpha and color of the Object you want at the end of the Move Effect. SWiSH Max will automatically generate the Frames in between.

See the Motion tab Effect Setting for more information.

4.4.1.4 Play

The Play Effect ***only*** applies to Sound Objects.

The settings for the Play Effect can be displayed by double-clicking the Play Effect in the Timeline (to open the Effect Settings Panel). The Effect Settings Panel allows you to set the audio volume at the start and end of the Effect. SWiSH Max will automatically generate the Frames in between.

See the Audio tab Effect Setting for more information.

4.4.2 Basic Effects

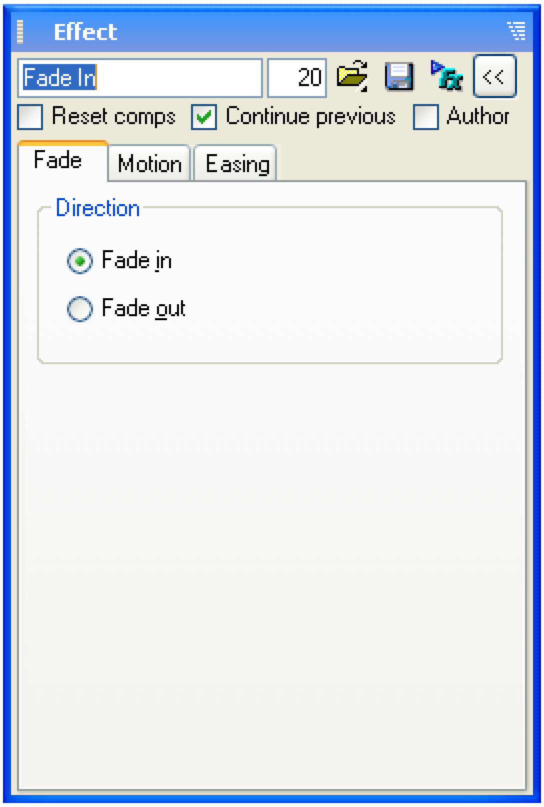
Basic Effects comprise commonly used Effects. The following Basic Effects are provided:

- Fade In
- Fade Out
- Zoom In
- Zoom Out
- Slide In
- Slide Out
- Blur
- Repeat Frames
- Revert.

4.4.2.1 Fade In

The Fade In Effect fades in by increasing the opacity (or alpha) of the Object from completely transparent (0% alpha) until it is completely opaque (100% alpha).

The 'Fade Effect' dialog box is represented below. It can be displayed by double-clicking the Fade Effect in the Timeline to display the Effect Setting Panel.



Effect

Fade In | 20

□ Reset comps ☑ Continue previous □ Author

Direction

● Fade in

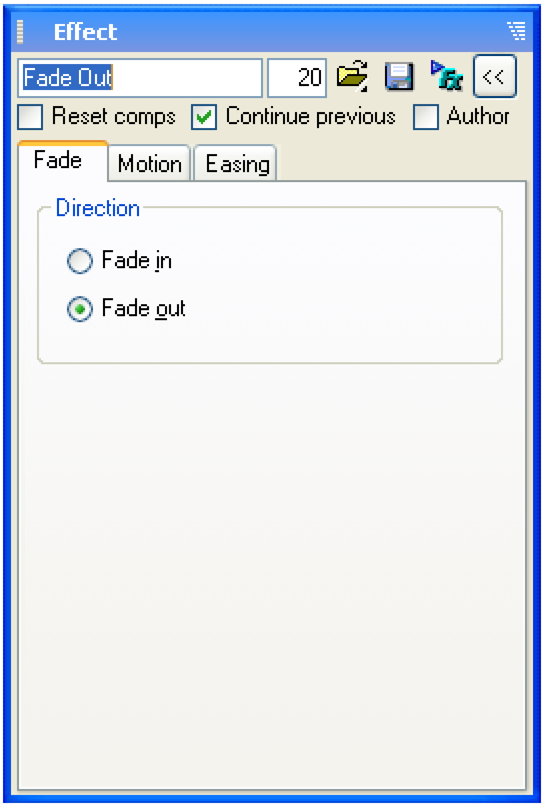
○ Fade out

Direction
Allows you to change the Fade In Effect to a Fade Out Effect.

4.4.2.2 Fade Out

The Fade Out Effect fades out by decreasing the opacity (or alpha) of the object from completely opaque (100% alpha) until it is completely transparent (0% alpha).

The 'Fade Effect' dialog box is represented below. It can be displayed by double-clicking the Fade Effect in the Timeline to display the Effect Setting Panel.



Fade Out 20

□ Reset comps ☑ Continue previous □ Author

Direction

○ Fade in

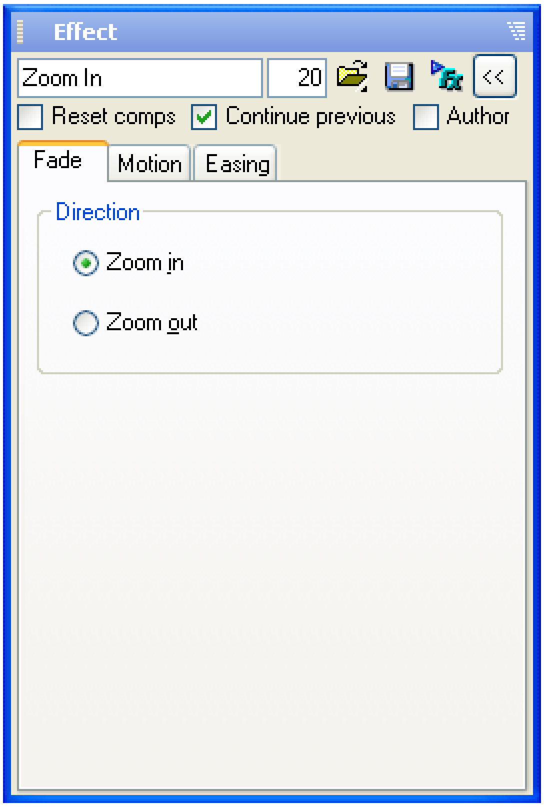
● Fade out

Direction
Allows you to change the Fade Out Effect to a Fade In Effect.

4.4.2.3 Zoom In

The Zoom In Effect zooms in by increasing the scale of an Object.

The 'Zoom Effect' dialog box is represented below. It can be displayed by double-clicking the Zoom Effect in the Timeline.to display the Effect Setting Panel.



Zoom In 20

□ Reset comps ☑ Continue previous □ Author

Direction

● Zoom in

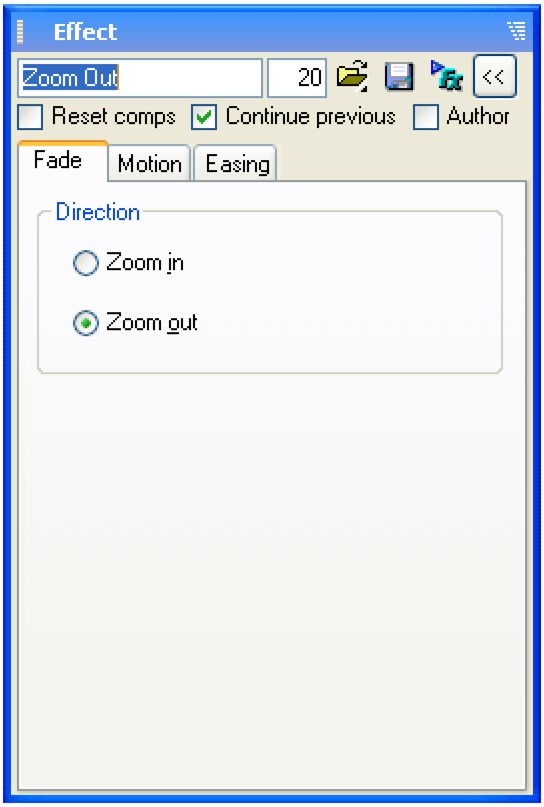
○ Zoom out

Direction
Allows you to change the Zoom In Effect to a Zoom Out Effect.

4.4.2.4 Zoom Out

The Zoom Out Effect zooms out by decreasing the scale of an Object.

The 'Zoom Effect' dialog box is represented below. It can be displayed by double-clicking the Zoom Effect in the Timeline to display the Effect Setting Panel.



Zoom Out 20

□ Reset comps ☑ Continue previous □ Author

Direction

○ Zoom in

● Zoom out

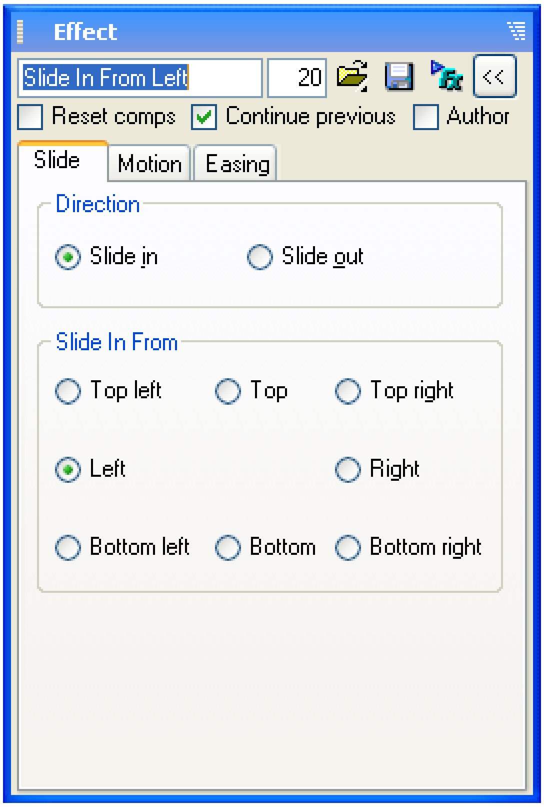
Direction
Allows you to change the Zoom Out Effect to a Zoom In Effect.

4.4.2.5 Slide In

The Slide In Effect moves an Object from outside the Movie (any corner or side) to it's original position.

The 'Slide Effect' dialog box is represented below. It can be displayed by double-clicking the Slide Effect in the Timeline to display the Effect Setting Panel.

In the illustration below, the object will slide in from the left.



Effect

Slide In From Left | 20

□ Reset comps ☑ Continue previous □ Author

Direction

● Slide in ○ Slide out

Slide In From

○ Top left ○ Top ○ Top right

● Left ○ Right

○ Bottom left ○ Bottom ○ Bottom right

Direction
Allows you to change the Slide In Effect to a Slide Out Effect.

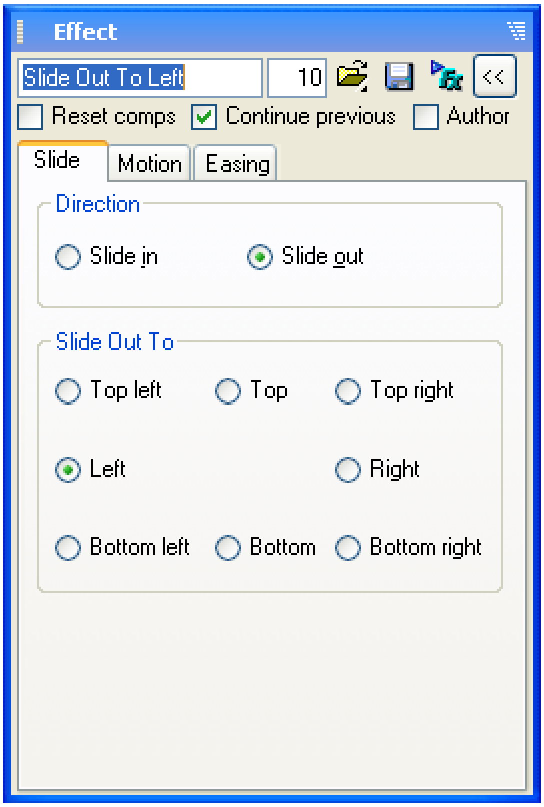
Slide In From
Specifies which way the Effect will Slide In from.

4.4.2.6 Slide Out

The Slide Out Effect moves an Object from it's original position to outside the Movie (any corner or side).

The 'Slide Effect' dialog box is represented below. It can be displayed by double-clicking the Slide Effect in the Timeline to display the Effect Setting Panel.

In the example below, the object will slide out to the left.



Slide Out To Left 10

□ Reset comps ☑ Continue previous □ Author

Direction

○ Slide in ● Slide out

Slide Out To

○ Top left ○ Top ○ Top right

● Left ○ Right

○ Bottom left ○ Bottom ○ Bottom right

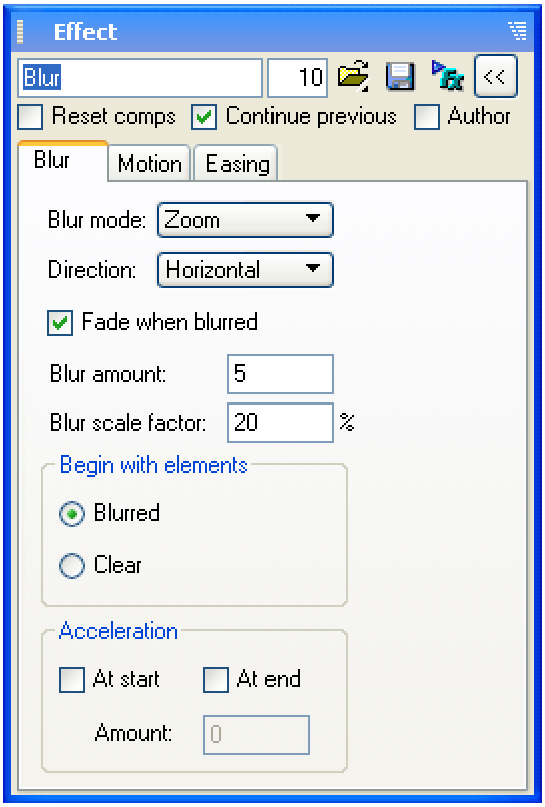
Direction
Allows you to change the Slide Out Effect to a Slide In Effect.

Slide Out To
Specifies which way the effect will Slide Out to.

4.4.2.7 Blur

The Blur Effect creates many copies of an Object, with each copy having either a progressively increasing or decreasing alpha value. By changing the position, scale or alpha value of each copy over time, it can create a motion-blur Effect.

Below is an illustration of the Blur tab in the Blur Effect. It can also displayed by double-clicking on a Blur Effect in the Timeline to display the Effect Setting Panel.



Effect

Blur 10

□ Reset comps ☑ Continue previous □ Author

Blur | Motion | Easing

Blur mode: Zoom

Direction: Horizontal

☑ Fade when blurred

Blur amount: 5

Blur scale factor: 20 %

Begin with elements

● Blurred

○ Clear

Acceleration

□ At start □ At end

Amount: 0

Blur Mode
Determines the style of the Blur Effect. The styles available are:

- **Zoom Blur:** duplicates of the Object are displayed at progressively larger scales, with smaller alpha values. When the Object is blurred in, the duplicates are faded in and squeezed into the size of the original Object. When the Object is blurred out, the duplicates are faded out and stretched outward
- **Mirror Blur:** when the Object is blurred, two sets of duplicates move away from the object's original position in opposite directions. When the Object is brought into focus, two sets of duplicates move towards the original position from opposite directions. The further away the duplicates are, the smaller the alpha value
- **Slide Blur:** duplicates of the Object move from the off-screen position defined by the 'Direction' option (and the Effect Transform) to the Object's original position. Selection of either the 'Blurred' or 'Clear' radio button determines whether the Effect travels to or from the Object's original position on-screen to the off-screen point. Each duplicate is displayed with progressively smaller alpha values. As the Object approaches its original position, the duplicates catch-up with the Object so they all finish at the same place.

Blur Amount
Determines how many copies of the Object are duplicated to carry out the Blur Effect.

Direction
Controls the direction of squeeze/stretch, moving in/away or slide in/out.

Blur Scale Factor
Specifies the maximum scaling change or offset between two duplicates.

Begin with elements
Controls the initial state of an Object and whether the alpha values are increasing or decreasing in the Effect (i.e. fading in or fading out). If the Blurred option is selected, the Object emerges blurred and focuses to a

sharp, solid object by the end of the Effect. If the Clear option is selected, the Object is sharp and solid at the beginning, blurring and fading to invisible by the end of the Effect.

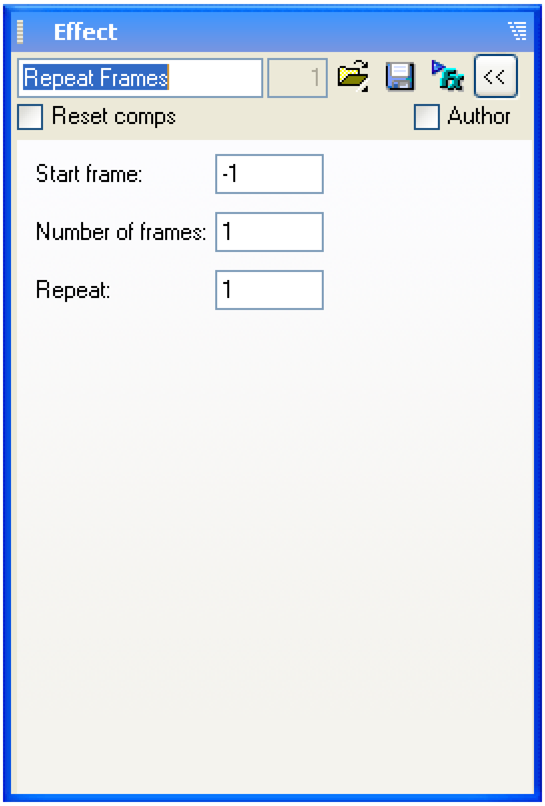
Acceleration
Controls the blur speed at the start and at the end of the Effect.

- **At Start**: accelerates the blur at the start of the Effect
- **At End**: decelerates the blur at end of the Effect
- **Amount**: controls the blur speed at the start and at the end of the Effect. A value of zero means no acceleration. A positive value accelerates at the start or decelerates at the end. A negative value decelerates at the start or accelerates at the end

4.4.2.8 Repeat Frames

The Repeat Frames Effect replays the stipulated Frames the specified number of times. Nominate a starting Frame, duration and a number of repeats and the Effect will repeat those Frames. It will apply its own Transforms.

The Repeat Frames tab of the 'Repeat Frames' Effect Panel is represented below. It can be displayed by double-clicking on a Repeat Frames Effect in the Timeline to display the Effect Setting Panel.



Repeat Frames 1

□ Reset comps □ Author

Start frame: -1

Number of frames: 1

Repeat: 1

Start Frame
Specifies the starting Frame. It can be either an absolute Frame number, or a relative number (if negative).

Number of Frames
Specifies the number of Frames to be repeated from the starting Frame.

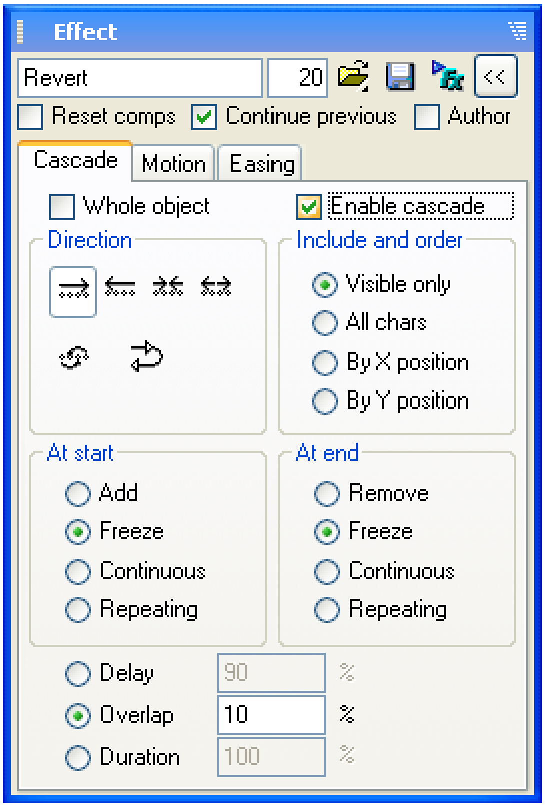
Repeat
Specifies how many times to repeat.

The Repeat Frame Effect will not permit the repetition of only part of an Effect. Settings are always adjusted such that Effects are either entirely included or excluded from the range of Frames.

4.4.2.9 Revert

The Revert Effect lets you play from the end of an effect to the original position smoothly. The Revert Effect must follow the Effect you want to revert.

The Revert Effect tab of the 'Repeat Frames' Effect Panel is represented below. It can be displayed by double-clicking on a Repeat Frames Effect in the Timeline to display the Effect Setting Panel.



Effect

Revert 20

□ Reset comps ☑ Continue previous □ Author

□ Whole object ☑ Enable cascade

Direction

Include and order: ● Visible only ○ All chars ○ By X position ○ By Y position

At start: ○ Add ● Freeze ○ Continuous ○ Repeating

At end: ○ Remove ● Freeze ○ Continuous ○ Repeating

○ Delay 90 %

● Overlap 10 %

○ Duration 100 %

The 'Cascade' tab enables you to control the order elements of a Complex Object are animated. Cascading animates elements one after another instead of all together. Further a detailed overview of cascade settings please see Cascade tab settings.

4.4.3 Authored Effects

Effects other than Place and Basic Effects are referred to as Authored Effects. These Effects are stored on the hard disk as a ".sfx" file.

SWiSH Max comes with many pre-defined Authored Effects. As these Effects are stored on disk in the Effect Library, existing Effects can be customized and resaved or new Effects can be created by altering the settings of existing Effects and then saving the ".sfx" file under a new name. Effects Authoring provides the ability to present the user with a subset of parameters that are available for alteration.

See Creating Effects and Effects Authoring for more information on creating custom and Authored Effects.

The Authored Effects provided are divided into the following categories:

- Appear into position - these Effects animate text or Objects to bring them into the current position
- Disappear from position - these Effects animate text or Objects away from where they are currently placed. They will have disappeared by the end of the Effect
- Loop continuously - these Effects animate text or Objects in a repeating pattern. You can loop the Frames containing these Effects or string multiples of them together to make a longer animation
- One off - these Effects animate text or Objects, but don't have any particular start or end point
- Return to start - these are similar to the Loop continuously Effects, but will always start and finish with the current placement.

4.4.4 Core Effects

Create new Effects, based on one of the Core Effect types, that you can use or save in your Effect Library.

The Core Effects you can create new Effects from are:

- Transform (a Basic Effect)
- Squeeze (a Basic Effect)
- Alternate (a Basic Effect)
- Snake (a Basic Effect)
- Explode (a 3D Effect)
- 3D Spin (a 3D Effect)
- 3D Wave (a 3D Effect)
- Vortex (a 3D Effect)
- Wave (a Special Effect)
- Typewriter (a Special Effect)

The category of Effect determines which additional configuration tabs are associated with it.

Basic Effects
The additional configuration tabs associated with Basic Effects are Transform, Cascade, Motion, Easing, StartAt.

3D Effects
The additional configuration tabs associated with 3D Effects are Camera, Cascade, Motion, Easing, StartAt.

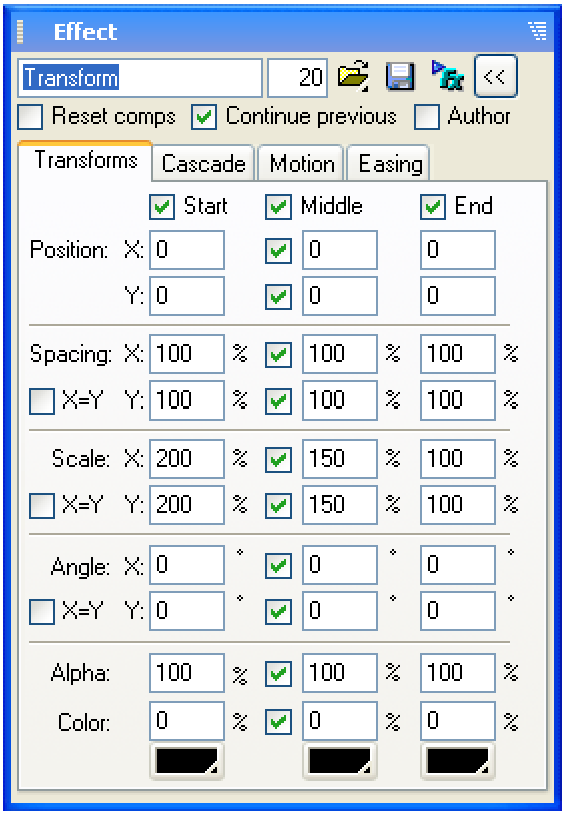
Special Effects
The additional configuration tabs associated with Special Effects are Motion, Easing, StartAt.

4.4.4.1 Transform

The Transform Effect applies a sequence of transforming animations to each element of a Complex Object.

The Transform Effect can be applied to a Complex Object to introduce each element with the same animation. By using the settings in the Cascade tab, the element animations can follow one after another or overlap. The Transform Effect can be applied forwards or backwards, and can be used to add or remove elements.

The 'Transform' setting tab of the 'Transform' Effect Panel is represented below. It can be displayed by double-clicking on a Transform Effect in the Timeline to display the Effect Setting Panel.



Effect

Transform 20

☐ Reset comps ☑ Continue previous ☐ Author

Transforms | Cascade | Motion | Easing

		☑ Start	☑ Middle	☑ End
Position:	X:	0	☑ 0	0
	Y:	0	☑ 0	0
Spacing:	X:	100 %	☑ 100 %	100 %
☐ X=Y	Y:	100 %	☑ 100 %	100 %
Scale:	X:	200 %	☑ 150 %	100 %
☐ X=Y	Y:	200 %	☑ 150 %	100 %
Angle:	X:	0 °	☑ 0 °	0 °
☐ X=Y	Y:	0 °	☑ 0 °	0 °
Alpha:		100 %	☑ 100 %	100 %
Color:		0 %	☑ 0 %	0 %
		■	■	■

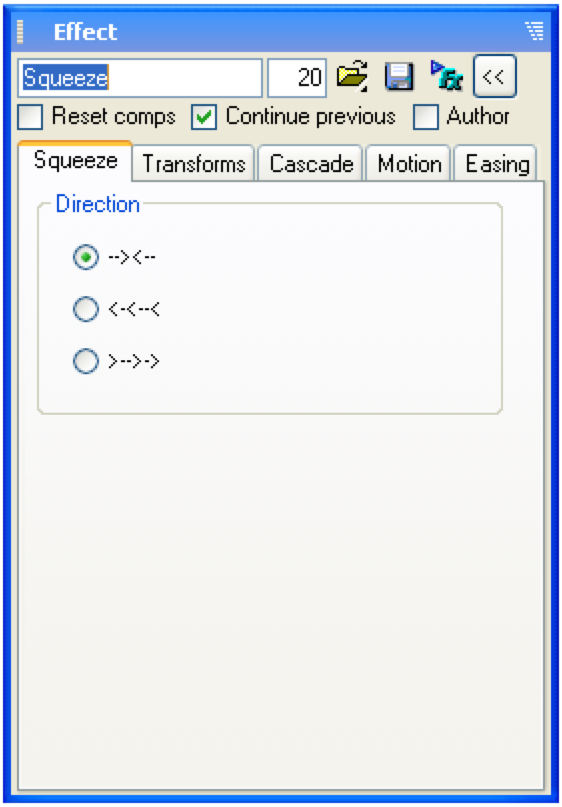
See the Transforms tab and Cascade tab for more information.

4.4.4.2 Squeeze

The Squeeze Effect squeezes or stretches the elements of a Complex Object over time.

The Squeeze Effect can be applied to a Text Object to change the kerning over time. Words can be made to squeeze inward or stretch outward.

The 'Squeeze' setting tab of the 'Squeeze' Effect Panel is represented below. It can be displayed by double-clicking on a Squeeze Effect in the Timeline to display the Effect Setting Panel.



Squeeze 20

□ Reset comps ☑ Continue previous □ Author

Direction

● --><--

○ <-<--<

○ >-->->

Direction

Controls the direction of squeezing or stretching:

- --><--: Squeezes/stretches to/from center
- <-<--<: Squeezes/stretches to/from left
- >-->->: Squeezes/stretches to/from right.

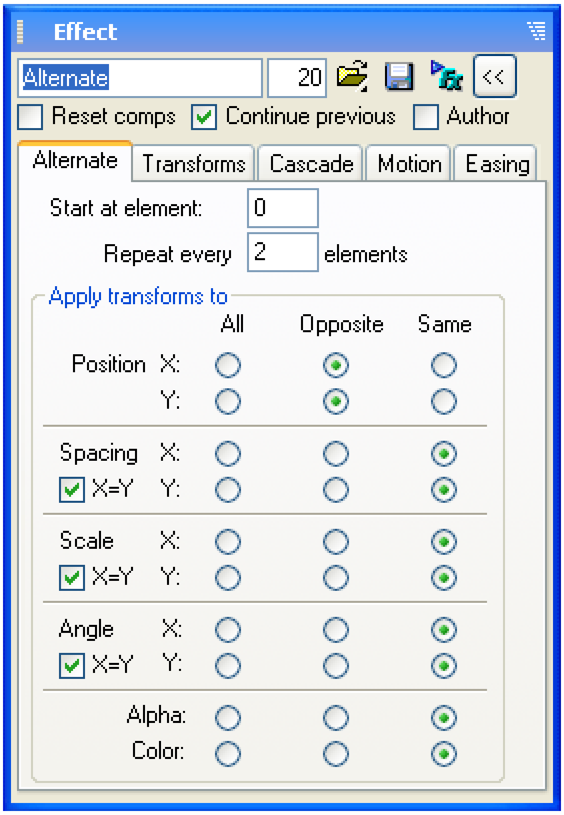
Note:

- The element (X) and line (Y) spacing factors at Start/Middle/End of the Effect are specified in the Transforms tab. Using the Y spacing factor, you can squeeze/stretch a multiple-line Complex Object vertically
- Setting the Direction to center makes the Squeeze Effect equivalent to the Transform Effect

4.4.4.3 Alternate

The Alternate Effect applies alternating proportions and directions of animation to elements of a Complex Object.

The 'Alternate' setting tab of the 'Alternate' Effect Panel is represented below. It can be displayed by double-clicking on a Alternate Effect in the Timeline to display the Effect Setting Panel.



Effect

Alternate 20 <<

□ Reset comps ☑ Continue previous □ Author

Alternate | Transforms | Cascade | Motion | Easing

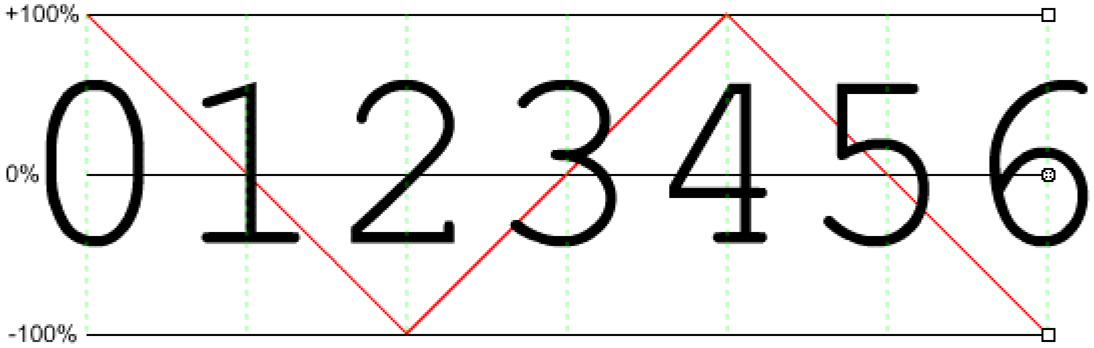
Start at element: 0

Repeat every 2 elements

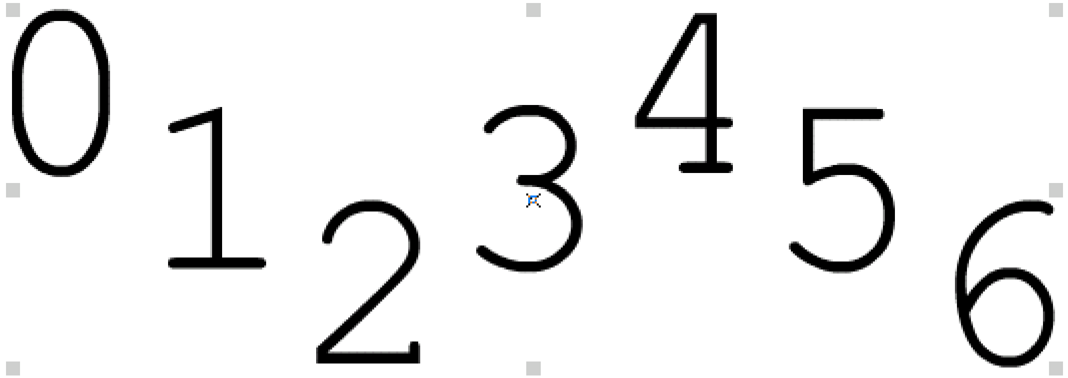
Apply transforms to

		All	Opposite	Same
Position	X:	○	●	○
	Y:	○	●	○
Spacing	X:	○	○	●
☑ X=Y	Y:	○	○	●
Scale	X:	○	○	●
☑ X=Y	Y:	○	○	●
Angle	X:	○	○	●
☑ X=Y	Y:	○	○	●
	Alpha:	○	○	●
	Color:	○	○	●

To see how this Effect works, imagine a zig-zag line that goes between +100% and -100%, for example, with a cycle that repeats every four elements starting at element position 0. You would get a figure like that shown below.



The red line represents the amount and direction of Transform applied for each element position. You can see that element position 0 gets a full 100% of the Transform, element position 1 gets zero, element position 2 gets -100%, and so on. If the Transform for the Alternate Effect is one that adds 50 pixels to the Y value, then the first Frame of the Effect would look like this:



You can see that the first element was moved up by the full 50 pixels, the second element was unchanged, the third moved down by 50 pixels, and so on.

Start at element
The maximum proportion in the positive direction occurs at this element position in the sequence. The default is the 0 position (first element), so the first element always get the full amount of the Transform.

Repeat every 'n' elements
The Alternate Effect goes through a complete cycle every 'n' elements. The default is a four-element cycle. In that case, every fourth element will have the same amount of Alternate Effect as shown in the example above, that is:

- maximum positive direction (100%)
- zero (no transform)
- maximum negative direction (-100%)
- zero (no transform).

Setting the repeat every 'n' elements figure to two, for example, would specify a repeating cycle like this:

- maximum (100%) positive direction
- maximum (100%) negative direction.

Apply Transforms to
The Transform which is alternated is set in the in the Transforms tab. For each Transform setting (position, spacing, scale, angle, alpha and color), there are three options:

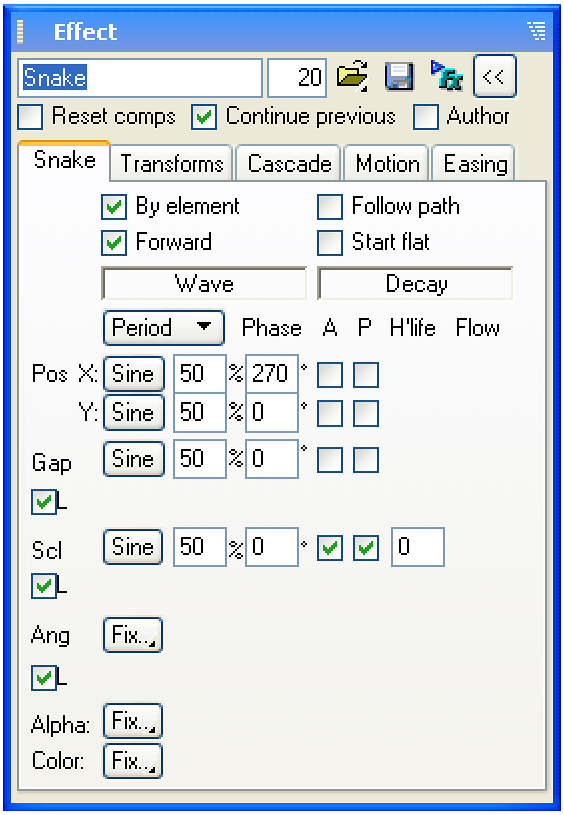
- **All**: applies the full amount of the Transform to all elements. There is no alternation for this setting
- **Opposite**: applies the proportion and direction of the Transform depending on the element position
- **Same**: applies the proportion of the Transform depending on the element position, but always in the positive direction.

X=Y checkboxes
This option forces the Y value to equal the X value.

4.4.4.4 Snake

The Snake Effect applies wave forms to vary the Transforms for elements of a Complex Object over time.

The 'Snake' setting tab of the 'Snake' Effect Panel is represented below. It can be displayed by double-clicking on a Snake Effect in the Timeline to display the Effect Setting Panel.



Effect

Snake 20

□ Reset comps ☑ Continue previous □ Author

Snake | Transforms | Cascade | Motion | Easing

☑ By element □ Follow path

☑ Forward □ Start flat

Wave | Decay

Period | Phase | A | P | H'life | Flow

Pos X: Sine 50 % 270 ° □ □

Y: Sine 50 % 0 ° □ □

Gap Sine 50 % 0 ° □ □

☑L

Scl Sine 50 % 0 ° ☑ ☑ 0

☑L

Ang Fix...

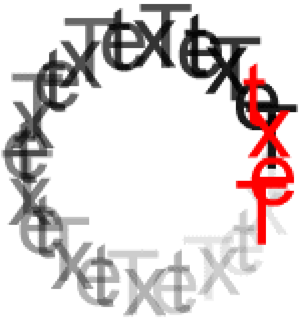
☑L

Alpha: Fix...

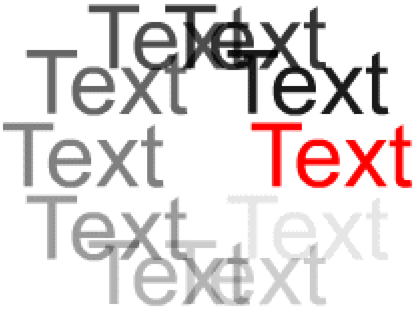
Color: Fix...

By element
When the 'By element' option box is checked, the Snake Effect applies to each element in turn, based on its position within the Object. When the option is turned off, the Snake applies to all the elements at the same time.

For example, the following images show a time-lapse view of some text moving in a circle. The first image shows the Effect with the By element option turned on.



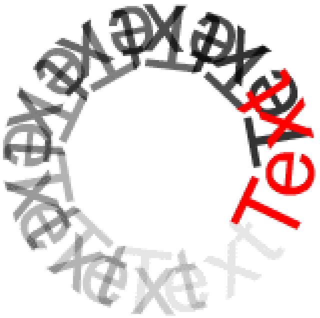
The second image shows the Effect with the By element option turned off.



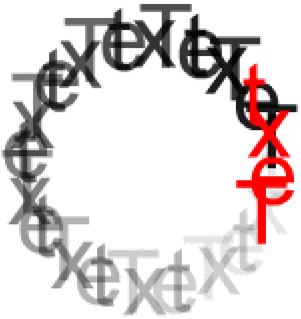
Follow path

When the Follow path option is turned on, the elements automatically rotate to follow the path. When it is turned off, the elements maintain their original orientation. Any angle Transform is added on to this calculated angle.

For example, the following images show a time-lapse view of some text moving in a circle. The first image shows the Effect with the Follow path option turned on.



The second image shows the effect with the Follow path option turned off.

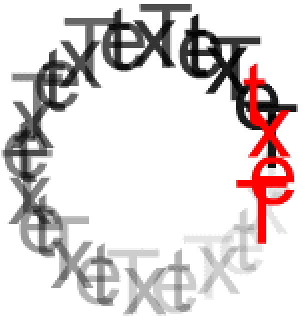


Forward

The Forward option moves the elements in order. When the option is turned off, the elements move in reverse order, as shown below.

Note: This Effect only works when the By element option is checked.

For example, the following images show a time-lapse view of some text moving in a circle. The first image shows the Effect with the Forward option checked.

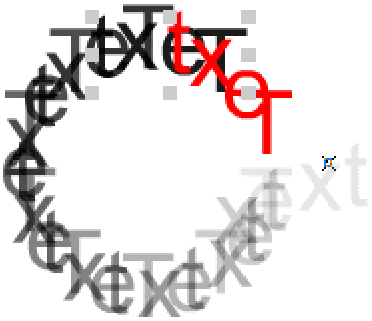


The second image shows the Effect with the Forward option turned off.

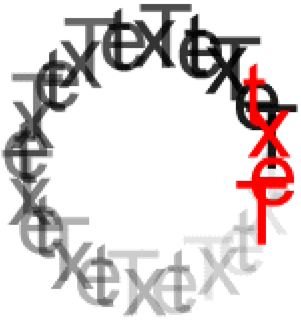
Start flat
When the Start flat option is turned on, elements start from their initial positions and gradually move into the Snake Effect. When unchecked, the Snake Effect starts immediately.

Note: This only has an effect when By element is ticked.

For example, the following images show a 'time-lapse' view of some text moving in a circle. The first image shows the Effect with the Start straight option turned on.

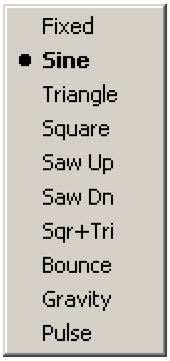


The second image shows the Effect with the Start straight option turned off.



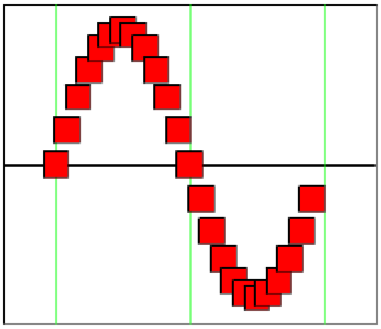
Wave
This set of columns has the Wave settings for each Transform. The Wave settings are the waveform itself, the Period is number of Frames over which the wave repeats its 360 degree cycle, and the Phase is where the wave starts within its 360 degree cycle.

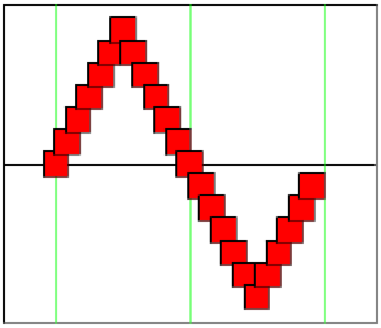
Waveform
There are nine separate waveforms, plus a Fixed value indicating that no waveform applies.

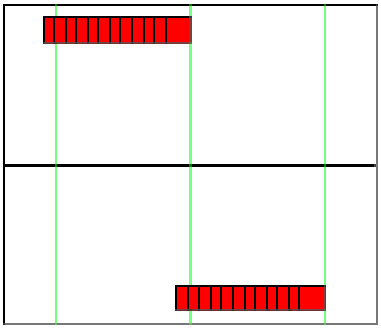


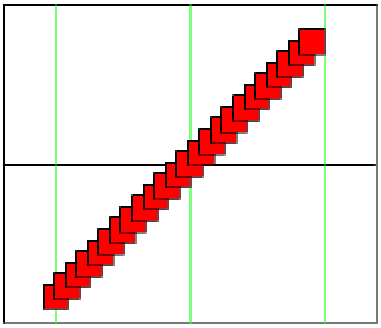
Fixed
● Sine
Triangle
Square
Saw Up
Saw Dn
Sqr+Tri
Bounce
Gravity
Pulse

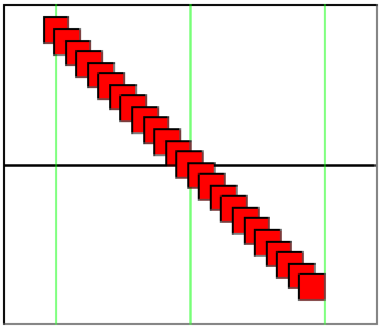
The following illustrations show the effect of each waveform on the y value of a rectangle.

Sine:  moves smoothly between the start and end values

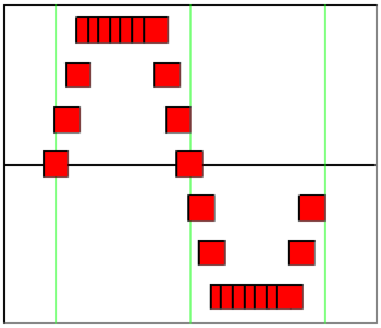
Triangle:  moves linearly between the start and end values

Square:  jumps between the start and end values

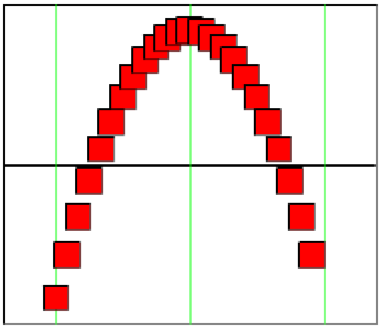
Saw Up:  moves linearly from start to end, then jumps back to start



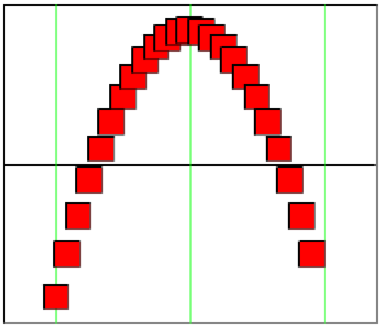
Saw Down: moves linearly from end to start, then jumps back to end



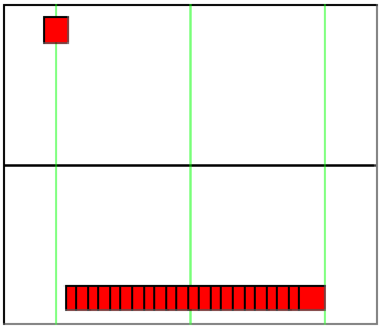
Sqr+Tri: moves linearly to end values, stays, move linearly to start
value, stay



Bounce: moves in a parabola from start to end values and back



Gravity: same as bounce, except, rather than repeating, the curve continues past the start point



Pulse: jumps to end value and back at the start of the cycle

Cycles/Period
Controls the number of Frames it takes for the wave to complete a cycle, or the number of cycles for the Effect duration.

If the Period setting is selected, the values refer to the number of Frames per cycle. The larger the period, the less cycles during the Effect and the slower the wave moves. Changing the Effect duration does not change the speed of the Effect, only how long it plays.

If the Cycles setting is selected, the values refer to number of cycles during the Effect. The larger the number of cycles, the longer the period and the faster the wave moves. Changing the Effect duration changes the speed of the Effect, not only how long it plays.

Note: When the Cycles setting is selected, the speed of animation can be controlled by adjusting the overall Effect duration. The shorter the duration, the faster the animation

Phase
This determines where the wave starts in it cycle. A value of 0 degrees starts at the start of the cycle, a value of 180 degrees starts halfway through, and so on.

Decay
The parameters in these columns determines how the waveform changes over time.

A (amplitude)
When checked, the amplitude (or strength or range) can decrease toward zero or increase toward the full value (depending on the direction). When this option is turned off, the amplitude stays the same for the duration of the Effect.

P (period)
When checked, the period (or wavelength) can decrease toward zero or increase toward the full value (depending on the direction). When this option is turned off, the period stays the same for the duration of the Effect.

Halflife
This is the number of Frames it takes for the amplitude or period to halve (or double) its value. The larger the half-life, the slower the rate of decay.

Direction
This is the direction and (for amplitude) the final value that the decay takes. The amplitude can decay toward (or from) the minimum value, maximum value or halfway between.

X=Y
This option forces the Y value to equal the X value.

Position X/Y
Controls the maximum distance each element moves from its original position. A negative X is left, and a positive X is right. A negative Y is up and a positive Y is down. A value of 0 means no change in position.

Spacing X/Y
Controls the element(X) / line(Y) spacing factors for the elements when the wave is farthest from its original position. A value of 100% means no change in spacing.

Scale X/Y
The amount by which each element is scaled horizontally/vertically as the wave passes through. Maximum change in scale will occur when the wave is farthest from its original position. A scale of 100% means the scale will not be varied. A negative value will shrink the element horizontally/vertically as wave passes through.

Angle X/Y
The amount by which each element's X/Y axis is rotated as the wave passes through. Maximum change in angle will occur when the wave is farthest from its original position. An angle of zero means the axis will not be rotated.

Alpha
The amount by which each element is faded out as the wave passed through. Maximum fade out will occur when the wave is farthest from its original position. An alpha of 100% means the alpha will not change. A value of 0% means the element will completely fade out as the wave passes through.

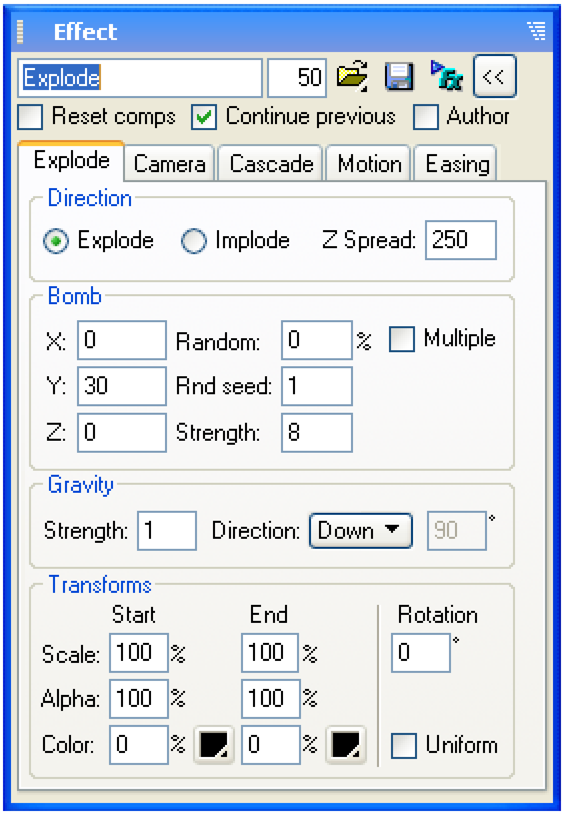
Color
This controls the color Transform for the element when the wave is farthest from its original position. The element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means no change in color.

4.4.4.5 Explode

The Explode Effect explodes or implodes the elements of a Complex Object.

The Explode Effect can be applied to a Text Object to blow apart the letters of the words. Letters can be made to spin, change size, fade out and fall to the ground. The strength and speed of the explosion can be altered, as can the position of the 'bomb'.

The 'Explode' setting tab of the 'Explode' Effect Panel is represented below. It can be displayed by double-clicking on a Explode Effect in the Timeline to display the Effect Setting Panel.



Effect

Explode 50

□ Reset comps ☑ Continue previous □ Author

Explode | Camera | Cascade | Motion | Easing

Direction

● Explode ○ Implode Z Spread: 250

Bomb

X: 0 Random: 0 % □ Multiple

Y: 30 Rnd seed: 1

Z: 0 Strength: 8

Gravity

Strength: 1 Direction: Down 90 °

Transforms

Start End Rotation

Scale: 100 % 100 % 0 °

Alpha: 100 % 100 %

Color: 0 % ■ 0 % ■ □ Uniform

Direction
Specifies an explosion or an implosion.

Z Spread: When zero, objects explode sideways only. Bigger values make objects also explode toward and away from you for a more realistic 3D explosion.

Bomb
Controls the strength and the position of the explosion.

- **Position (X,Y,Z)**: this is the center point of the explosion relative to the center of the object along the X, Y and Z axis. The default is 30 pixels below the center of the object, so the object appears to explode upwards. Modifying the Z position will make the explosion appear to explode in a forward and backward direction
- **Random**: offsets the bomb by a random amount as a percentage of the distance to the bomb
- **Rnd Seed**: this number is used as a seed for the random number generator. Changing this number will produce slightly different results. You can keep changing this value until it produces an explosion that is esthetically pleasing
- **Strength**: a value of 1 will produce a slow, weak explosion that is quickly overwhelmed by gravity. A value of 20 will produce a fast, powerful explosion that blows the fragments off the Movie in a few Frames. A value of 5-10 is a good approximation for a 'real' explosion
- **Multiple**: multiple bombs position relative to each element, or a single bomb positioned relative to the whole object

Gravity
Controls the strength and the direction of gravity.

- **Strength**: this is the strength of the force pulling the fragments downward. A value of 5 will quickly overwhelm a moderate-strength explosion. A value of 0.5 will have little effect on a moderate-strength explosion. A value of 1 is a good approximation for 'real' gravity
- **Direction**: controls the direction of gravity. Gravity may pull the fragments up, down, left, right, or any in any direction

Transforms
Controls the Transforms applied to elements as they move away from the bomb.

- **Scale**: this controls the scaling applied to elements at the Start/End of the Effect. Fragments that began nearer the bomb will grow faster. A scale value of 100% means the fragments do not change size
- **Alpha**: this controls the fading applied to elements at the Start/End of the Effect. A value of 100% means the alpha value of the fragments remains unchanged. A value of 0% means the fragments will fade out completely
- **Color**: this specifies the color Transform for the elements at the Start/End of the Effect. The element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means no change in color
- **Rotation**: this controls the amount of rotation applied to elements as they move away from the bomb. The X and Y axis of the fragments are rotated independently (resulting in skewing) unless **Uniform** is checked. A rotation value of zero means the fragments will not be rotated

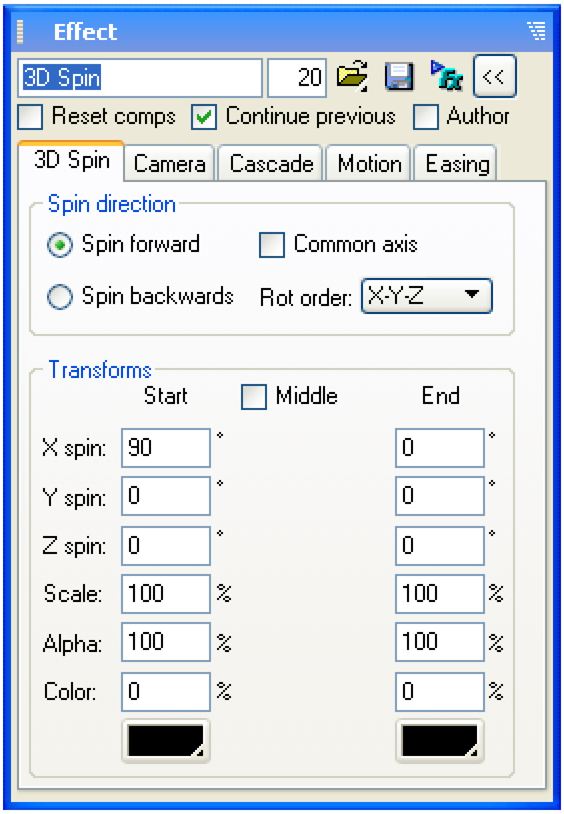
Note: For Text Objects, elements correspond to letters.

4.4.4.6 3D Spin

The 3D Spin Effect spins the elements of a Complex Object on the vertical or horizontal axis in 3D space over time.

The 3D Spin Effect can be applied to a Text Object to change the orientations of the elements over time. Elements can be made to face left, top or front at the beginning and turn a specific number of degrees around the vertical or horizontal axis for the duration of the Effect. The element animations can follow one after another, or they can overlap. The 3D-Spin Effect can be applied forwards or backwards, and elements can also be scaled and faded.

The '3D Spin' setting tab of the '3D Spin' Effect Panel is represented below. It can be displayed by double-clicking on a 3D Spin Effect in the Timeline to display the Effect Setting Panel.



3D Spin 20

□ Reset comps ☑ Continue previous □ Author

Spin direction

● Spin forward □ Common axis

○ Spin backwards Rot order: X-Y-Z

Transforms

	Start		□ Middle	End	
X spin:	90	°		0	°
Y spin:	0	°		0	°
Z spin:	0	°		0	°
Scale:	100	%		100	%
Alpha:	100	%		100	%
Color:	0	%		0	%
	■			■	

Spin Direction
Controls the initial orientations of elements.

- **Spin forwards**: the elements are animated forwards from Start Transform to End Transform
- **Spin backward**: the elements are animated backwards from End Transform to Start Transform
- **Common axis**: the elements will spin relative to the object's center point instead of their own center points
- **Rotation order**: the order is which rotations are applied

Transforms
Controls the Transforms applied to elements as they spin.

- **X Spin**: specifies the angle the element is rotated around the X axis at the start, middle and end of the Effect
- **Y Spin**: specifies the angle the element is rotated around the Y axis at the start, middle and end of the Effect
- **Z Spin**: specifies the angle the element is rotated around the Z axis at the start, middle and end of the Effect
- **Scale**: specifies how much the element is scaled at the start, middle and end of the Effect. A value of 100% means no change in scale
- **Alpha**: specifies how much the element is faded at the start, middle and end of the Effect. A value of 100% means the alpha of the element is unaffected. A value of 0% means the element is completely faded out

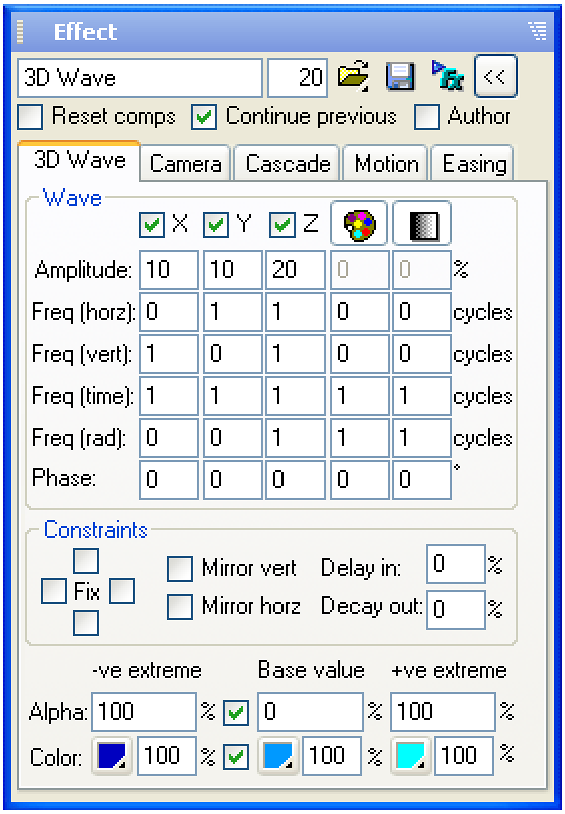
- **Color**: this specifies the color Transform for the element at start, middle and end of the Effect. The element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means no change in color

4.4.4.7 3D Wave

The 3D Wave Effect applies a rolling wave through the elements of a Complex Object in three dimensions.

The 3D Wave Effect can be applied to a Text Object to produce waving banners. The 3D Wave Effect can move letters and shapes up and down or side-to-side, rotating and scaling them as required to give the impression of 3D undulations. The color and alpha of each letter or shape can be varied as a function of wave height. The wave can be clamped to edges of the stage for special effects such as flag waving. Cyclic image warping can be achieved by breaking an image into pieces and passing a 3D wave through the resulting shape.

The '3D Wave' setting tab of the '3D Wave' Effect Panel is represented below. It can be displayed by double-clicking on a 3D Wave Effect in the Timeline to display the Effect Setting Panel.



3D Wave 20

□ Reset comps ☑ Continue previous □ Author

Wave

	☑ X	☑ Y	☑ Z			
Amplitude:	10	10	20	0	0	%
Freq (horz):	0	1	1	0	0	cycles
Freq (vert):	1	0	1	0	0	cycles
Freq (time):	1	1	1	1	1	cycles
Freq (rad):	0	0	1	1	1	cycles
Phase:	0	0	0	0	0	°

Constraints

□ □ Fix □ □

□ Mirror vert Delay in: 0 %

□ Mirror horz Decay out: 0 %

	-ve extreme			Base value		+ve extreme	
Alpha:	100	%	☑	0	%	100	%
Color:	100	%	☑	100	%	100	%

X,Y
If these options are turned on, then the effect is applied to the horizontal (X)/vertical (Y) value of each element.

Z
If this option is turned on, then the effect is applied to the Z value of each element, which is the wave height or distance out from screen. Ensure that "Use projective projection" is selected in the Camera settings for best effect when using Z values.

Color
If this option is turned on, then the Color of each element is modified according to wave height. Different colors can be specified at the peaks, zero crossings and troughs of waves as follows:

- -ve extreme color,
- Base value color (optional), and
- +ve extreme color.

The percentages associated with each of these colors determine the amount of color to be blended based on the current wave height.

Alpha
If this option is turned on, then the Alpha value of each element is modified according to wave height. Different Alpha values can be specified at the peaks, zero crossings and troughs of waves as follows:

- -ve extreme Alpha,
- Base value Alpha (optional),
- +ve extreme Alpha.

The percentages associated with each of these values determine the amount of Alpha to be multiplied based on the current wave height.

Amplitude
Specifies the size of the wave in each dimension as a percentage of the object's overall size.

Freq (horz)
The number of cycles to be applied based on each element's horizontal position.

Freq (vert)
The number of cycles to be applied based on each element's vertical position.

Freq (time)
The number of cycles to be applied over the duration of the animation. This must be a whole number if a cyclic animation in which all elements return to their starting position is required.

Freq (rad)
The number of cycles radiating outwards from the object's midpoint. Use this value to create waves that spread out in concentric rings like ripples in water.

Notes on frequency settings: The behaviour of the 3D wave is defined by a combination of all four frequency values, however the overall frequency can be adjusted conveniently using the master frequency value Freq (Time). Freq (Horz) and Freq (Vert) values can be combined for interesting effects. Adjusting values in opposed pairs (Horz Y with Vert X) creates regular sway, in coincident pairs (Horz X with Vert Y) creates pulses aligned to the axes, and in mixed combinations creates billowing undulations. The direction of wave travel can be reversed by using negative frequencies. Setting the Time frequency to 0 results in a stationary wave. This is useful for effects such as fixing blended colors in place across the elements of an object.

Phase
Phase offset that defines the starting point of the wave in degrees. The wave height can be set to start anywhere between -1 and +1 using the appropriate phase offset.

Fix
Constraint that dampens the wave as it approaches edges of the stage. Check the left, top, right and/or bottom boxes to specify which edges to dampen. This setting is useful for creating waving flags or undulations within the center of an object that do not disturb its edges.

Mirror vert/horz
If these items are checked then the wave is reflected vertically and/or horizontally about the object's midpoint. Use this option to create waves that bulge outwards to the left and right, or up and down.

Delay in/Decay out
If these items are checked then the wave height is dampened to zero at the start/end of the animation. The delay/decay takes effect once the specified percentage of duration is reached.
Delay in and Decay out should be used sparingly. Due to the nature of the transformations being applied to elements, the transitions between cycles may appear slightly discontinuous if excessive delay and decay is used.

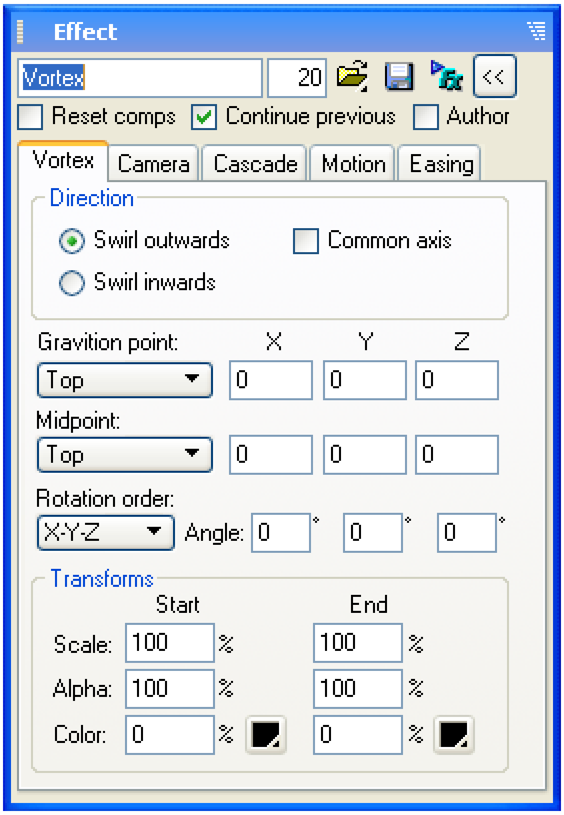
-ve extreme / Base Value / +ve extreme
See 'Alpha' and 'Color' sections above.

4.4.4.8 Vortex

The Vortex Effect spins the elements of a Complex Object in a 3D space and pulls them down into a gravitation point (or blows them out from a gravitation point) over time.

The Vortex Effect can be applied to a Text Object to attract the elements of text into a gravitation point (or blows them out from a gravitation point). Elements can be made to spin, change size and fade out. The location of the gravitation point and the midpoint of motion track can be altered. The element animations can follow one after another, or they can overlap. The Vortex Effect can be applied forwards or backwards.

The 'Vortex' setting tab of the 'Vortex' Effect Panel is represented below. It can be displayed by double-clicking on a Vortex Effect in the Timeline to display the Effect Setting Panel.



Vortex 20

☐ Reset comps ☑ Continue previous ☐ Author

Direction

● Swirl outwards ☐ Common axis

○ Swirl inwards

	X	Y	Z
Gravition point: Top	0	0	0
Midpoint: Top	0	0	0
Rotation order: X-Y-Z Angle:	0 °	0 °	0 °

Transforms

	Start	End
Scale:	100 %	100 %
Alpha:	100 %	100 %
Color:	0 %	0 %

Direction
Controls the direction of swirling.

- **Swirl Outwards**: the elements of a Complex Object are blown out from a gravitation point over time
- **Swirl Inwards**: the elements of a Complex Object are attracted into a gravitation point over time

- **Common Axis**: the elements will spin relative to the object's center point instead of their own center points

Gravitation Point

Specifies the location of the Gravitation point. Note each (1) unit is equal to 1 pixel and the origin is at the center of the Object.

- **Custom**: Gravitation point = (X, Y, Z), where the X axis points left, the Y axis points down, the Z axis points to the front,
- **Left**: Gravitation point = (left of text - element width/2 + X, Y, Z)
- **Right**: Gravitation point = (right of text + element width/2 + X, Y, Z)
- **Top-Left**: Gravitation point = (left of text - element width/2 + X, top of text - element height/2 + Y, Z)
- **Top**: Gravitation point = (X, top of text - element height/2 + Y, Z)
- **Top-Right**: Gravitation point = (right of text + element width/2 + X, top of text - element height/2 + Y, Z)
- **Bottom-Left**: Gravitation point = (left of text - element width/2 + X, bottom of text + element height/2 + Y, Z)
- **Bottom**: Gravitation point = (X, bottom of text + element height/2 + Y, Z)
- **Bottom-Right**: Gravitation point = (right of text + element width/2 + X, bottom of text + element height/2 + Y, Z)
- **Current X-Y**: Gravitation point = (element's x + X, element's y + Y, Z)
- **Current X**: Gravitation point = (element's x + X, Y, Z)
- **Current Y**: Gravitation point = (X, element's y + Y, Z)

Midpoint

Specifies the Midpoint of the motion track.

- **Custom**: Midpoint = (X, Y, Z)
- **Left**: Midpoint = (left of text - element width/2 + X, Y, Z)
- **Right**: Midpoint = (right of text + element width/2 + X, Y, Z)
- **Top-Left**: Midpoint = (left of text - element width/2 + X, top of text - element height/2 + Y, Z)
- **Top**: Midpoint = (X, top of text - element height/2 + Y, Z)
- **Top-Right**: Midpoint = (right of text + element width/2 + X, top of text - element height/2 + Y, Z)
- **Bottom-Left**: Midpoint = (left of text - element width/2 + X, bottom of text + element height/2 + Y, Z)
- **Bottom**: Midpoint = (X, bottom of text + element height/2 + Y, Z)
- **Bottom-Right**: Midpoint = (right of text + element width/2 + X, bottom of text + element height/2 + Y, Z)
- **Current X-Y**: Midpoint = (element's x + X, element's y + Y, Z)
- **Current X**: Midpoint = (element's x + X, Y, Z)
- **Current Y**: Midpoint = (X, element's y + Y, Z)
- **Cur. X-Y only**: Midpoint = (element's x + X, element's y + Y, none)
- **Cur. X only**: Midpoint = (element's x + X, none, none)
- **Cur. Y only**: Midpoint = (none, element's y + Y, none)
- **Custom X-Y**: Midpoint = (X, Y, none)
- **Custom X-Z**: Midpoint = (X, none, Z)
- **Custom Y-Z**: Midpoint = (none, Y, Z)
- **Custom X**: Midpoint = (X, none, none)
- **Custom Y**: Midpoint = (none, Y, none)
- **Custom Z**: Midpoint = (none, none, Z)
- **None**: no Midpoint (the motion track is a straight line)

Rotation order/Angle

Specifies the order and the X,Y,Z angles to rotate around the Spin Axes during the Effect. For example, if the order was set to Y-X-Z, the element would first rotate Y degrees around Y axis, then rotate X degrees around X axis and finally rotate Z degrees around Z axis.

Transforms
Controls the Transforms applied to elements as they spin.

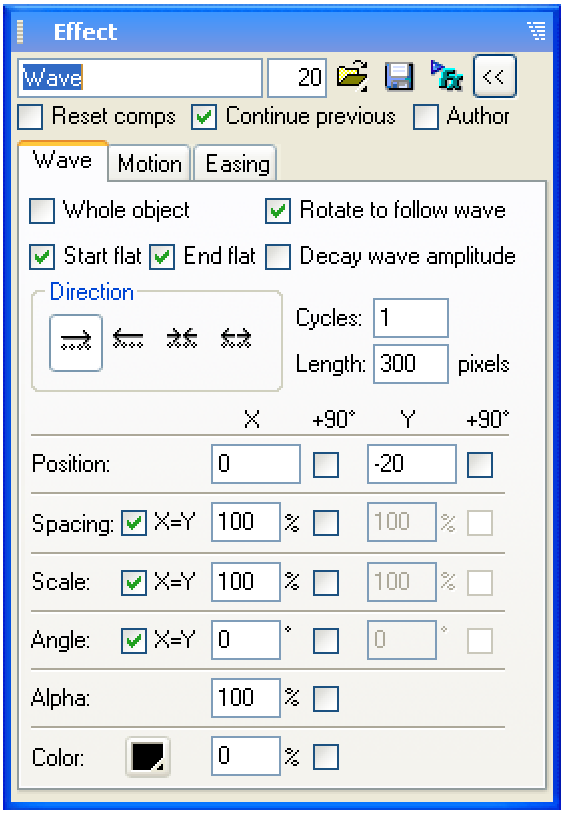
- **Scale**: specifies how much the element is scaled at the start/end of the Effect. A value of 100% means no change in scale
- **Alpha**: specifies how much the element is faded at the start/end of the Effect. A value of 100% means the alpha of the element is unaffected. A value of 0% means the element will completely faded out
- **Color**: this specifies the color Transform for the element at start/end of the Effect. The element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means no change in color

4.4.4.9 Wave

The Wave Effect applies a rolling wave through the text or group of objects.

The Wave Effect can be applied to a Text Object to produce waving banners. The Wave Effect can move letters up and down or side-to-side. Letters can be rotated to follow the wave, and the scale and alpha of each letter can be varied as the wave passes through.

The 'Wave' setting tab of the 'Wave' Effect Panel is represented below. It can be displayed by double-clicking on a Wave Effect in the Timeline to display the Effect Setting Panel.



Effect

Wave 20

☐ Reset comps ☑ Continue previous ☐ Author

☐ Whole object ☑ Rotate to follow wave

☑ Start flat ☑ End flat ☐ Decay wave amplitude

Direction

Cycles: 1

Length: 300 pixels

		X		+90°	Y		+90°
Position:		0		☐	-20		☐
Spacing:	☑ X=Y	100	%	☐	100	%	☐
Scale:	☑ X=Y	100	%	☐	100	%	☐
Angle:	☑ X=Y	0	°	☐	0	°	☐
Alpha:		100	%	☐			
Color:	■	0	%	☐			

Whole Object
If this option is turned on, the Effect applies to the object as a whole, and not to the individual elements.

Rotate to follow wave
If this option is turned on, the elements will rotate to follow the direction of the wave. If you have also specified an Angle in the Transform tab, then this is added on to the calculated angle.

Start flat
All elements will start in a straight position for the Wave Effect. Otherwise elements will appear to be waving at the start of the Effect.

End flat
All elements will end in a straight position for the Wave Effect. Otherwise elements will appear to be waving at the end of the Effect.

Decay wave amplitude
If this option is turned on, the amplitude of the wave gradually decreases to zero over the duration of the Effect.

Cycles
The number of cycles of the complete wave to occur in the given number of Frames.

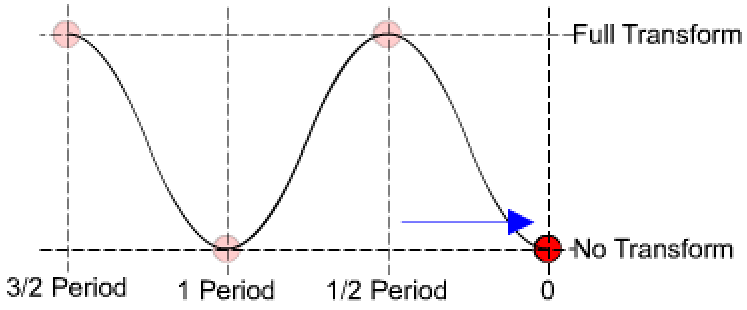
Length
The distance in pixels from the start to the end of the wave. For example, say a Text Object is 100 pixels long. If the length is set to 100 you will experience one full wave, set to 50 you will experience 2 full waves in the same length etc.

Direction
Controls the direction of the rolling wave.

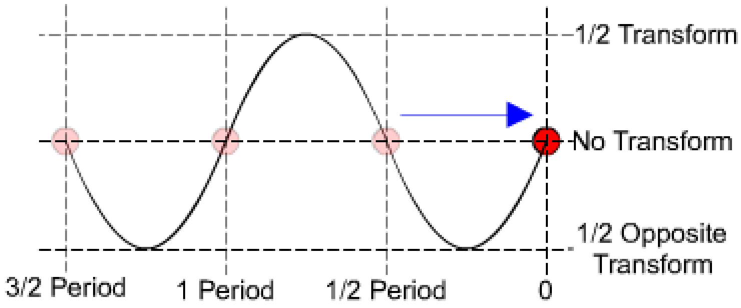
- →: the wave begins from the left side
- ←: the wave begins from the right side
- →←: the wave begins from the two sides
- ←→: the wave begins from the center

+90°
Shifts the phase of the wave by 90°.

No Shift:



Shift by 90°:



X=Y
This option forces the Y value to equal the X value.

Position X/Y
Controls the maximum distance each element moves from its original position. A negative X is left, and a positive X is right. A negative Y is up and a positive Y is down. A value of 0 means no change in position.

Spacing X/Y
Controls the element(X) / line(Y) spacing factors for the element when the wave is farthest from its original position. A value of 100% means no change in spacing.

Scale X/Y
The amount by which each element is scaled horizontally/ vertically as the wave passes through. Maximum change in scale will occur when the wave is farthest from its original position. A scale of 100% means the scale will not be varied. A negative value will shrink the element horizontally/vertically as the wave passes through.

Angle X/Y
The amount by which each element's X/Y axis is rotated as the wave passes through. Maximum change in angle will occur when the wave is farthest from its original position. An angle of zero means the axis will not be rotated.

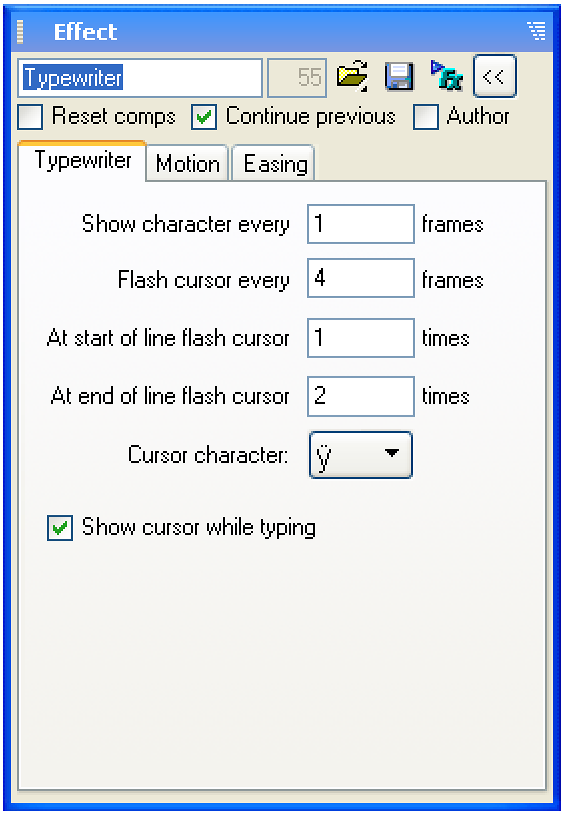
Alpha
The amount by which each element is faded out as the wave passed through. Maximum fade out will occur when the wave is farthest from its original position. An alpha of 100% means the alpha will not varied. A value of 0% means the element is completely faded out as the wave passes through.

Color
This controls the color Transform for the element when the wave is farthest from its original position. The element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means no change in color.

4.4.4.10 Typewriter

The Typewriter Effect shows the elements of a Complex Object one after another, like someone typing text onto the screen, with an optional cursor character. The cursor will only appear when you apply the Typewriter Effect to a Text Object.

The 'Typewriter' setting tab of the 'Typewriter' Effect Panel is represented below. It can be displayed by double-clicking on a Typewriter Effect in the Timeline to display the Effect Setting Panel.



Effect

Typewriter 55

□ Reset comps ☑ Continue previous □ Author

Show character every 1 frames

Flash cursor every 4 frames

At start of line flash cursor 1 times

At end of line flash cursor 2 times

Cursor character: ÿ

☑ Show cursor while typing

Show character every 'n' frames
Controls the speed at which characters are displayed. A value of 1 will produce the fastest 'typing' possible. Fractions can be used in this field.

Flash cursor every 'n' frames
Controls the speed at which the cursor flashes. A value of 1 will produce the fastest flashing possible. Fractions can be used in this field.

At start of line flash cursor 'n' times
This feature determines the number of times the cursor flashes at the start of a line. A value of zero means the cursor will not flash at the start of a line.

At end of line flash cursor 'n' times
This feature determines the number of times the cursor flashes at the end of a line. A value of zero means the cursor will not flash at the end of a line.

Cursor Character
Lets you choose the cursor character for the Typewriter Effect. Choosing a space will result in no cursor being displayed. If the Typewriter Effect is being applied to a Text Object, the cursor font will be the same as the Text Object font.

Show cursor while typing
Controls whether the cursor is shown during the Effect. Not showing the cursor produces smaller .swf files.

4.4.5 Common Effect Settings

Many Effects share common settings that are displayed as tabs in the Effect Settings Panel.

These settings include:

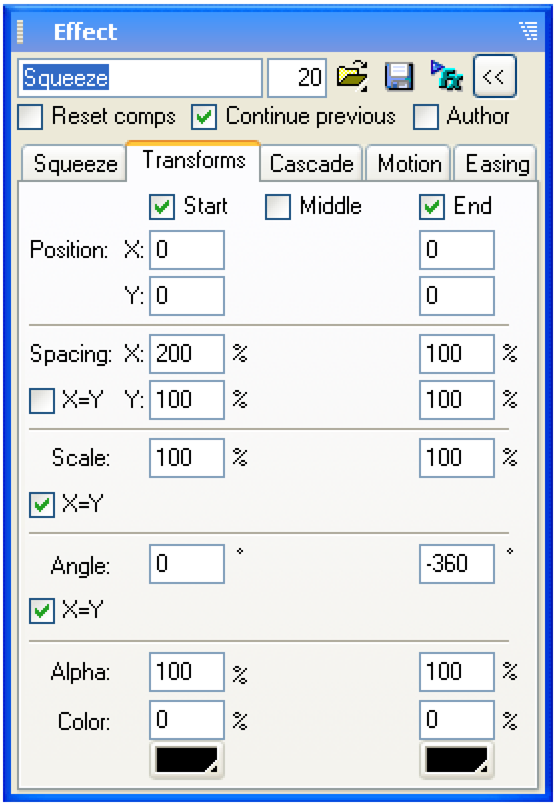
- Motion

- Easing
- Start At
- Cascade
- Camera
- Transforms.

Some of these tabs are not visible when first opening the Effect Settings Panel. To display all Effects press the 'More Tabs' button.

4.4.5.1 Transforms

The 'Transforms' tab sets the sequence of transforming animations to each element of a Complex Object. This tab is common to all the Basic Complex Effects.



Effect

Squeeze 20

☐ Reset comps ☑ Continue previous ☐ Author

	☑ Start	☐ Middle	☑ End
Position: X:	0		0
Y:	0		0
Spacing: X:	200 %		100 %
☐ X=Y Y:	100 %		100 %
Scale:	100 %		100 %
☑ X=Y			
Angle:	0 °		-360 °
☑ X=Y			
Alpha:	100 %		100 %
Color:	0 %		0 %

Name (shown above as "Squeeze"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

Start/Middle/End check boxes
Enables or disables the settings for the element at the start/middle/end of the Effect. When the checkbox for Start or End is cleared, then the Effect starts or ends with the initial positions. When the checkbox for Middle is cleared, then the middle value is always halfway between the start and end values and the Transform is linear.

Position X/Y
Specifies the X/Y position for the element at the start/middle/end of the Effect. The X/Y value is relative to the original position of the element. A negative X/Y means the new position is on the left/top of the original position;a positive X/Y means the new position is on the right/bottom of the original position; a value of zero means there is no change in Scale Factor.

Spacing X/Y
Specifies the element (X) and line (Y) spacing factors for the element at the start/middle/end of the Effect. A value of 100% means there is no change in Scale Factor.

Scale X/Y
Specifies the width (X)/height (Y) Scale Factor for the element at the start/middle/end of the Effect. A value of 100% means there is no change in Scale Factor.

Angle X/Y
Specifies the rotation angle for the element's X/Y axis at the start/middle/end of the Effect. A value of zero means there is no change in angle or rotation.

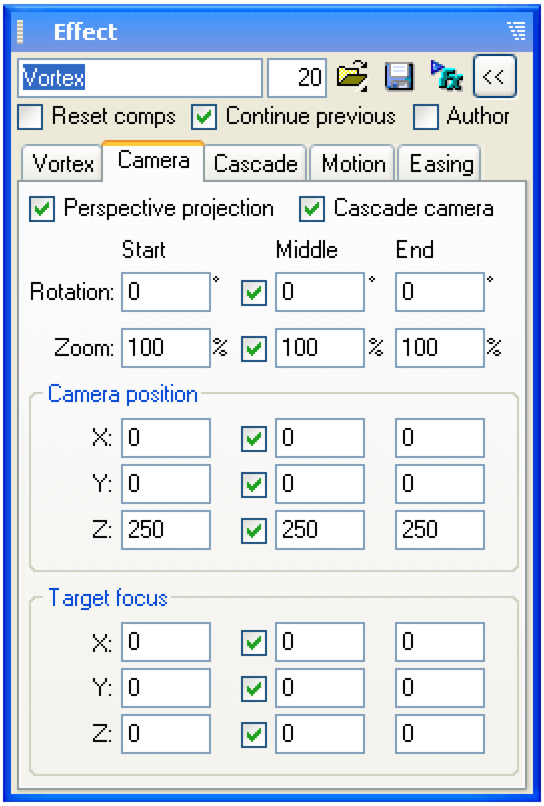
X=Y check boxes
Forces the Y value equal to the X value for Spacing, Scale or Angle.

Alpha
Specifies the change to the alpha value of the element at the start/middle/end of the Effect. A value of 100% means there is no change in Scale Factor.

Color percentage and button
Specifies the color Transform for the element at the start/middle/end of the Effect. The new element color is calculated by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color. A value of 0% means there is no change in color.

4.4.5.2 Camera

The 'Camera' tab adjusts the 3D viewing settings for the 3D Effects. By adjusting the parameters for the camera, the user is able to observe the Object in a 3D world from different angles or positions over time.



Effect

Vortex 20

☐ Reset comps ☑ Continue previous ☐ Author

Vortex | Camera | Cascade | Motion | Easing

☑ Perspective projection ☑ Cascade camera

	Start		Middle	End
Rotation:	0 °	☑	0 °	0 °
Zoom:	100 %	☑	100 %	100 %

Camera position

	Start		Middle	End
X:	0	☑	0	0
Y:	0	☑	0	0
Z:	250	☑	250	250

Target focus

	Start		Middle	End
X:	0	☑	0	0
Y:	0	☑	0	0
Z:	0	☑	0	0

Name (shown above as "Vortex"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

Use Perspective Projection
Determines whether 3D points are projected onto the Layout Panel using perspective rendering. When checked, specify a starting and ending Zoom Factor. Zoom Factors make no difference, and hence are not enabled, when the Perspective Projection option is turned off.

Cascade Camera
Causes the camera to maintain a fixed perspective. Unchecking this option causes the camera perspective to move in coordination with the effect settings.

Each of the following Camera options can be assigned a value for the Start, Middle and End of the Effect.

Rotation
Specifies the angle the camera is rotated around the view vector.

Zoom
Specifies the Zoom Factor of the camera. This is only valid for Perspective Projections.

Camera Position
Specifies the position of the camera, where the X axis points left, the Y axis points down, Z axis points to the front, 1 unit is equal to 1 pixel and the origin is at the center of the Object.

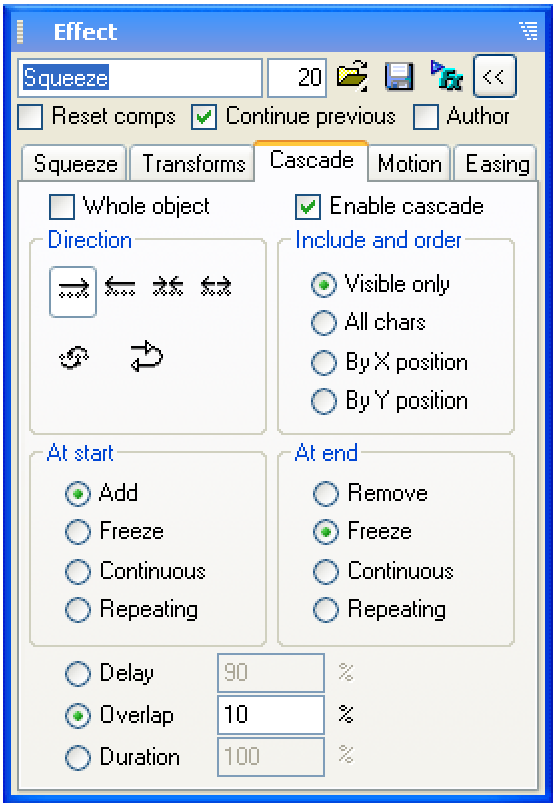
- **X**: the x coordinate of the camera's position
- **Y**: the y coordinate of the camera's position
- **Z**: the z coordinate of the camera's position

Target Focus
Specifies the point at which the camera is aimed.

- **X**: the x coordinate of the camera's target focus
- **Y**: the y coordinate of the camera's target focus
- **Z**: the z coordinate of the camera's target focus

4.4.5.3 Cascade

The 'Cascade' tab controls the order elements of a Complex Object are animated. Cascading animates elements one after another instead of all together.



Effect

Squeeze 20

□ Reset comps ☑ Continue previous □ Author

Squeeze | Transforms | Cascade | Motion | Easing

□ Whole object ☑ Enable cascade

Direction

Include and order
- ● Visible only
- ○ All chars
- ○ By X position
- ○ By Y position

At start
- ● Add
- ○ Freeze
- ○ Continuous
- ○ Repeating

At end
- ○ Remove
- ● Freeze
- ○ Continuous
- ○ Repeating

○ Delay 90 %

● Overlap 10 %

○ Duration 100 %

Name (shown above as "Squeeze"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

Note: If neither 'Whole' object or ' Enable Cascade' is checked then cascading is not applied to the Effect.

Whole Object
If ticked, then the Effect applies to the Object as a whole and not to the individual elements. The other controls on the tab do not apply and so are hidden.

Enable Cascade
If ticked, then the Cascade function is turned on and the relevant settings for this function revealed on the tab.

Direction
Instructs the Effect to animate the elements of a Complex Object sequentially following the specified order.

- →: forwards
- ←: backwards
- →←: from two sides to center
- ←→: from center to two sides
- ⟲: follows the specified direction but alternates between the direction specified and the opposite direction so the animations are interleaved
- ⮌: play Effect forward and then backward again

At Start

Specifies the start state of the elements.

- Add: the element is not visible until it is animated
- Freeze: all elements are visible at the start of the Effect, but do not animate until it is their turn in the cascade order
- Continuous: all elements are visible and animated in specified phases at the start of the Effect
- Repeating: repeats or loops cascading Effects

At End

Specifies the end state of the elements.

- Remove: the element is removed when the element animation ends
- Freeze: the element is left as it is when the element animation ends
- Continuous: the element animation is continued until the effect ends
- Repeating: repeats or loops cascading Effects

Include and Order

Applies the Effect to the elements of a Complex Object sequentially following the specified order.

- Visible only: animates the visible elements only. For text, this means that spaces and line breaks do not take part in the Cascade sequence
- All chars: animates all elements including space and line break characters
- By X position: animates the elements in the order of their X position. All elements with the same X position are animated at the same time
- By Y position: animates the elements in the order of their Y position. All elements with the same Y position are animated at the same time

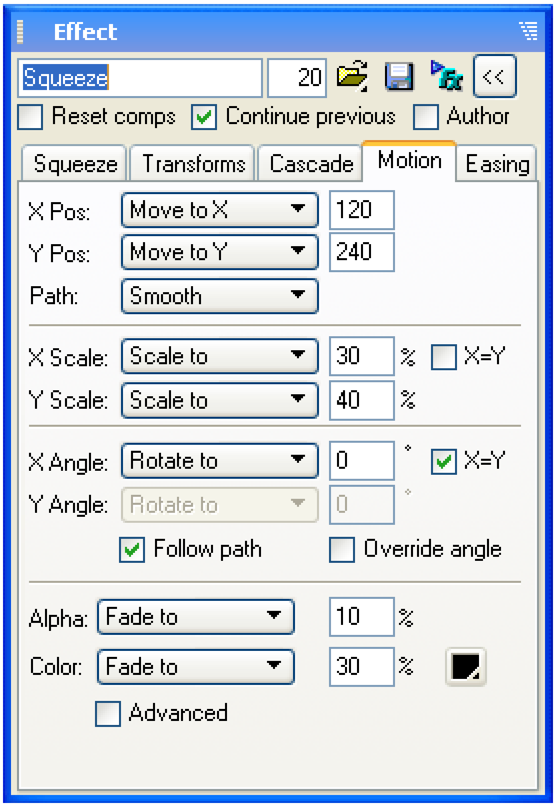
Delay/Overlap/Duration

Determines when the animation begins and ends for the elements of a Complex Object. Specify one of the three values, the other two are calculated and depend on the overall duration of the Effect and, more importantly, the number of elements in the Object.

- Delay: specifies the delay between each element animation as a percentage of the whole Effect's duration
- Overlap: specifies the overlap between each element animation as a percentage of the whole Effect's duration
- Duration: specifies the duration of a element animation as a percentage of the whole Effect's duration

4.4.5.4 Motion

The 'Motion' tab sets the position, scale, rotation, alpha (transparency) and color of the Object at the end of the Effect.



Squeeze 20

□ Reset comps ☑ Continue previous □ Author

X Pos: Move to X 120

Y Pos: Move to Y 240

Path: Smooth

X Scale: Scale to 30 % □ X=Y

Y Scale: Scale to 40 %

X Angle: Rotate to 0 ° ☑ X=Y

Y Angle: Rotate to 0 °

☑ Follow path □ Override angle

Alpha: Fade to 10 %

Color: Fade to 30 %

□ Advanced

Name (shown as "Squeeze"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

X/Y Position
X/Y Position is the coordinate of the selected Object's center. There are four selections for X/Y Position control:

- **Unchanged**: the X/Y position for the selected Object is unchanged
- **Move to X/Y**: moves the selected Object to the specified X/Y coordinate
- **Move Right/Down by**: moves the selected Object from the original position to the right and down by the specified distance
- **Move Left/Up by**: moves the selected Object from the original position to the left and up side by the specified distance.

Path at Key
Path at Key controls the shape of the Motion Path at the current Keyframe. There are two selections for Path at Key control:

- **Smooth**: the motion path is smooth at the current Keyframe
- **Sharp**: the motion path is cornered at the current Keyframe.

X/Y Scale
X/Y Scale is the scale factor of the selected Object's width/height. When the '**X=Y**' checkbox is checked the Object is scaled uniformly. When the function is turned off the Object can be stretched or shrunk in the X or Y directions independently. When scaling a Shape Object, the width of the line around the border of the shape is scaled by the maximum value of the X and Y factors. There are five selections for X/Y Scale control:

- **Unchanged**: the X/Y scale for the selected Object is unchanged
- **Scale to 100%**: changes the X/Y scale factor to 100%
- **Scale to**: changes the X/Y scale factor to the specified value

- **Increase by**: increases the X/Y scale factor by the specified percentage
- **Decrease by**: decreases the X/Y scale factor by the specified percentage.

X/Y Angle

X/Y Angle is the clockwise rotation angle of the selected Object's X/Y axis. With the '**X=Y**' checkbox ticked the Object can be rotated, when the checkbox is cleared, the Object can be skewed.

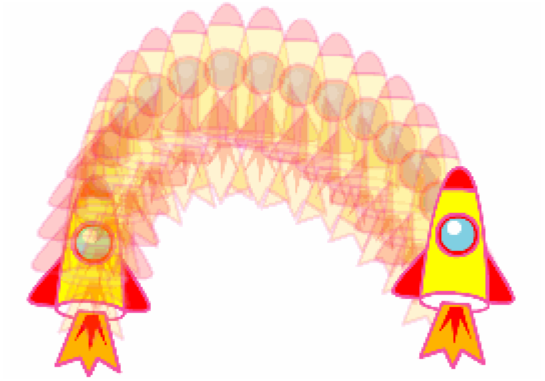
There are five selections for X/Y Angle control:

- **Unchanged**: the rotation angle of the selected Object's X/Y axis is unchanged
- **Rotate to Zero**: changes the rotation angle of the selected Object's X/Y axis to zero degree
- **Rotate to**: changes the rotation angle of the selected Object's X/Y axis to that specified
- **Rotate CW by**: rotates the selected Object's X/Y axis clockwise by the specified degrees
- **Rotate CCW by**: rotates the selected Object's X/Y axis counter-clockwise by the specified degrees.

'X=Y' This check box forces the Y value to be equal to the X value for Scale or Angle.

Follow path

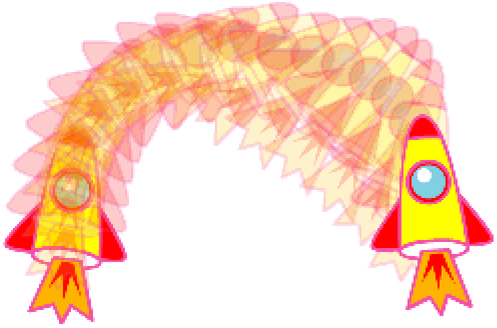
If this setting is not checked, the in-between orientation of the selected Object is decided by the X/Y Angle settings.



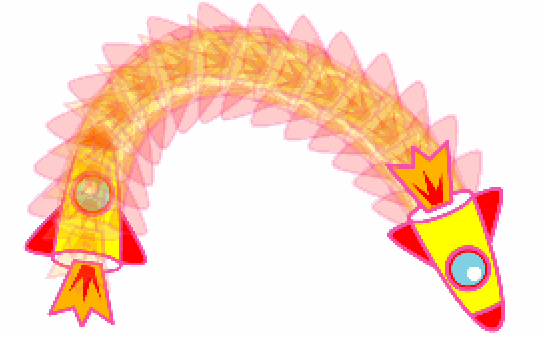
If this setting is checked, the orientation of the selected object will follow the tangent of the Motion Path while the Object is moving.

Override angle

If this setting is not checked, the in-between orientation is a blend of the original orientation, the tangent of the Motion Path and the settings of X/Y Angle. At the end of the Effect, the orientation of the object is always the same as the X/Y Angle settings.



If this setting is checked, when the selected Object is moving, the orientation always follows the tangent of the Motion Path and the X/Y Angle settings are ignored.



Alpha
Alpha controls the transparency of the selected object. There are six selections for Alpha control:

- **Unchanged**: the alpha (transparency) for the selected Object is unchanged
- **To Transparent**: changes the alpha value to 0%, making the Object completely transparent
- **To 100% Opaque**: changes the alpha value to 100%, making the Object completely opaque, with no transparent or semi-transparent areas
- **Fade to**: changes the alpha value to the specified value
- **Increase by**: increases the alpha value by the specified value
- **Decrease by**: decreases the alpha value by the specified value.

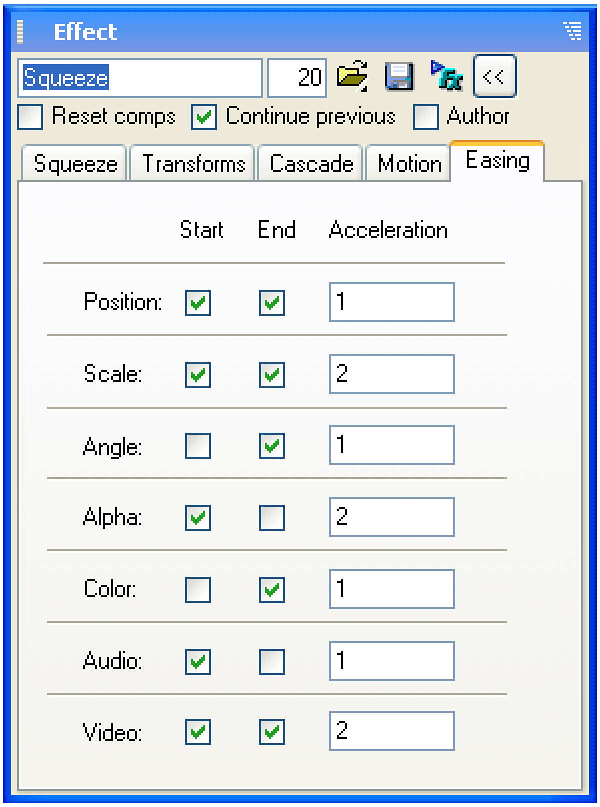
Color
The color option applies a color Transform to the selected Object. There are five selections for Color control:

- **Unchanged**: the color of the selected Object is unchanged
- **Fade to Black**: the Object is changed to black (even if the Object had multiple colors before)
- **Fade to White**: the Object is changed to white (even if the Object had multiple colors before)
- **Fade to** [Color]: the Object is changed to the specified color. The Object's color is determined by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color.

Advanced
The Advanced option specifies the new Alpha/Red/Green/Blue value by taking a percentage of the current Alpha/Red/Green/Blue value and adding a value to the result. This checkbox is only visible if the color transform is used.

4.4.5.5 Easing

The 'Easing' tab controls the position/scale/angle/alpha/color Transform speed at the start and at the end of the Effect. If the effect is applied to video or audio these options will also be active.



Effect

Squeeze 20

□ Reset comps ☑ Continue previous □ Author

	Start	End	Acceleration
Position:	☑	☑	1
Scale:	☑	☑	2
Angle:	□	☑	1
Alpha:	☑	□	2
Color:	□	☑	1
Audio:	☑	□	1
Video:	☑	☑	2

Name (shown as "Squeeze"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

Start
Accelerates the change at the start of the Effect.

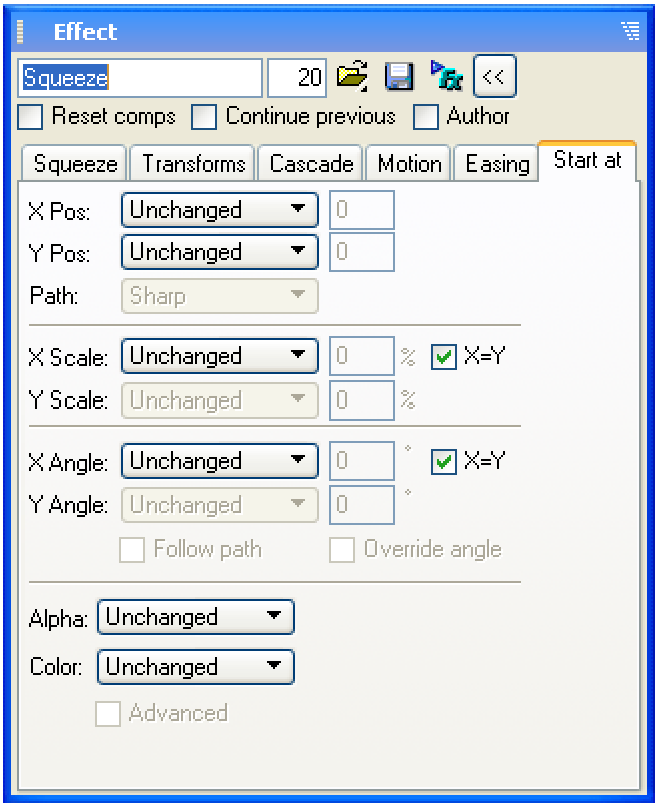
At End
Decelerates the change at the end of the Effect.

Acceleration
Controls the amount of acceleration or deceleration. A value of zero mean no acceleration. A positive value accelerates at the start or decelerates at the end. A negative value decelerates at the start of accelerates at the end.

4.4.5.6 Start At

The 'Start At' tab sets the position, scale, rotation, alpha (transparency), and color of the Object before the start of the Effect.

Note: The Start At tab is only displayed if the 'Continue from previous' effect checkbox on the Effect Settings Panel is unchecked.



Squeeze 20

□ Reset comps □ Continue previous □ Author

X Pos: Unchanged 0

Y Pos: Unchanged 0

Path: Sharp

X Scale: Unchanged 0 % ☑ X=Y

Y Scale: Unchanged 0 %

X Angle: Unchanged 0 ° ☑ X=Y

Y Angle: Unchanged 0 °

□ Follow path □ Override angle

Alpha: Unchanged

Color: Unchanged

□ Advanced

Name (shown as "Squeeze"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

X/Y Position
X/Y Position is the coordinate of the selected Object's center. There are four selections for X/Y Position control:

- **Unchanged**: the X/Y position for the selected Object is unchanged
- **Jump to X/Y**: places the selected Object to the specific X/Y coordinate
- **Jump Right/Down by**: moves the selected Object from the original position to the right and down by the specified distance
- **Jump Left/Up by**: moves the selected Object from the original position to the left and up by the specified distance.

For example, if a Typewriter Effect follows an Explode Effect, the Reset elements to original transforms option will reset the positions of the letters before the Typewriter Effect begins.

X/Y Scale
X/Y Scale is the scale factor of the selected Object's width/height. There are three selections for X/Y Scale control:

- **Unchanged**: the X/Y Scale for the selected Object is unchanged
- **Unscaled (100%)**: sets the X/Y Scale Factor to 100%
- **Scale of**: sets the X/Y Scale Factor to the specified value.

X/Y Angle
X/Y Angle is the clockwise rotation angle of the selected Object's X/Y axis. There are three selections for X/Y Angle control:

- **Unchanged**: the rotation angle of the selected Object's X/Y axis is unchanged
- **Zero angle**: sets the rotation angle of the selected Object's X/Y axis to zero degree
- **Angle of**: sets the rotation angle of the selected Object's X/Y axis to the specified degrees.

X=Y

These two check boxes force the Y value to be equal to the X value for Scale or Angle.

Alpha

Alpha controls the transparency of the selected Object. There are four selections for Alpha control:

- **Unchanged**: the alpha (transparency) for the selected Object is unchanged
- **Transparent**: sets the alpha value to 0, making the Object completely transparent
- **100% Opaque**: sets the alpha value to 100%, making the Object completely opaque, with no transparent or semi-transparent areas
- **Alpha Value of**: sets the alpha value to the specified value.

Color

The color setting applies a color Transform to the selected Object. There are five selections for Color control:

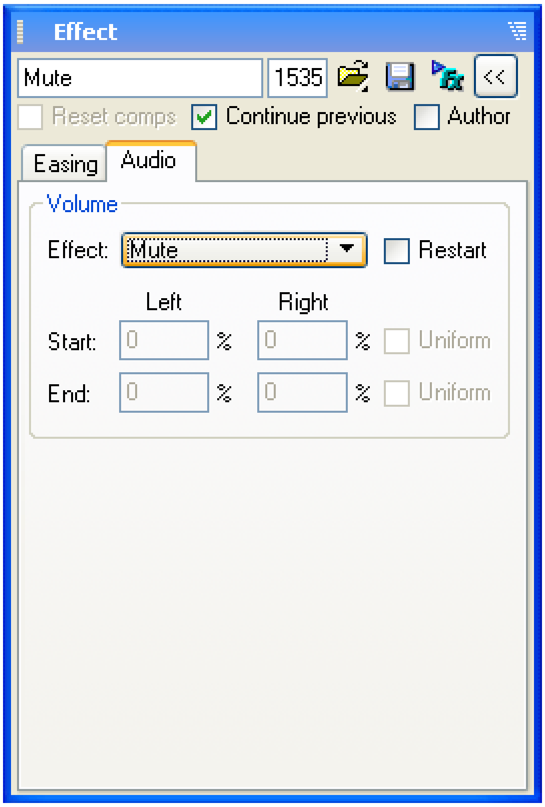
- **Unchanged**: the color of the selected Object is unchanged
- **Black**: the object appears black (even if the Object had multiple colors before)
- **White**: the object appears white (even if the Object had multiple colors before)
- **Color**: specifies the color of the Object. The Object's color is determined by taking the percentage of the selected color and adding the complementary percentage of the original color. For example, if the given percentage is 30%, the result will be 30% of the selected color mixed with 70% of the original color.

Advanced

The advanced setting specifies the new Alpha/Red/Green/Blue value by taking a percentage of the current Alpha/Red/Green/Blue value and adding a value to the result.

4.4.5.7 Audio

The 'Audio' tab sets the volume state at the start and end of the Effect.



Mute 1535

□ Reset comps ☑ Continue previous □ Author

Effect: Mute □ Restart

Start: 0 % 0 % □ Uniform

End: 0 % 0 % □ Uniform

Name (shown as "Fade Out"), Duration, Load, Save, Preview Effect, Reset comps, Continue Previous and Author are all described in the Effect Settings Panel.

Restart
Reset the volume to 100% on both channels before applying Effect.

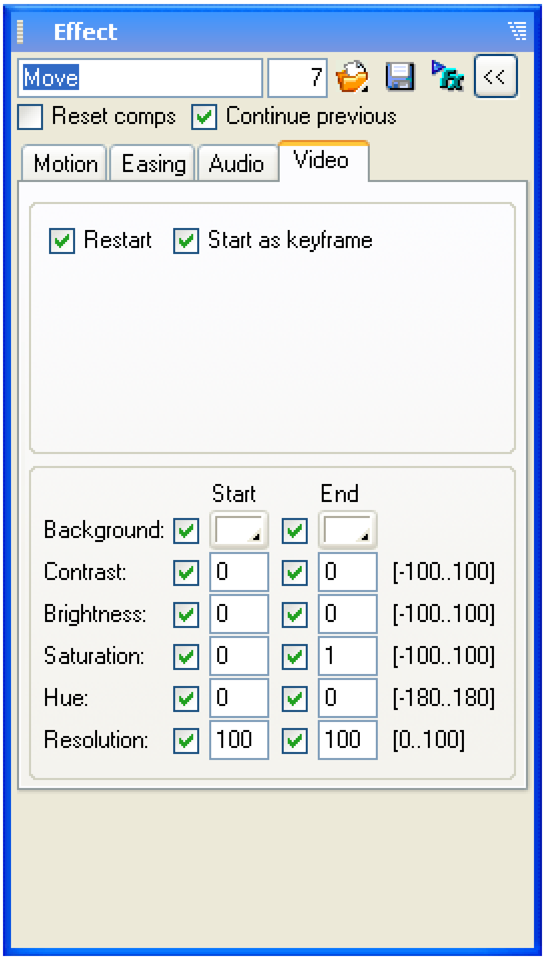
Effect
A number of options are available to change the sound volume envelope:

- **Unchanged**: Volume is not changed for the duration of the effect
- **Reset to 100%**: Reset volume to 100% on both channels
- **Mute**: Volume to 0% on both channels for the duration of effect
- **Fade In**: Ramp volume on both channel to 100% over the duration of the effect
- **Fade Out**: Ramp volume on both to 0% over the duration of the effect
- **Left channel only**: Turn off Right channel, leaving only the Left channel
- **Right channel only**: Turn off Left channel, leaving only the Right channel
- **Pan left to right**: Sound crosses over (pans) from Left channel to Right channel
- **Pan right to left**: Sound crosses over (pans) from Left channel to Right channel
- **Jump to**: Volume jumps from the previous volume to an End volume (%) immediate and stays at End volume for the duration of the effect
- **Ramp to**: Volume ramps from the previous volume to an End volume (%) over the duration of the effect
- **Custom**: Volume ramps from a Start volume (%) to an End volume (%) over the duration of the effect

Note: If the sound is in mono some options will not be selectable: Left Channel only, Right Channel only, Pan Left to Right, Pan Right to Left.

4.4.5.8 Video

The 'Video' tab sets the state at the start and end of the Effect.



Move 7

☐ Reset comps ☑ Continue previous

☑ Restart ☑ Start as keyframe

		Start		End	
Background:	☑		☑		
Contrast:	☑	0	☑	0	[-100..100]
Brightness:	☑	0	☑	0	[-100..100]
Saturation:	☑	0	☑	1	[-100..100]
Hue:	☑	0	☑	0	[-180..180]
Resolution:	☑	100	☑	100	[0..100]

Restart
Reset the volume to 100% on both channels before applying Effect.

Start as keyframe
Forces the video frame at the start of the effect to be exported as a keyframe.

Background
Changes the background color of the video from start to end, over the duration of the effect.

Contrast
Changes the contrast of the video from start to end, over the duration of the effect.

Sets the contrast at the start and end of the effect

Brightness
Changes the brightness of the video from start to end, over the duration of the effect.

Saturation
Changes the saturation of the video from start to end, over the duration of the effect.

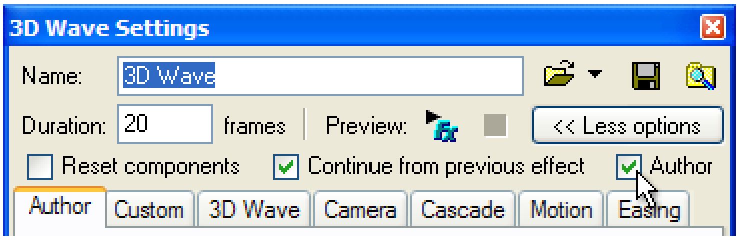
Hue
Changes the hue of the video from start to end, over the duration of the effect.

Resolution
Changes the resolution of the video from start to end, over the duration of the effect.

4.4.6 Effects Authoring

Use the Effects Authoring to build SWiSH Max Effects.

To author an Effect select the 'Author' tab in the Effect Settings Panel as shown below:



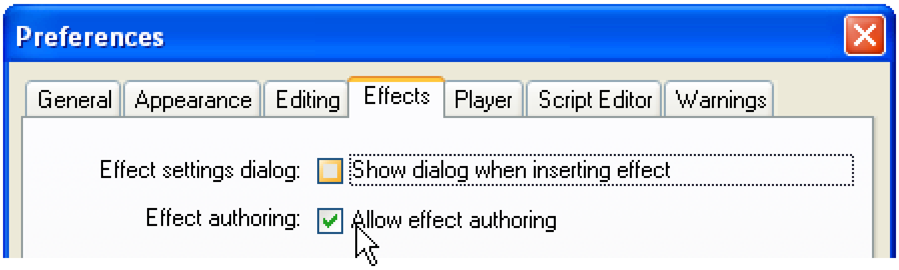
3D Wave Settings

Name: 3D Wave

Duration: 20 frames

□ Reset components ☑ Continue from previous effect ☑ Author

Note: The Author option is only available in the Effect Settings Panel if the 'Allow effect authoring' checkbox is checked in the 'Tools | Preferences | Effects' dialog box as shown below:



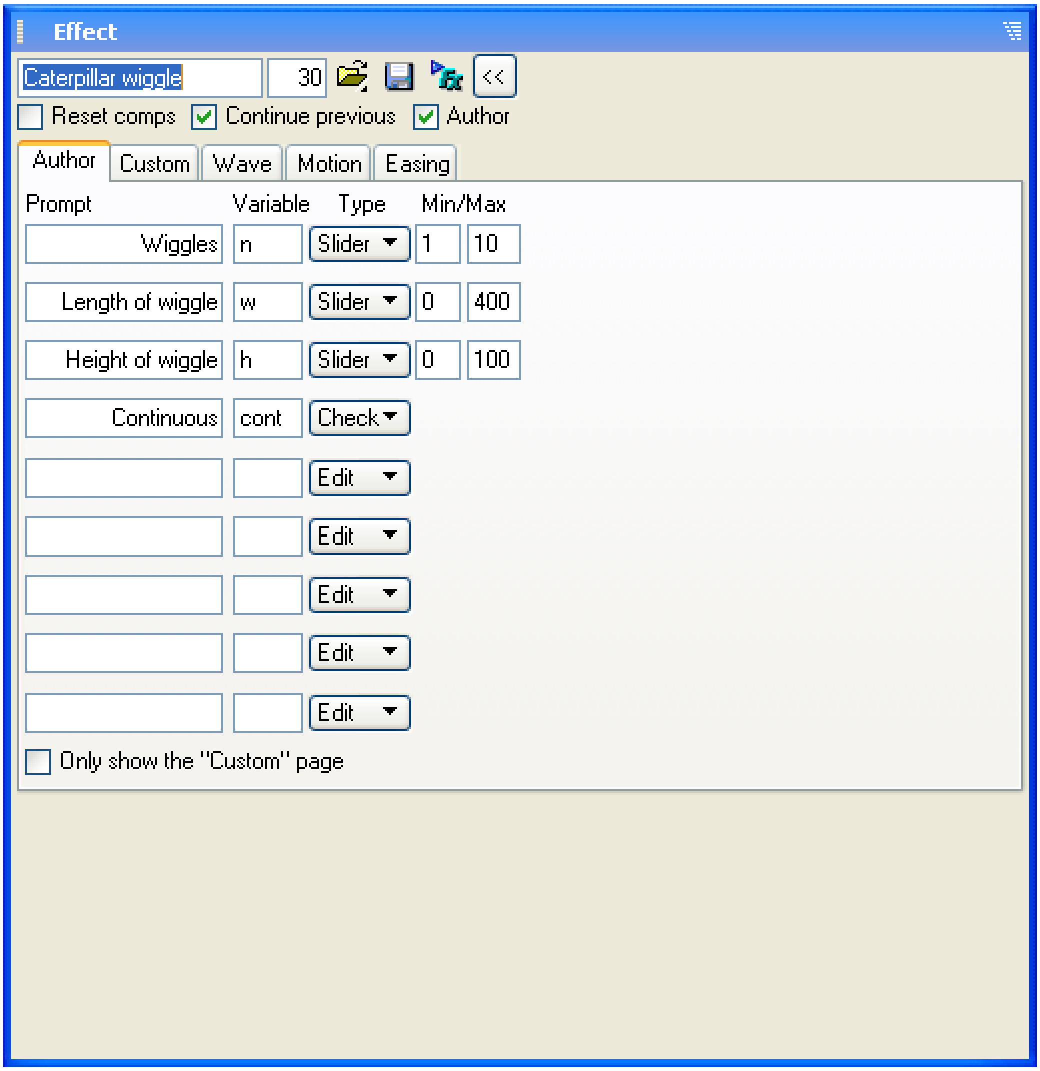
Preferences

Effect settings dialog: □ Show dialog when inserting effect

Effect authoring: ☑ Allow effect authoring

Once in author mode, two additional tabs, 'Author' and 'Custom', become available in the Effect Settings Panel. Use these tabs to add additional formulas and settings to Effects.

Author
The 'Author' tab is the place to add variables to be used in the Effect.



Caterpillar wiggle 30

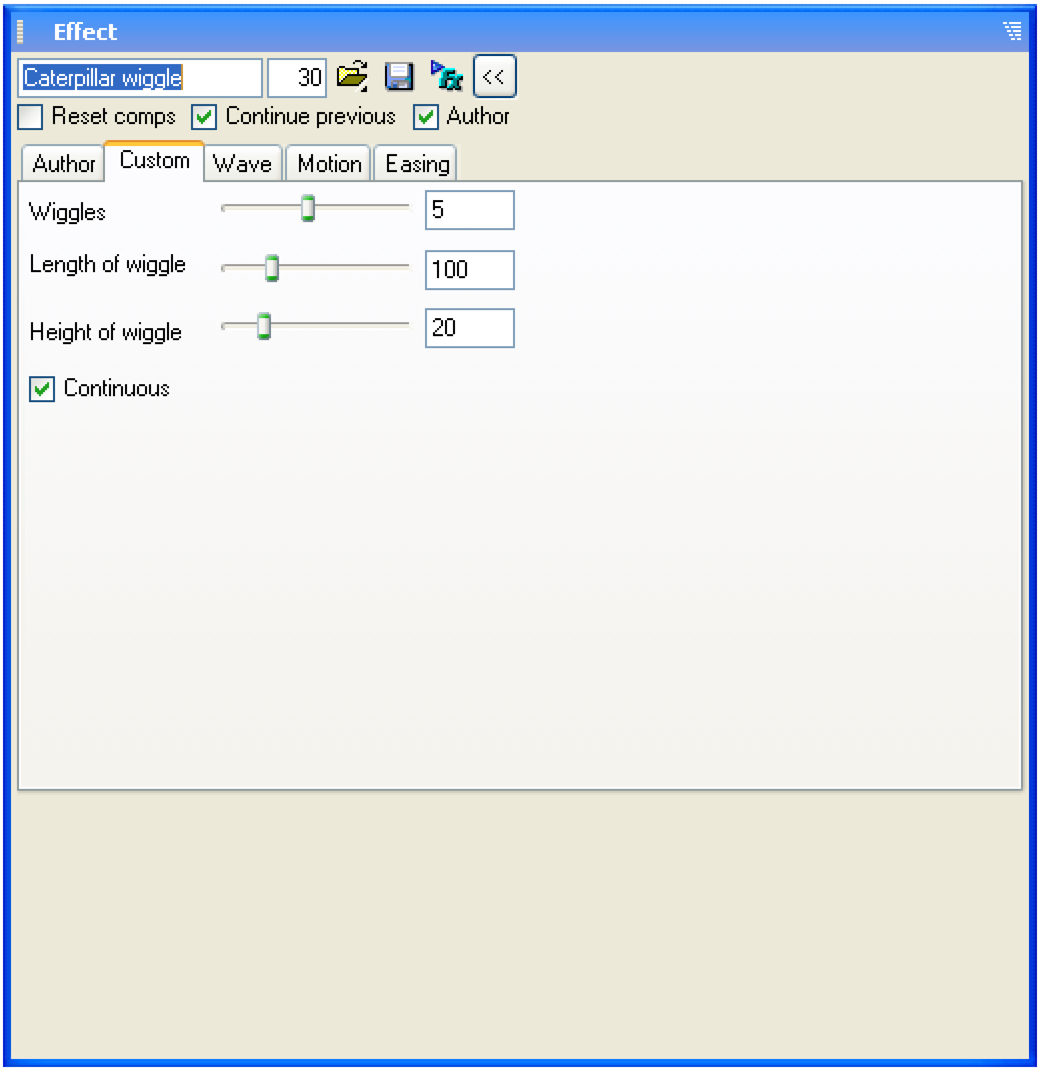
☐ Reset comps ☑ Continue previous ☑ Author

Prompt	Variable	Type	Min/Max	
Wiggles	n	Slider	1	10
Length of wiggle	w	Slider	0	400
Height of wiggle	h	Slider	0	100
Continuous	cont	Check		
		Edit		
		Edit		
		Edit		
		Edit		
		Edit		

☐ Only show the "Custom" page

- Prompt: the display name for the Effect variable
- Variable: the Effect variables name, for use in formulas
- Type: the Effect variable can either be an edit box, check box, slider or Color Picker
- Min/Max: the minimum and maximum values of the variable, used for slider variables only
- Only show the "Custom" page: stops the 'Author' tab from being shown when the Effect is opened. For example if the custom effect is based on the Wave Effect the 'Wave' tab will be hidden.

Custom
Use the 'Custom' tab to enter default values for the Effects variables as shown below:



Caterpillar wiggle 30

□ Reset comps ☑ Continue previous ☑ Author

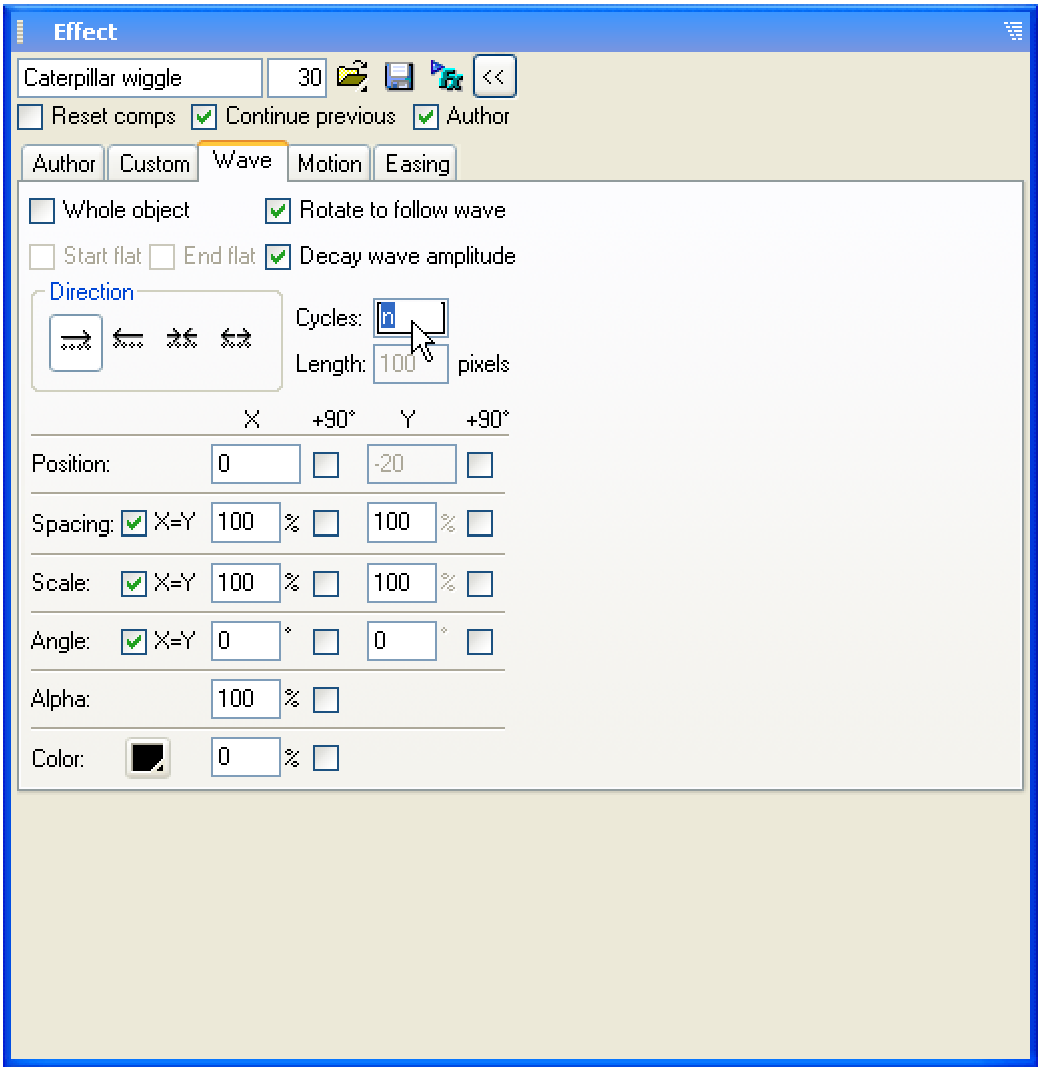
Wiggles 5

Length of wiggle 100

Height of wiggle 20

☑ Continuous

When in author mode, set the value of any field for the Effect to be one of the Effect variables or a formula containing a number of Effect variables. Effects variables and formulas are entered in a field by ***right-clicking the field***. In the example below the r-click on the 'Cycles' field sets the variable "n" which is changed when you slide the "Wiggles" controls. Grey items have been set to variables, r-click on these fields to edit the variable again.



Caterpillar wiggle 30

□ Reset comps ☑ Continue previous ☑ Author

□ Whole object ☑ Rotate to follow wave

□ Start flat □ End flat ☑ Decay wave amplitude

Cycles: n

Length: 100 pixels

		X	+90°	Y	+90°
Position:		0	□	-20	□
Spacing:	☑ X=Y	100 %	□	100 %	□
Scale:	☑ X=Y	100 %	□	100 %	□
Angle:	☑ X=Y	0 °	□	0 °	□
Alpha:		100 %	□		
Color:	■	0 %	□		

4.5 Components

A Component in SWiSH Max is a reusable, packaged module that adds a particular capability to a Flash movie. Components can include graphics as well as script code, so they are self-contained building blocks of functionality that you can easily drop into your projects. For example, a component can be a check box, a radio button, a dialog box, a pre-loader, or even something that has no graphics at all such as a timer.

Technically speaking a Component Object is a Movie Clip that presents important configuration parameters to the user via the Parameters Panel. This simplifies the re-use of Movie Clips in different applications.

A component can hide Child Objects and script from the user. This simplifies configuration. i.e. when a Component is added to a Movie all of the Child Objects and script belonging to the Component may not visible in the user-interface.

Components can have associated properties and parameters. The Properties are general attributes of the Object (e.g. name) as shown in the Properties Panel. Parameters are user selectable settings defined by the component creator. The only way to set the parameters of a Component is via the Parameters Panel.

A number of standard Components are supplied with SWiSH Max. To use a Component, simply drag it into your current project from the Components Panel.

New components can be authored for use in the current or future projects. See Authoring Components for more information.

Tutorials

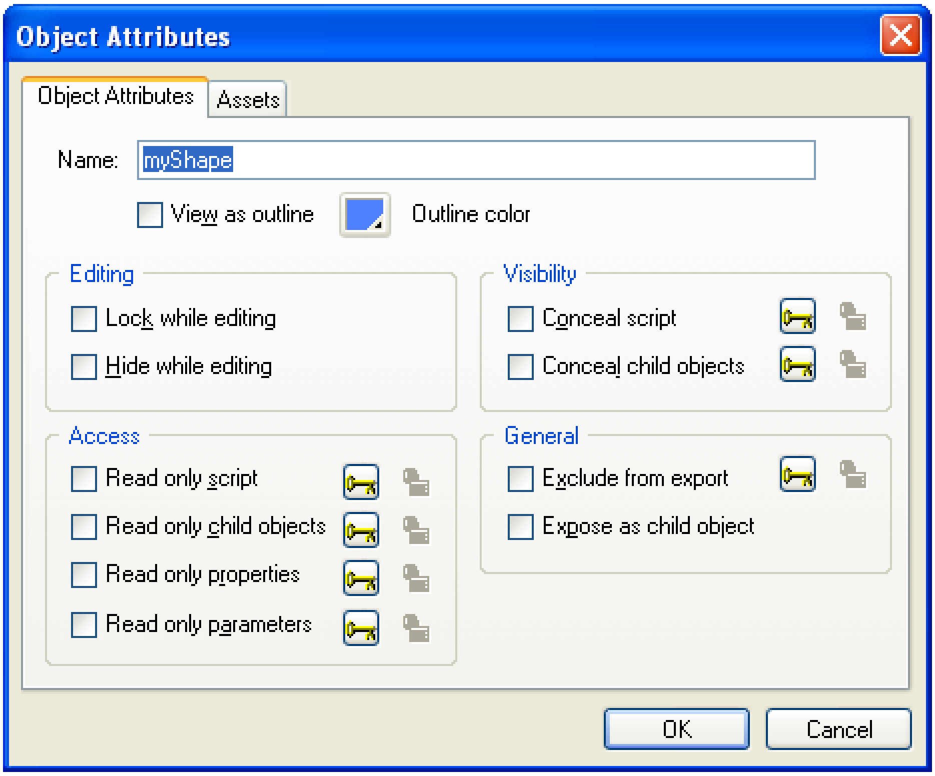
See the Component Tutorials section as an overview of how to use and write components.

4.5.1 Object Attributes

Modify | Object Attributes

This dialog allows the Object attributes to be set or cleared. Though applicable to all Object it is particularly important to the development of Components since locking attributes of a Component can secure it from unwanted change.

Attributes that have a key 🔑 button can have passwords applied to them so that they cannot be altered by other users unless the password is known. See Password Locking for more information.



Name: myShape

□ View as outline Outline color

Editing
□ Lock while editing
□ Hide while editing

Visibility
□ Conceal script
□ Conceal child objects

Access
□ Read only script
□ Read only child objects
□ Read only properties
□ Read only parameters

General
□ Exclude from export
□ Expose as child object

Note: for a description of the Assets tab on the Object Attributes dialog see the 'Asset' section of Library.

Name
Object name also shown in the Object Properties Panel (shown here as "myShape").

View as outlines and Outline color
This attribute if set shows the Object as an outline while in the edit mode. This can be useful to allow underlying objects to be shown while in the edit mode. This attribute can also be set by setting the rectangle icon in the Outline Panel. The color of the outline can be selected with the corresponding color selection tool.

There are four attribute groups: Editing, Access, Visibility and General.

Editing

These are attributes associated with the placement and editing of the Object.

Lock while editing This attribute prevents modification of the Object if it is set. The attribute can also be set by setting the padlock icon on the Outline Panel.

Hide while editing
If this attribute is set, the associated Object is hidden from the stage while editing. The attribute can also be set by setting the eye icon in the Outline Panel. The hide attribute can be used to temporarily hide a Object allowing editing of the Objects that are placed below it.

Access

These are attributes associated with the user accessing the parameters and properties of the Object.

Read only script
If set, this attribute prevents other users from modifying the script of the current Object. Note that the script of child Objects can still be modified if "Read only child objects" is not set and the attributes of a child object allow script modification.

Read only child objects
If set, prevents user modification of a child object name and other properties and parameters that may prevent the Object from working correctly. Note that some parameters such as color, font, etc. may still be editable.

Read only properties
If set, prevents the user from modifying any properties of the current Object.

Read only parameters
If set, prevents the user from modifying the component parameters via the component parameters panel.

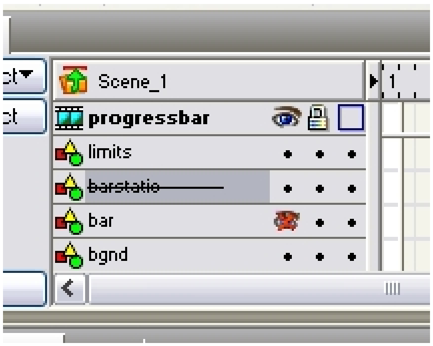
Visibility

Conceal script
This attribute, if set, conceals the script associated with the Object.

Conceal child objects
This attribute if set, conceals all child objects unless a specific child object has the "Expose as child object" attribute set. Setting this attribute prevents people from viewing or altering the contents of a Component.

General

Exclude from export
If set, prevents the current Object from being exported when the .swf file is created. The object appears with a strikethrough in the Outline panel view.



Object with strikethrough is excluded from export

This allows shapes to be added to a Component for the purposes of positioning or instruction without any overhead in the final exported movie.

Expose as child object
If set allows an object to be seen even if the parent object had the "Conceal child objects" attribute set. This is a convenient way to allow end user tailoring of Component parameters without exposing the bulk of the Component scripting and structure.

Note: If any of the following attributes are set, the user cannot access the Author Component dialog: Read only script, Read only properties, Read only parameters or Conceal script.

Useful combinations

Attributes can be used in various combinations to achieve different levels of protection of the intellectual property within your components. Some examples are given below:

1. High Security
Set and password protect the following attributes:

- Conceal script
- Conceal child objects
- Read only script - this is necessary to prevent access to the Author Component dialog.

With these options set, the Component can be resized and the Component parameters can still be modified however the internal structure of the component cannot be seen or modified.

2. Medium Security
Set and password protect the following attributes:

- Conceal child objects
- Read only child objects
- Read only script
- Select child objects have Expose as child object set

The Read only script attribute allows the methods defined in the parent script to be observed. The scripting within the parent object can also contain copyright information which cannot be altered by users without the password. Script that is intended to be kept secret can be placed in a child movie clip that is hidden. The child objects that have the "Expose as a child object" attribute set can be modified by the end user possibly to set final colors, font etc.

3. Trial Distribution
Use either of High or Medium Security options described above and add 'Exclude from export' attribute with a different password.
This configuration allows a users to paste the Component onto a .swi and examine how it works, however

the Component cannot be exported in a .swf until the password to unlock the 'Exclude from export' attribute is entered.

4.5.1.1 Password Locking

The state of many of an object's attributes can be fixed through the use of a password system. Once a password is applied to an attribute, the attribute state can only be changed with knowledge of the appropriate password. This allows the author to create and distribute .swi's and Components to other users but protect the intellectual property by setting attributes and applying passwords so that the end user cannot view script or internal structure of the Component.

Attributes that can be protected with a password are displayed with two icons. The first icon is a key 🔑. The second icon is a padlock 🔒.

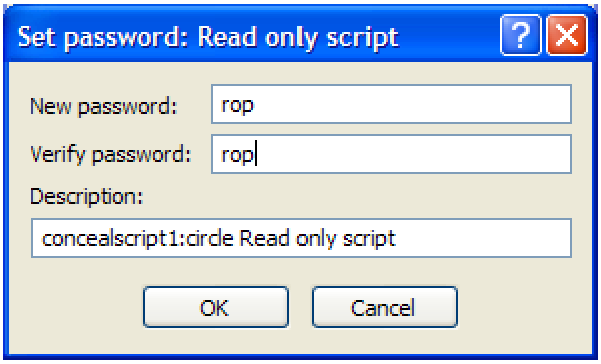
The shape and color of the padlock icon depends on the current state of the password applied to the object attribute:

- 🔒 - indicates the attribute has been locked. The attribute cannot be changed until the password is entered via the key icon. Entering a valid password will unlock the attribute and the icon will be displayed as an open padlock.
- 🔓 - indicates that a password has been assigned to the attribute, however the attribute is currently not locked and the status can be changed.
- ▨ - indicated a password is not associated with the attribute. In this state the attribute setting can be altered via the attribute checkbox. If needed, a password can be associated with the attribute via the key 🔑 button.

Adding a Password to an Attribute

To add a password to an attribute that does not have a password associated with it (as shown by ▨) press the key 🔑 button.

The following dialog will be displayed:



Set password: Read only script

New password: rop

Verify password: rop

Description:

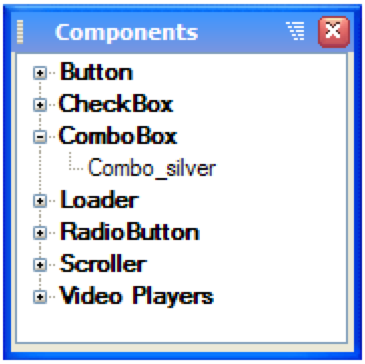
concealscript1:circle Read only script

Enter the password (in this case "rop") into the 'New password:' and 'Verify password:' fields. A description of the password is automatically entered. This description can be modified if desired. The description is initially based on the .swi name, the Object name and the attribute. In this case the .swi was named concealscript1.swi, the object was called circle and the attribute was "Read only script". The description allows the purpose of the password to be identified.

When the OK button is pressed, the password and description is entered into the Passwords section of the Tools | Preferences dialog. This allows the attribute to be modified without re-entry of the password.

4.5.2 Components Panel

The Components Panel allows the user to add a specific components to their current Movie.



In the default configuration, the Components Panel will appear in the bottom right hand corner along with the **Outline**, **Content** and **Effect** panels.

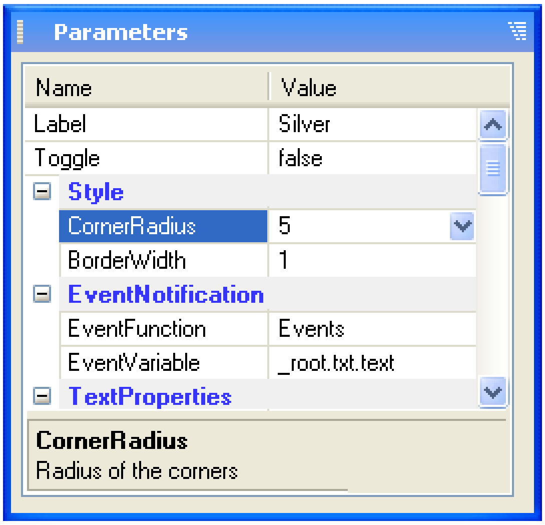
Components are grouped according to broad categories. The categories can be opened or closed with the **+** / **-** symbol. When opened, the individual Components are displayed.

To use a component, use the mouse to click-select and drag it from the the Parameters Panel onto the appropriate position in the Layout Panel.

When a Component is selected the Parameters Panel will become visible. The Parameters Panel is used to adjust the settings for the component.

4.5.3 Parameters Panel

The Parameters Panel allows the user to adjust the parameters for the selected component. The available parameters depend on the component that is selected.



Parameters

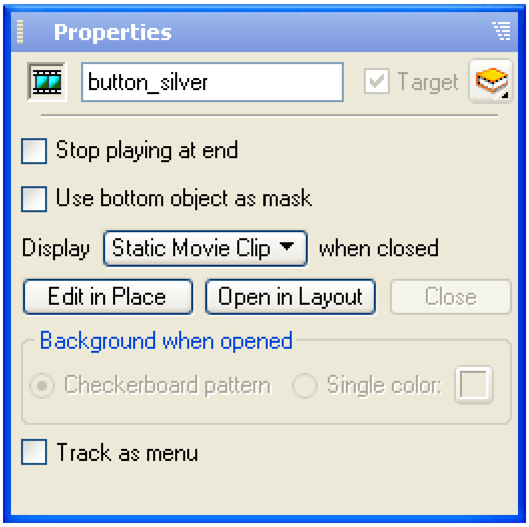
Name	Value
Label	Silver
Toggle	false
⊟ **Style**	
CornerRadius	5
BorderWidth	1
⊟ **EventNotification**	
EventFunction	Events
EventVariable	_root.txt.text
⊟ **TextProperties**	

CornerRadius
Radius of the corners

The columns lists the available parameters that can be altered. The **Name** column shows the current value of the parameter and allows alteration of that parameter. Depending on the Parameter and the Component, the value may be altered via either a **edit**, **list**, **combo**, **spin**, **color** or **group** control.

Underneath the Property / Name table, additional prompting is supplied for the currently selected parameter. In the case of the image above, the selected parameter 'CornerRadius' is used to define the "Radius of the corners". This field accepts a numeric value which will change the radius of curvature of the button corners.

Whenever a Component is selected the Properties for the Component will also be visible in another panel.



Properties

button_silver ☑ Target

□ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip ▾ when closed

Edit in Place | Open in Layout | Close

Background when opened

● Checkerboard pattern ○ Single color: □

□ Track as menu

Refer to the section on Properties Panel for more details but in this example the top section of the panel shows the type of component, (in this case it is a Movie Clip) and the component's name (currently "button_silver"). As Components are targets, the name should be chosen so that it does not clash with any names currently in use in your Movie.

4.5.4 Authoring Components

Components can be constructed from any SWiSH Max Movie. Simply save the .swi file to the components folder. The components folder is typically the folder named "components" that exists in the main SWiSH Max installation folder however this can be customized using Tools | Preferences | Components 'Components folder'. On a typical installations this will be 'c:\Program Files\SWiSH Max\components'.

To save a Component under an existing category, save it in the appropriate sub folder. New categories can be created by creating new sub folders named appropriately. For example, a folder named 'C:\Program Files\SWiSH Max\components\my testing' will cause a sub category "my testing" to be displayed in the Components Panel. After adding to the component folder, the Parameters Panel can be refreshed by pressing the 'F5' key.

To have the Component parameters modifiable via the Parameters Panel, the Movie should only contain one object at the Scene (or _root) level. This (single) Object could be a Group of Movie Clip containing other objects. User editable Component Parameters can be setup for the component through the Author Component Dialog. A knowledge of Scripting is required for this level of authoring.

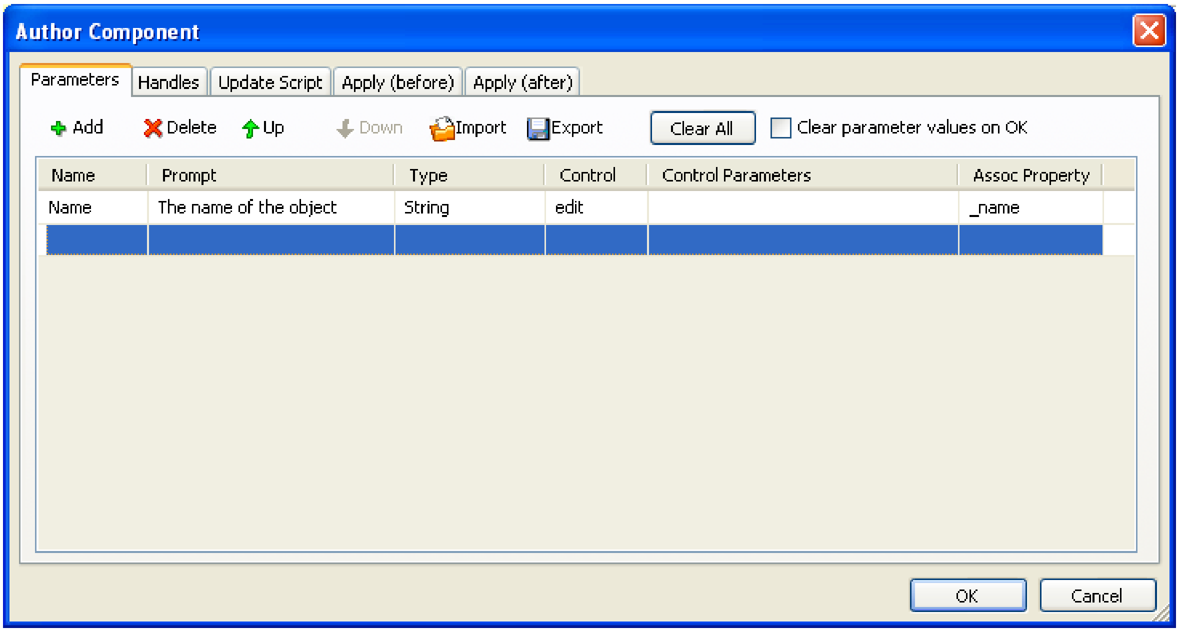
4.5.4.1 Author Component Dialog

Modify | Author Component ...(Ctrl+0)

To create a Component, select an Object then open the Author Component Dialog (as shown below). The dialog is also available via the mouse right-click context menu when an Object is selected. See sections on Components and Authoring components.

Each of the tabs in the Author Component dialog is discussed in detail in the following sections: Parameters tab, Scripting tab, Apply (before) tab, Apply (after) tab.

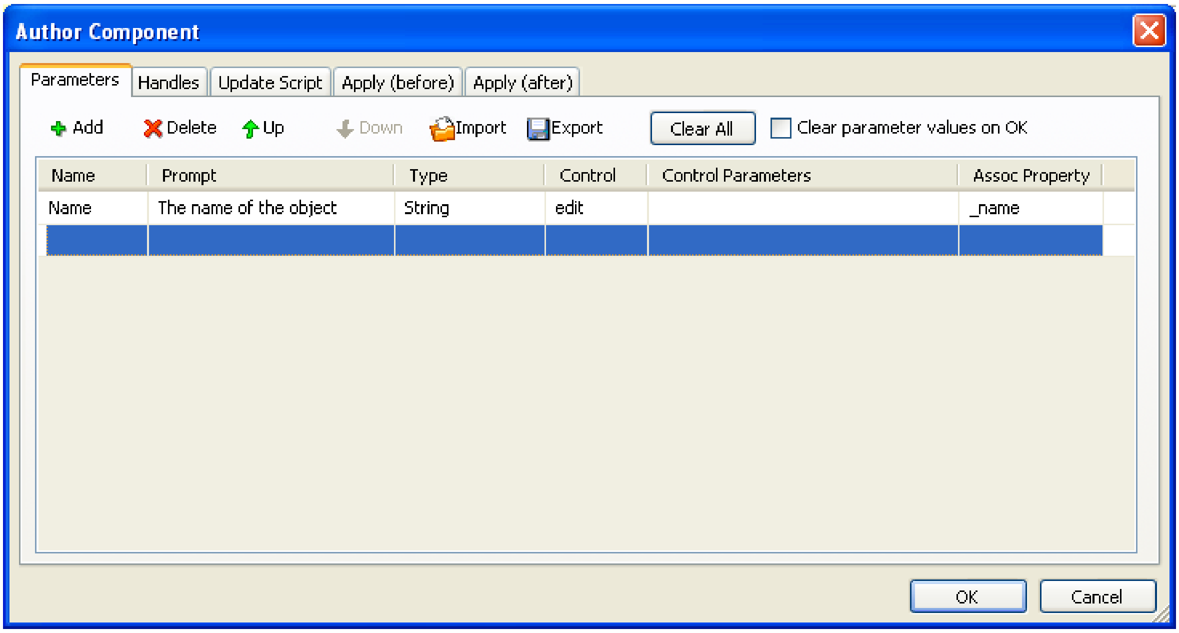
Note: The dialog may not be available if specific Object Attributes have been set. See Object Attributes for more information.



Name	Prompt	Type	Control	Control Parameters	Assoc Property
Name	The name of the object	String	edit		_name

4.5.4.1.1 Parameters Tab

This tab allows the user to define the parameters that are user modifiable.



□ Clear parameter values on OK

Name	Prompt	Type	Control	Control Parameters	Assoc Property
Name	The name of the object	String	edit		_name

Add **Add**
To add a new parameter.

Delete **Delete**
To delete the selected parameter.

Up **Up /** Down **Down**
To change the display location of the selected parameter.

Import
Reads in parameters from the selected XML file.

Export
Writes out all parameters to the chosen XML file.

Clear All
Clears all column entries in the Parameters tab.

Clear parameter values on OK
When checked and the OK button is pressed, the parameter values of the object will be cleared leaving the parameters as defined in this dialog with empty or blank values.

Name	Prompt	Type	Control	Control Parameters	Assoc Property

Name

This column allows the user to specify the name of the parameter. The supplied name will be displayed in the parameter column of the Parameters Panel as a single word to prompt the user for input. The value entered by the user in the Parameters Panel can be referenced in the script via the object ***parameters.<name>*** where ***<name>*** is the name entered in this column.

Note: The Name cannot contain any spaces.

Prompt
This is the longer multi-word prompt that appears in the bottom of the Parameters Panel when the user selects a specific parameter for data entry. This prompt is intended to assist the user by providing them additional information about the selected parameter.

Type
This column defines the type of data that the parameter holds. The available data types are:

String Any text characters. eg. "my string"
Number A positive or negative floating point number. eg. 3.1
Int A positive or negative integer. eg. -4
Boolean A logical value. Either true or false. A list box control containing true / false is implied.
Color A color value. A color control is implied.
Group Used to group multiple parameters.
Documentation Pseudo parameters that are used to document the component.

Control
This column defines the type of user input device that will be used to allow user input. The available types are:

edit User enters the value by typing values from the keyboard.
With Number and Int types, invalid characters are ignored. eg. if Type is Int, entry of 1.5a results in 1
Valid Types: String, Number and Integer.

list Allows entry of a value from a specific list of options. The available options are specified according to the Control Parameters. The items true and false are used if the Type is Boolean.
Valid Types: String, Number, Integer and Boolean types.

combo This is a combination of edit and list controls. Data can be entered directly as with the edit control or selected from a list like the list control.
Valid Types: String, Number and Integer

spin Up down arrows allow selection of higher or lower integer. Useful when specifying %. The value can also be typed in directly in a manner similar to the edit control.
Valid Types: Integer

color This is a color picker dialog. This can only be used with the Color type. It is set by and returns a 32bit integer value. When the integer is considered as a hex number, it has the format: 0xAARRGGBB where AA is the alpha value, RR is the red value, GG is the green value and BB is the blue value.
Note if you wish to initialize the associated parameter with a solid color (alpha = 100%) do a bitwise or with of the RRGGBB value with 0xFF000000.
eg. solid green is represented by 0xFF00FF00
Valid Types: Color

color+alpha This is a color picker dialog with the additional control that allows setting of the alpha value. It is set by and returns a 32bit integer value. When the integer is considered as a hex number, it has the format: 0xAARRGGBB where AA is the alpha value, RR is the red value, GG is the green value and BB is the blue value.
Valid Types: Color

group This is a group control. It can only be used with the Group type.
Valid Types: Group

Local file The corresponding Name value is displayed as a hyperlink. When the hyperlink is clicked, the file referenced in the Control Parameters area is opened according to the currently associated application.
Valid Types: Documentation

Static text The corresponding Name field is displayed without any parameter field in a read only fashion. This allows the Name and the accompanying Prompt to hold static text information about the component.
Valid Types: Documentation

URL The corresponding Name value is displayed as a hyperlink. When the hyperlink is clicked, the URL defined in the Control Parameters area is opened via the default browser.
Valid Types: Documentation

Control Parameters

This column defines additional parameters associated with the selected control. The parameters are entered as a semi colon (;) separated list.
eg. Param1;Param2;Param3

edit No parameters required

list The options that are displayed in the list box.
eg. option1;option2;option3

combo The options that are displayed in the list box.
eg. option1;option2;option3

spin The maximum and minimum values for the spin control as well as the optional step value.
eg. min;max;step
if omitted 0 to 100, step 1 is assumed.

color No parameters required

color+alpha No parameters required

group No parameters required

Local file The file that is opened when the hyperlink is clicked.
This file could represent a help file for the component. .txt, .chm or .rtf file formats could be used.
If the full path is NOT specified, the file location is relative to the position that the component was loaded from.
If the full path is specified, the file is opened according to that path.
The macro %EXEPATH% is provided to allow an absolute file reference to be derived from the current Max2 .exe location.
Macros for %TEMPLATES% %COMPONENTS% %EFFECTS% %TEST% and %INCLUDES% are also provided. Those references relate to the directories as defined via Tools | Preferences.

Static text No parameters required

URL The specified URL is opened when the hyperlink is clicked.

Assoc. parameter

This column can be used to bind a parameter defined in the table to a parameter associated with the object. Properties bound in this fashion do not require additional scripting via the Scripting tabs. eg. If "_alpha" is placed in this column, then the user modifiable parameter will modify the alpha of the component when the user sets the parameter value.
This field should be left blank for all of the **Documentation** types.

Examples

ID	Name	Prompt	Type	Control	Control Param.	Assoc parameter.

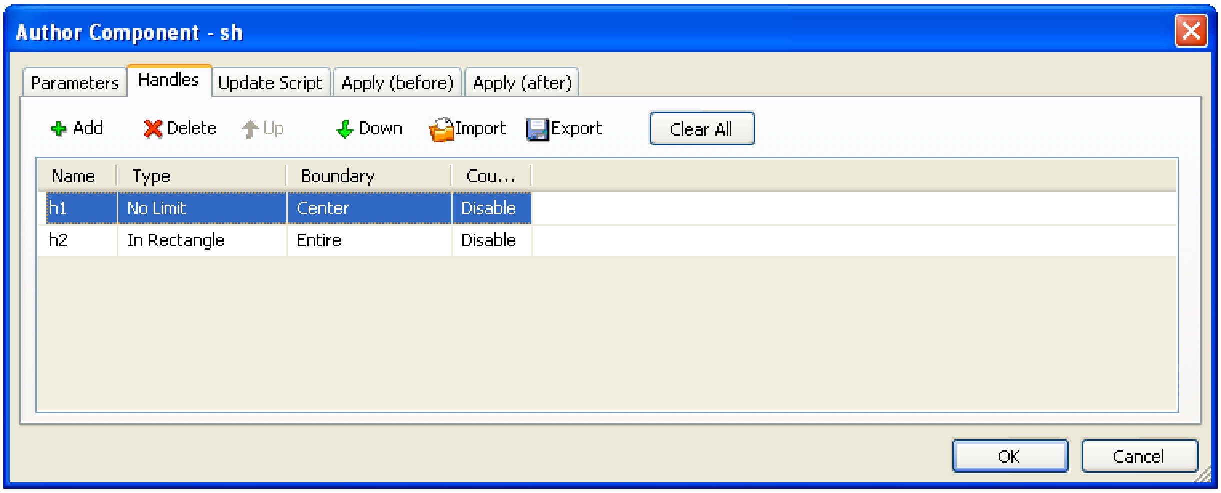
1	Name	Object Name	String	edit		_name
2	Alpha	Object Alpha (visibility)	Int	spin	50;100	_alpha
3	AlignLeft	Alignment of internal objects	Boolean	list		
4	BgndColor	Background Color	Color	color		
5	Style	Choose from one of the available styles	String	list	Big;Small; Medium	

Examples Description

1. This prompts the user for the object name. The name is a string that is entered via an edit control. It is associated with the _name parameter of the component. User updates of this field will cause the _name parameter of the component to be modified.
2. This prompts the user for the Alpha value. The value is a integer that is entered via a spin control. Minimum spin value is 50, maximum is 100. The parameter is associated with the component _alpha parameter.
3. This parameter defines left or right alignment of objects that are internal to the component. This alignment is done via scripting in the Scripting tabs.The value is a boolean (true / false) and it is selected (by default) via a list box.
4. This color parameter is selected (by default) via the color picker control. Color change is achieved via scripting within the onSelfEvent(load) event of the component. Color change can now be set via the Fill Object.
5. The style parameter is set from one of the available options: Big, Small or Medium. The use of the selected style is applied via the Scripting tabs.

4.5.4.1.2 Handles Tab

This tab allows the user to define handles that the user can drag to shape and configure the component. The handles are the same as those that appear in Autoshapes.



Name	Type	Boundary	Cou...
h1	No Limit	Center	Disable
h2	In Rectangle	Entire	Disable

Once a handle is defined, its current properties can be obtained via component scripting and the handles properties.

Add **Add**

To add a new handle.

Delete **Delete**

To delete the selected handle.

Up **Up /** Down **Down**

To change the display location of the selected handle.

Reads in handles from the selected XML file.

Writes out handles to the chosen XML file.

Clear All
Clears all column entries in the Handles tab.

Name
This column allows the user to specify the name of the handle.
Note: The Name cannot contain any spaces.

Type
This column defines the type of handle. The available types are:

No Limit The handle can be dragged anywhere inside and outside of the object.
In Rectangle The handle can be dragged anywhere within the bounding rectangle of the object.
In Ellipse The handle can be dragged anywhere within an ellipse that is sized according to the bounding rectangle
On Rectangle The handle can be dragged to any corner of the bounding rectangle.
On Ellipse The handle can be dragged anywhere on the edges of the ellipse that is sized to fit inside of the bounding rectangle
On Edge The handle can be dragged anywhere on the edges of the bounding rectangle
On Point The handle is fixed at the position defined by Boundary and cannot be moved. This is typically used with handles that are counters.

Boundary
This column defines the boundary that defines where the handle can be dragged. In the case of the type "On Point" this column defines the position of the handle.

The available values are shown in the table below:

Boundary Name	**Reference Point**	**Notes**
Top Left	Top left hand corner	
Top	Top center	
Top Right	Top right hand corner	
Left	Left center	
Center	Center	For the type **On Point**. The handle is placed in the center
Entire	Center	No Boundary is applied
Right	Right center	
Bottom Left	Bottom left hand corner	
Bottom	Bottom center	
Bottom Right	Bottom right hand corner	

Counter
This boolean value defines if the handle can be used to increment or decrement an internal count when clicked.
This type of handle is normally used to increase / decrease the number of points associated with a component. Counter type handles are generally of the type **On Point**.

See Also

See the handles section in Component Script Properties for a description of how to access the current handle position and settings from component script.

4.5.4.1.3 Scripting Tabs

The scripting tabs are provided so that the user can apply scripted changes to the object when:
1. The component object is modified on the stage (ie. Layout panel)
2. A component parameter is modified via the Parameters Panel

These scripts provide a way for the user to:
1. Initialize the parameters displayed in the Parameters Panel.
2. Initialize the Object with new parameters once the parameters in the Parameters Panel have been modified.

The Update Script Tab is used to enter script that is called when the component is modified. This script is also called when the component is first dragged onto the stage.

The Apply (Before) and Apply (After) Tabs allow the user to enter script that is called when parameters in the Parameters Panel have been modified. The script in the Apply (Before) Tab is called before the child objects are created. The Script in the Apply (After) Tab is called after the child objects have been created.

In most cases, either tab can be used. However in some circumstances it is necessary to use a specific tab.

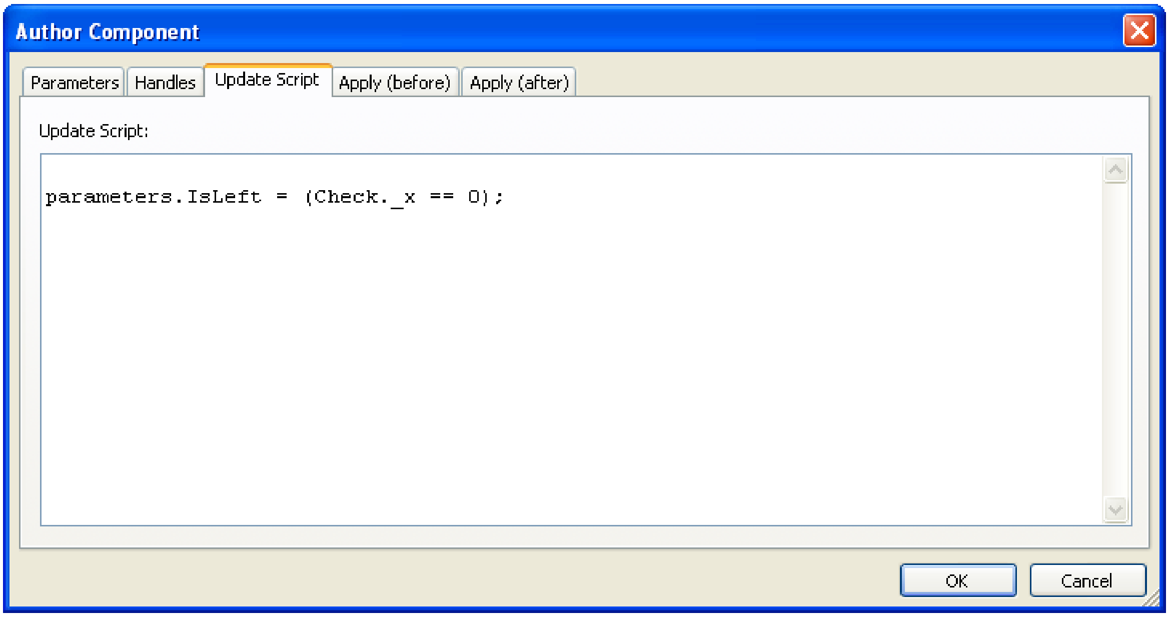
Scripting Attributes	**Apply (Before)**	**Apply (After)**
Script can access and alter properties/parameters in child objects	No	Yes
Script can dynamically create child objects	Yes	No
Child objects can refer to updated properties/parameters in the parent object	Yes	No

For all scripts, including script within the component, the values entered by the user in the Parameters Panel can be referenced in the script via the object ***parameters.<name>*** where ***<name>*** is the name entered in the **Name** column of the Properties Tab. The Parameters Object is the top level of the component. Child objects must reference this via **_parent.parameters.<*name*>**

Note:
- Script can be placed in both tabs if necessary
- Scripting and parameter names are case sensitive. Failure to match the case may result in errors

4.5.4.1.3.1 Update Script Tab

Script entered in this tab is used to initialize the Parameters Panel based on the current way the component is displayed on the stage.
Script within this tab is NOT intended to modify the component.



```
parameters.IsLeft = (Check._x == 0);
```

Example: If the component was to define a checkbox, two possibilities exist for its layout.
1. Checkbox appears the left of the text or
2. Checkbox appears to the right of the text.

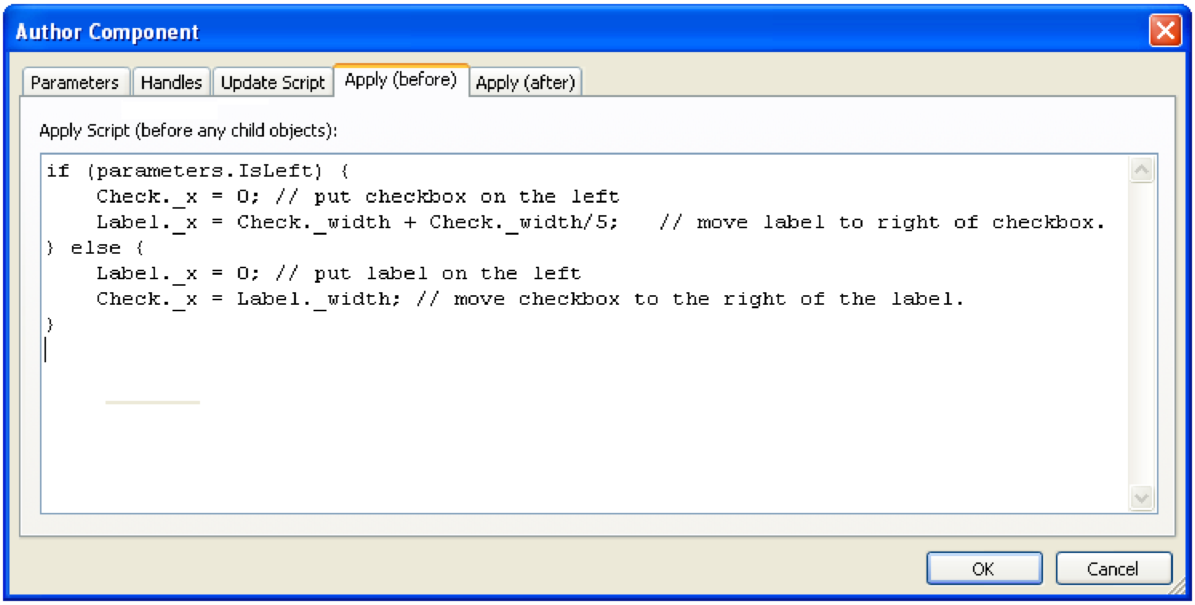
The IsLeft parameter (a boolean property) in the Parameters Panel could be initialized as follows by the Update Script:

```
parameters.IsLeft = (Check._x == 0);
```

This means that if the Checkbox has an offset of 0 from the component, it is positioned on the left hand side. Consequently parameters.IsLeft is set to an initial state of True if the checkbox is on the left hand side.

4.5.4.1.3.2 Apply (Before) Tab

Script entered in this tab is applied when the user updates the properties in the Parameters Panel. The script is run before any child objects are updated. Script entered in this tab can be used to apply changes to the component as a result of user changes to the settings in the Parameters Panel.



Apply Script (before any child objects):

```
if (parameters.IsLeft) {
    Check._x = 0; // put checkbox on the left
    Label._x = Check._width + Check._width/5;   // move label to right of checkbox.
} else {
    Label._x = 0; // put label on the left
    Check._x = Label._width; // move checkbox to the right of the label.
}
```

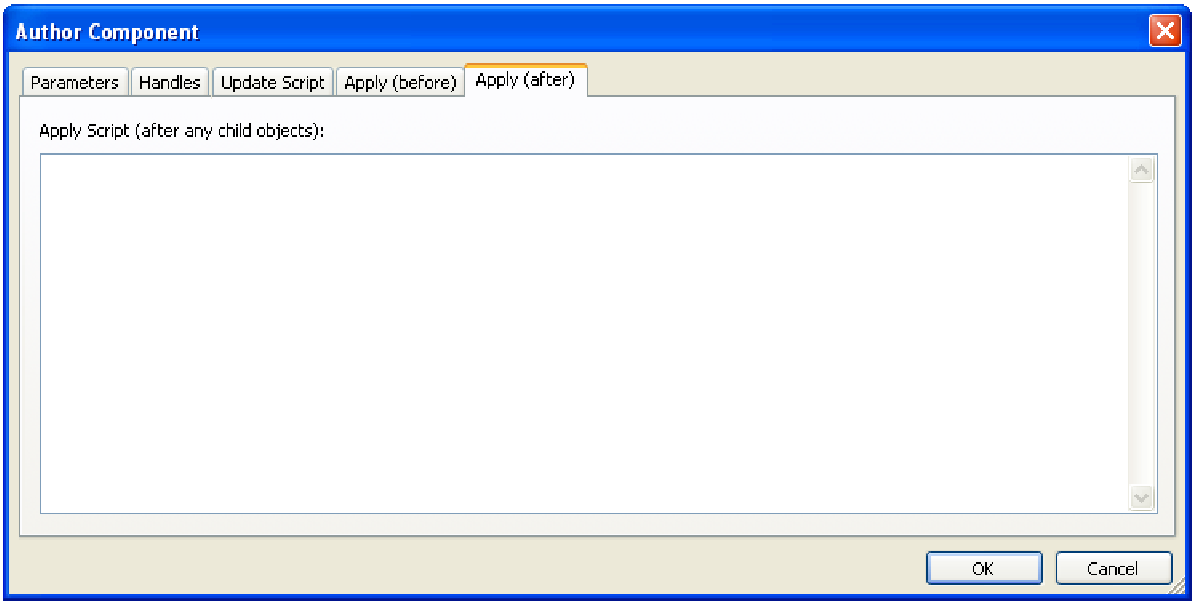
Example: If the component was to define a checkbox, two possibilities exist for its layout.
1. Checkbox appears the left of the text or
2. Checkbox appears to the right of the text.

The position of the check box and text must therefore be altered if the user was to modify the value of the IsLeft parameter.

```
if (parameters.IsLeft) {
    Check._x = 0; // put checkbox on the left
    Label._x = Check._width + Check._width/5;        // move label to right of checkbox.
} else {
    Label._x = 0; // put label on the left
    Check._x = Label._width; // move checkbox to the right of the label.
}
```

4.5.4.1.3.3 Apply (After) Tab

Script entered in this tab is applied when the user updates the properties in the Parameters Panel. The script is run after any child objects are updated. Script entered in this tab can be used to apply changes to the component as a result of user changes to the settings in the Parameters Panel.



Apply Script (after any child objects):

4.5.4.1.4 Component Scripting

Component scripting has some differences when compared to regular scripting.
1. The scripting that is applied in the Scripting Tabs can contain additional Properties that are not available in the normal scripting language.
2. It is not necessary to access the raw methods / properties of text and button objects via _text and _button properties.
3. Function declaration and calls are not supported.
4. Some scripting objects and commands are only available to component scipt.

1. Component Specific properties
See the section Component Script Properties for more information on these properties.

2. Access to Raw Methods / Properties
Text objects within a component can make use of the TextFormat() object to alter the default format according to settings in the parameters panel. Note that unlike regular scripting, use of the _text property is not required. See Script Object for more information about the _text property.

Example: This code snippet could be used in the Update Script Tab to load the current format values into the Parameters Panel.

```
var my_fmt = new TextFormat();
my_fmt = this.getTextFormat();
parameters.labelAlign = my_fmt.align;
parameters.labelBold  = my_fmt.bold;
parameters.labelColor  = my_fmt.color;
parameters.labelFont  = my_fmt.font;
parameters.labelFontSize = my_fmt.size;
parameters.labelItalic = my_fmt.italic;
parameters.labelSpaceLine = my_fmt.leading;
```

This code snippet could be used in the Apply (Before) Tab to update the text format after modification via the Parameters Panel.

```
var my_fmt = new TextFormat();
my_fmt = this.getTextFormat();

my_fmt.align   =      parameters.labelAlign;
my_fmt.bold    =      parameters.labelBold;
my_fmt.color   =      parameters.labelColor;
my_fmt.font    =      parameters.labelFont;
my_fmt.size    =      parameters.labelFontSize;
my_fmt.italic  =      parameters.labelItalic;
my_fmt.leading =      parameters.labelSpaceLine;

this.setTextFormat(my_fmt);
```

3. Declaration and use of function calls.
Currently it is not possible to define and call a user defined function.

4.5.4.1.4.1 Component Script Properties

The properties listed below are available within component scripting. Some properties are only available in Action Script (A), Component Script (C) or Both (B). Items that are read only are shown with **r**. eg. **Cr** means a read only component scripting property.
Where a property is available in action script, please view its description within the Script Reference Dictionary.

Property	**Avail.**	**Type**	**Brief Description**
_alpha	B	Num	See Script Reference section.
_height or height	Br	Num	See Script Reference section. This value is updated to be the overall height when the object is rotated, skewed, resized or rotated.
_heightOriginal or heightOriginal	C	Num	The original height of the object before any transform has been applied. Writing to this property will cause the object height to be resized.
_heightScaled or heightScaled	C	Num	The scaled height of the object. This is not adjusted by rotation or skewing transformations. Writing to this property will cause the object height to be rescaled. eg setting _heightScaled = 100 will cause the object to be rescaled so that the new height is 100.
_name or name	B	String	See Script Reference section.
_parent	B	String	See Script Reference section.
_root	B	String	See Script Reference section.
_rotation	B	Num	See Script Reference section.
_scene	Cr	String	Shows the name of the scene the component is placed in.
_skew	C	Num	Allows setting of the skew angle. Applies to the skew field in the Transform or Reshape panels.
_target	B	String	See Script Reference section.
_visible or visible	B	Boolean	See Script Reference section.
_width or width	Br	Num	See Script Reference section. This value is updated to be the overall width when the object is rotated, skewed, resized or rotated.
_widthOriginal or widthOriginal	C	Num	The original width of the object before any transform has been applied. Writing to this property will cause the object width to be resized.
_widthScaled or widthScaled	C	Num	The scaled width of the object. This is not adjusted by rotation or skewing transformations. Writing to this property will cause the object width to be rescaled. eg setting _widthScaled = 100 will cause the object to be rescaled so that the new width is 100.
_x	B	Num	See Script Reference section.
_xscale	B	Num	See Script Reference section.
_y	B	Num	See Script Reference section.
_yscale	B	Num	See Script Reference section.
body	C	String	Allows the code section of a script object to be read or modified. eg. *script.namedChildren["function myfunc(x)"].body = "{return x*x*

			+ 3;}"; will modify the script for the function myfunc() to be x*x + 3 This can be very useful where code needs to be conditionally inserted depending on options chosen in the parameters panel.
bottom	Cr	Num	Bottom of the bounding rectangle with respect to the reference point. See also bottom, left, right and top.
buttonTracking	Cr	String	Returns "menu" if trackAsMenu is true or "button" if trackAsMenu is false.
children	Cr	String[]	Returns an array of the immediate child objects. Each array item contains the name of a child object. See also namedChildren[]
concealChildren	C	Boolean	Allows setting / resetting of the Object Attribute
concealScript	C	Boolean	Allows setting / resetting of the Object Attribute
currentSelection	Cr	String	null or the name of the currently selected object within the current component. This corresponds to selection[0]
depth	Cr	Num	Shows the current depth of the object on the stage or within the containing movie or movie clip
effects	C	effects	Returns an effects object that contains an array of children. Each of the child objects allows the that are applied to the object.
elements			Synonym of children
elementType	Cr	String	Defines the type of object.
events	C	Object[]	Deprecated, Synonym of scripts.
excluded	C	Boolean	If set excludes the associated object from the exported .swf
expanded	C	Boolean	True if the object is expanded in the outline panel. Objects that have no children always return false. setting this property to false will close the object in the outline panel.
exposeAsChild	C	Boolean	Allows setting / resetting of the Object Attribute
frames	Cr	Num	Number of frames in the timeline of the object.
handles	Cr	handle	Allows reading of handle attributes.
hasEffects	Cr	Boolean	True if the associated object has effects.
hasEvents	Cr	Boolean	True if the associated object contains any scripting.
hasTimeline	Cr	Boolean	True if the associated object has it own timeline. This is normally true for Movie Clips.
hidden	C	Boolean	True if the associated object is hidden in the outline panel.
instance	Cr	Boolean	True if the associated object is an instance.
instanceType	Cr	String	Defines the type of object that the linked object is. One of "Movie Clip" "Group" "Shape" or "Button"
isAudio	Cr	Boolean	True if the object is a audio clip
isButton	Cr	Boolean	True if the object is a button
isButtonState	Cr	Boolean	True if the object is a button state
isDoc	Cr	Boolean	True if the object is a document
isEffect	Cr	Boolean	True if the object is an effect

		n	
isEvent	Cr	Boolean	True if the object is an event
isExternal	Cr	Boolean	True if the object is an external object
isGroup	Cr	Boolean	True if the object is a group
isMovieclip	Cr	Boolean	True if the object is a Movie Clip
isScene	Cr	Boolean	True if the object is a Scene
isShape	Cr	Boolean	True if the object is a Shape
isText	Cr	Boolean	True if the object is Text
isVideo	Cr	Boolean	True if the object is a Video
left	Cr	Num	Left position of the bounding rectangle with respect to the reference point. See also bottom, left, right and top.
libraryItem	Cr	Boolean	True if the item is a library item.
libraryItemName	Cr	String	Name of the library item.
linkageClassName	Cr	String	Corresponds to the **Class name** setting in the Asset tab.
linkageExport	Cr	Boolean	Corresponds to the **Enable asset** tickbox setting in the Asset tab.
linkageExportForAS			Synonym of linkageExport
linkageExportForRS			Synonym of linkageExport
linkageExportFrame	Cr	Num	Corresponds to the **Define in frame** setting in the Asset tab.
linkageExportName	Cr	String	Corresponds to the **Asset name** setting in the Asset tab.
linkageIdentifier			Synonym of linkageExportName
linkageImportForRS	Cr	Boolean	Corresponds to the **Enable import** checkbox setting in the Asset tab.
linkageImportName	Cr	String	Corresponds to the **Import name** setting in the Asset tab.
linkageURL	Cr	String	Corresponds to the **From SWF file** setting in the Asset tab.
locked	C	Boolean	True if the associated object is locked in the object outline panel.
makeHotspot	C	Boolean	If set to true, it make the object invisible but the area can be used as a "hotspot" to define mouse events.
matrix	Cr	Num[]	The transformation matrix expressed as a,b,tx,c,d,ty
namedChildren	C	Object	Allows access to the properties of child objects via: this.namedChildren[*varname*].xxx eg. if mc has a child object sh1 then mc.namedChildren["sh1"]._name would be the name of the child object. See also children[]
parameters or properties	C	various	Allows writing / reading the parameters that are defined via the parameters tab.
preloadMode	C	Num	Returns / Sets the type of preload mode for this object. The following values apply: 0:Disabled 1:Before Scene 2:Before Movie 3:At Preload frame 4:Scene Default

properties	C	various	Synonym of parameters.
readonlyChildren	C	Boolean	Allows setting / resetting of the Object Attribute
readonlyParameters	C	Boolean	Allows setting / resetting of the Object Attribute
readonlyProperties	C	Boolean	Allows setting / resetting of the Object Attribute
readonlyScript	C	Boolean	Allows setting / resetting of the Object Attribute
referencePosition	C	String	Sets the reference point of the component via script.
referenceX	C	Num	Defines the X position of the reference point if referencePosition is custom.
referenceY	C	Num	Defines the Y position of the reference point if referencePosition is custom.
right	Cr	Num	Right position of the bounding rectangle with respect to the reference point. See also bottom, left, right and top.
script	Cr	String	Shows the script associated with the object.

scripts	C	object[]	Returns an events object that contains an array of children. The _name property of the child shows the event name.
			eg. the following script in the update tab
			`trace("t1:" add scripts.children[0]._name);`
			`trace("t2:" add scripts.children[1]._name);`
			will display
			`t1:onSelfEvent (press)`
			`t2:onSelfEvent (release)`
			in the debug window if the object contains onSelfEvent press and release events.

selected	Cr	Boolean	True if the object is currently selected
selection	Cr	String[]	A string array of selected objects. This allows multiple selections to be processed. currentSelection corresponds to selection[0]
stopAtEnd	C	Boolean	Sets, resets and shows the status of the "Stop playing at end" movie clip property.
targetable	C	Boolean	Sets, resets and shows the status of the target checkbox in objects. Note that object must be named to enable the setting of the target checkbox.
top	Cr	Num	Top of the bounding rectangle with respect to the reference point. See also bottom, left, right and top.
trackAsMenu	C	Boolean	Sets, Resets or shows the status of the Track as Menu option that is available on the Group, Button and Movie Clip object property panels.
transformPosition	C	String	Allows setting of the transform position.
transformX	C	Num	Sets / returns the X offset of the transformation point with respect to the center of the object.
transformY	C	Num	Sets / returns the Y offset of the transformation point with respect to the center of the object.
useBoundsForHitRectangleWhenUsedInButton	Cr	Boolean	If true, the defined bounds will be used as the hit rectangle if the object is used as a button.

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to hide / expose the children objects of a component. It is assumed that the Parameter ExposeChildren exists as a boolean parameter in the Parameters tab.

```
concealChildren = !parameters.ExposeChildren;
```

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to hide / expose the script of a component. It is assumed that the Parameter ExposeScript exists as a boolean parameter in the Parameters tab.

```
concealScript = !parameters.ExposeScript;
```

This read only string property can be associated with any Movie Clip or object within a component. When an object within the current component is selected its name can be obtained from this property.

Example:
A simple component "mc1" contains a shape named "sh1". The following script is placed in the Update tab of mc1:

```
trace("currentSelection:" add currentSelection);
```

If mc1 is selected, the following trace statement is shown:

```
currentSelection:null
```

If the shape sh1 is slected, the following trace statement is shown:

```
currentSelection:Scene_1.mc1.sh1
```

This object contains an array of children. Each child can contain the standard Component Script Properties and will also contain the following effect specific properties:

Property	**Type**	**Comments**
complex	Boolean, R	True if this event manipulates individual elements within the object in a complex manner.
continueFromPrevious	Boolean, R	True if this event continues from the previous event
duration	Num, RW	Duration of the event in frames
endFrame	Num, RW	End frame. Zero based.
orientOverride	Boolean, R	
orientToPath	Boolean, R	
resetAtStart	Boolean, R	
smoothMotionPathAtKeyframe	Boolean, R	
startFrame	Num, RW	Start Frame. First frame is 0

The event properties are referenced via .events.children[n].*property*

Example:
A movie clip contains a shape sh1 which has the Drop in and Bounce effect applied.
The following code in the Update tab produces the debug trace shown below and moves the start of the effect to Frame 10 (shown as frame 11 in the timeline as the first frame of the timeline is frame 1).

```
trace(sh1.frames);
trace(sh1.effects.children[0]._name);
trace(sh1.effects.children[0].duration);
trace(sh1.effects.children[0].startFrame);
sh1.effects.children[0].startFrame = 10;
```

30
Drop in and bounce
30

0

When the script is run on a second occasion the debug will be:
40
Drop in and bounce
30
10

This read only string property can be associated with any Movie Clip or object within a component. It indicates the type of object. The value will be from the following list:
soundtrack
button
external media
shape
scene
soundtrack
text
video
movie clip
group
button state
instance - in this case the instanceType defines the type of the master object.

If the object is an instance, then elementType will be be "instance" and instanceType will represent the type of object that the library item is.
If the object is not an instance then elementType will be the type of the object and the instanceType will be undefined.

Example:
A simple component "mc1" contains a shape named "sh1". The following script:

```
trace("elementType:" add sh1.elementType);
```

Will display "elementType:shape" in the debug window.

Note: the is... properties such as isShape can be used to determine if an object is of a specific type.

This boolean property can be associated with any Movie Clip within a component. Setting this property to true will prevent the item from being exported to the final movie. This property can be used to allow selection (or exclusion) of various sub items within the component from the final Movie.

Example: A component contains both a rectangle and a circle. Both of these items are optional (user selectable) in the final component.
If the items are named "rect" and "circle" then the Properties Tab may contains items such as:
ShowCircle (boolean)
ShowRectangle (boolean).

The corresponding Update Script Tab scripting would be:

```
parameters.ShowRectangle = !rect.excluded;
parameters.ShowCircle = !circle.excluded;
```

The corresponding Apply (Before) Tab scripting would be:

```
rect.excluded = !parameters.ShowRectangle;
rect._visible = parameters.ShowRectangle;
circle.excluded = !parameters.ShowCircle;
circle._visible = parameters.ShowCircle;
```

Note: Changes to the _visible property are necessary to hide the excluded items from the stage when they are de-selected.

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to hide / expose a specific child object. This script is only of use if one of the parent objects has the concealChildren object attribute set.

```
shape1.exposeAsChild = true;
```

This object allows access to some general handle properties as well as access to the individual handles that are defined in the handles tab.

The handles object allows access to the global properties as well as access to the individual handle objects.

Property	**Type**	**Comments**
changed	Boolean	Set to true if a handle has been changed.
changedName	String	Name of the handle that has changed.
name	handle Object	*name* corresponds to the name of one of the handles defined in the handles tab.The properties of the named handle can then be viewed via the handle object.

Each named handle has the following properties:

Property	**Type**	**Comments**
angle	Number	Angle in degrees from the center point. 0 = to the right, +90 = below +/-180 = to the left -90 = above. Note that the angle is calculated from the current x, y position and represents the angle before object scaling is applied. Changing the xScale, yScale of the object may appear to alter the angle, but the reported angle will remain the same.
angleFromOutside	Number	Angle in degrees with respect to the outside reference point
count	Number	Only available when the handle has the counter field enabled. This property returns the current count. The count is increased when the handle is selected and the right mouse button is clicked. It is decreased when the left mouse button is used.
distance	Number	Distance from the center point expressed in pixels
distanceFromOutside	Number	Distance from the outside reference point expressed in pixels
distancePercent	Number	Distance from the center point expressed as a % This is based on sqrt(xPercent^2 + yPercent^2) see Pythagorean Theorem
distancePercentFromOutside	Number	Distance from the outside reference point expressed as a % This is based on sqrt(xPercentFromOutside^2 + yPercentFromOutside^2) see Pythagorean Theorem
x, y	Number	x and y position of the handle with respect to the center point. Note that the reported position does not take into account the current scaling of the object and relates to the original width and height.
xPercent, yPercent	Number	x and y position expressed as percentage values with respect to _width and _height.

xFromOutside, yFromOutside	Number	x and y position from the outside reference point expressed in pixels. Note that the reported position does not take into account the current scaling of the object and relates to the original width and height.
xPercentFromOutside, yPercentFromOutside	Number	x and y position from the outside reference point expressed as percentage values with respect to _width and _height

The **FromOutside** reference point is the point implied by the chosen boundary.

Boundary	Outside reference point
Top Left	Top Left
Top	Top Center
Top Right	Top Right
Left	Left Center
Center	Center
Entire	Center
Right	Right Center
Bottom Left	Bottom Left
Bottom	Bottom Center
Bottom Right	Bottom Right

Example: If a handle h1 is defined in the handles tab then

```
handles.h1.x
handles.h1.y
```

represent the x and y co-ordinates of the handle h1 with respect to the center of the object.

If the handle is has the Counter Enabled property set then

```
handles.h1.count
```

returns the current count associated with the handle.

Script in the Update Tab will be executed each time a handle is moved.

See Also:
Simple Trigonometry for a description of conversion between x,y and angle / distance.
This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to set / reset the read only property of child objects.

```
readonlyChildren = parameters.ChildrenReadOnly;
```

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to make a specific child object have read only parameters.

```
shape1.readonlyParameters = true;
```

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to make a specific child object have read only properties.

```
shape1.readonlyProperties = true;
```

This boolean property can be associated with any Movie Clip or object within a component. Setting this property to true will set the corresponding Object Attribute.

Example: Use the following script in the Apply (before) tab to set a specific child object's script to read only.

```
shape1.readonlyScript = true;
```

The properties **transformPosition** and **referencePosition** allow the user to set a specific transform and reference point or view the current transform and reference points via component script.

The possible values are:

transformPosition / referencePosition	**Description**
"topLeft"	Top Left
"top"	Top Center
"topRight"	Top Right
"left"	Center Left
"center"	Center
"right"	Center Right
"bottomLeft"	Bottom Left
"bottom"	Bottom Center
"bottomRight"	Bottom Right
"custom"	Custom position - Set the position according to transformX, transformY for transformPosition. Set the position according to referenceX, referenceY for referencePosition. **Note:** The custom positions are in terms of pixels from the center of the object.

Example: A component needs to have its reference point set to top left to work properly. Add the following script to the Update Script Tab:

```
referencePosition = "topLeft";
```

If a specific reference point (10, 20) was required then use the following script in the component script:

```
referenceX = 10;
referenceY = 20;
referencePosition = "custom";
```

Note: In early versions of max2, **referenceMode** was supported. This has been discontinued and replaced with **referencePosition**.
Properties that allow specification of the X and Y parts of a custom reference point.
This property is only used if referencePosition is set to "custom".
For an example see transform and reference points.

Properties that allow specification of the X and Y parts of a custom transform point.
This property is only used if transformPosition is set to "custom".

4.5.4.1.4.2 Component Script Methods

The following methods apply to component scripting only.

Selection Methods

Method	**Description**
addToSelection()	object.addToSelection() or addToSelection(object) or addToSelection("objectname") Any of the above formats are valid. This method will add the object to the current list of selected objects.

	See Also the property selected.
selectAll()	Adds all objects in the current movie clip to the selection list. Note that this will expand the object in the outline panel if the object contains child elements. To re-close the object finish your script with selectNone() then set the expanded property to false which will re-close the object.
select()	object.select() or select(object) or select("objectname") Deselects any currently selected objects and selects the referenced object only.
deselect()	object.deselect() or deselect(object) or deselect("objectname") Deselects the referenced object from the selection list.
selectNone()	Deselect all items in the selection list.
emptySelection()	Synonym of selectNone()
toggleInSelection()	object.toggleInSelection() or toggleInSelection(object) or toggleInSelection ("objectname") Toggles the referenced object's selection state.
getSelectionRectangle (r)	Returns the bounds of the selection rectangle in r. r is an object with left, right, top, bottom members.

Deletion Methods

Method	Description
deleteThis()	object.deleteThis() or deleteThis(object) or deleteThis("objectname") Deletes the referenced object.
deleteSelection()	Deletes all selected objects.

Add Object Methods

Method	Description
addItem(pos,"myLibraryItem")	Creates a new item at p based on the named library item. p is an object representing the x,y position. p = {x:100,y:150};
addNewRectangle(r)	Creates a new rectangle with the default color and line style. r is an object with left, right, top, bottom members.

Reshape / Resize Methods

The Methods listed here allow the component to be reshaped. Note that this is different to scaling.

Method	Description
rotateVertices(deg)	Rotates the vertices the specified number of degrees. _width and _height values are updated accordingly.
skewVertices(deg)	Skews the vertices the specified number of degrees. _width and _height values are updated accordingly.
resize(scale)	adjusts the _width and _height so that the object is resized uniformly by the specified percent.
resize(xscale,yscale)	adjusts the _width and _height according to the specified xscale and yscale parameters.

Library Methods

Method	Description
object. addToLibraryAsCopy()	Adds the object / shape to the library by copying it to the library.
object. addToLibraryAsLink()	Adds the object / shape to the library then replaces the current object with a link to the object in the library.

Ordering / Grouping

Method	Description
arrangeSelection(moveTo)	Changes the ordering level of the selected object(s). The parameter **moveTo** can be one of: "back" - move selection to the back "backward" - move the selection one level back "forward" - move the selection one level forward

	"front" - move the selection to the front.
arrange(moveTo)	object.arrange() Changes the ordering of the referenced object. Parameters are the same as for arrangeSelecton()
groupSelection(grouptype [, singlestyle])	Groups the selected objects according to the parameters. **grouptype** - how to group the selection. One of: "group"|"movieclip"|"button"|"shape" **singlestyle** - set to true if overlapped regions with the same fill style are to appear empty.
group(grouptype [,singlestyle])	object.group() Groups the referenced object according to the parameters. Parameters are the same as groupSelection()
convertSelection(convertto [,effectsStaysWithOriginal [,scriptStaysWithOriginal]])	Converts the selection according to the parameters. **convertto** - what to convert to, one of: "movieclip"|"buttons" **effectsStaysWithOriginal** - optional. If true, effects stay with the original object. If false, effect is moved to the converted object. Default: true **scriptStaysWithOriginal** - optional. If true, script stays with the original object. If false, script if moved to the converted object. Default: true
convert(convertto [,effectsStaysWithOriginal [,scriptStaysWithOriginal]])	object.convert() Converts the object according to the parameters. Parameters are the same as convertSelection()
breakIntoSelection(newobjecttype)	Breaks the selected objects according to the parameter "newobjecttype". newobjecttype - one of "shapes" or "letters" **shapes**: the object(s) are converted into grouped object(s) containing a shape object for each item. If one of the original objects was text, then each letter is converted into a shape object. **letters**: the object(s) are converted into grouped object(s). If the original object is a text object then each letter is converted into a individual text object.
breakInto(newobjecttype)	object.breakInto() Breaks the referenced object according to the parameter "newobjecttype". newobjecttype - one of "shapes" or "letters" Parameters are the same as breakIntoSelection()
unGroupSelection()	un groups all selected objects
unGroup()	object.unGroup() - un groups the referenced object.
alignSelection(alignment, withRespectTo)	Aligns the selected object(s) according to the parameters. **alignment** can be one of: "left" - Align to the left "right" - Align to the right "top" - Align to the top "bottom" - Align to the bottom "horizontal center" - Align in the Horizontal plane to the center "vertical center" - Align in the Vertical plane to the center All of the above alignments align according to the appropriate edges. "horizontal reference" - Aligns the reference points in the Horizontal plane "vertical reference" - Aligns the reference points in the Vertical plane **withRespectTo** can be one of: "all" - base alignment on all shapes "last" - align according to the last shape "parent" - align with the parent

	"stage" - align with the stage
align(alignment,withRespectTo)	object.align() Aligns the referenced object according to the parameters. The parameters are the same as described for alignSelection()

4.5.4.1.4.3 Objects and Commands

Shape Methods, Properties and Objects allow shapes to be manipulated via component script. Shapes have implicit Fill and Stroke objects. The parameters associated with those objects can be manipulated within the component script to alter the fill and border of the associated object.

The Methods and Properties that are available for shape manipulation are described below.

Note: These Methods, Properties and Objects are only available within component scripting.

contourCount : int - returns number of contours (sub-shapes within a shape)

Example

```
trace("Count: " add shape1.contourCount); // show number of contours in debug panel
```

fillCount(c : int) : int - returns the number of fill styles for contour number c (zero-based), if no c supplied, assumes first contour

Example

```
var contours = shape1.contourCount;
trace("Fill Count: " add shape1.fillCount(contours-1)); // show number of fills in the last contour in debug pane
```

strokeCount(c : int) : int - returns the number of stroke styles for contour number c (zero-based), if no c supplied, assumes first contour

Example

```
var contours = shape1.contourCount;
trace("Fill Count: " add shape1.strokeCount(contours-1)); // show number of strokes in the last contour in debug
```

getFill(c : int, f : int) : Fill - returns the Fill object for the specified fill of contour number c (zero-based), if no c supplied assumes first contour, if not f supplied assumes first fill.

Example

```
s1Fill = shape1.getFill(1,0); // s1Fill contains the first fill of the 2nd contour in shape 1.
```

getStroke(c : int, s : int) : Stroke - returns the Stroke object for the specified stroke of contour number c

(zero-based), if no c supplied assumes first contour, if no s supplied assumes first stroke.
Example

```
s1Stroke = shape1.getStroke(1,0); // s1Stroke contains the stroke for the 2nd contour, 1st stroke of shape1.
```

---

setFill(fill : Fill, c : int, f : int) - sets the specified fill of contour number c (zero-based), if no c supplied assumes first contour, if not f supplied assumes first fill.
Example

```
shape1.setFill(s1Fill, 1, 0); // sets contour 1, fill 0 of shape1 with the supplied fill s1Fill.
```

---

setStroke(stroke : Stroke, c : int, s : int) - sets the specified stroke of contour number c (zero-based), if no c supplied assumes first contour, if not s supplied assumes first stroke.
Example

```
shape1.setStroke(s1Stroke, 1, 0); // sets contour 1 (2nd contour), stroke 0 of shape1 with the supplied stroke s
```

---

fit() - refit the fill to the shape. This will cause re-scaling of the Fill object matrix to fit the size of the object.
Example

```
shape1.fit();
```

The following example shows how Fill and Stroke parameters can be applied to a simple object within a component.
The parameters for the component are shown in the xml excerpt below:

```
<parameters>
        <parameter type="Color" control="color+alpha" name="shapecolor"/>
        <parameter controlparams="noFill;solid;linearGradient;radialGradient;tiledImage;singleImage" type="String
        <parameter prompt="Degrees of gradient rotation" type="Number" control="edit" name="gradientangle"/>
        <parameter prompt="offset of center in x direction" type="Number" control="edit" name="gradientxoffset"/>
        <parameter prompt="offset of center in y direction" type="Number" control="edit" name="gradientyoffset"/>
        <parameter type="Color" control="color+alpha" name="linecolor"/>
        <parameter controlparams="-1;0;0.5;1;2;3;5;10" type="Number" control="list" name="linethickness"/>
        <parameter controlparams="noStroke;solid;dash;gDash;dot;dashDot;dashDotDot;alternative;custom" type="Stri
        <parameter prompt="enter combination of . and - Only applies if custom style is selected." type="String"
</parameters>
```

The following code is from the Apply (before) tab.

```
style.style = parameters.fillstyle;
this.setFill(style); // set the new fill style, now change the other items.
trace("CC " add this.contourCount);

this.fit();
style = this.getFill();
style.color = parameters.shapecolor;
style.colorArray = [parameters.shapecolor, parameters.shapecolor & 0xFFFFFF];
style.posArray = [0,255];

// setup rotation ratios and save current abcd values.
var cosrot = cosdeg(parameters.gradientangle);
var sinrot = sindeg(parameters.gradientangle);
var a = style.matrix.a;
var b = style.matrix.b;
var c = style.matrix.c;
var d = style.matrix.d;

var scale = (style.style == "tiledImage") ? 0.5:1;

// apply rotation. rotation is assumed to apply before translation.
style.matrix.tx = parameters.gradientxoffset;
style.matrix.ty = parameters.gradientyoffset;
style.matrix.a = (a*cosrot-b*sinrot)*scale;
style.matrix.b = (a*sinrot+b*cosrot)*scale;
style.matrix.c = (c*cosrot-d*sinrot)*scale;
style.matrix.d = (c*sinrot+d*cosrot)*scale;
```

```
this.setFill(style); // apply the new fillstyle
// display current matrix
style = this.getFill();
trace("style=" add style.style add " tx=" add style.matrix.tx add " ty=" add style.matrix.ty);
trace("a=" add style.matrix.a add " b=" add style.matrix.b);
trace("c=" add style.matrix.c add " d=" add style.matrix.d);


// now do line style stuff.
lstyle = this.getStroke();
lstyle.thickness = parameters.linethickness;
lstyle.color  = parameters.linecolor;
lstyle.style = parameters.linestyle;
lstyle.custom = parameters.customstyle;
this.setStroke(lstyle);
```

details of the matrix rotation and tx, ty, a, b, c, d can be found here

[****]

A Fill object has members:
style : String - "noFill","solid","linearGradient","radialGradient","tiledImage","singleImage"
color : int - color value (for solid fill)
colorArray : Array - array of int colors (for gradient fills)
posArray : Array - array of int positions 0..255 (for gradient fills)
matrix : Object - matrix with number values tx,ty,a,b,c,d
libraryItem : Object - reference to library item if linked
libraryItemName : String - name of library item if linked
A Stroke object has members:
style : String - "noStroke","solid","dash","gDash","dot","dashDot","dashDotDot","alternative","custom"
custom : String - made up of "-" and "." to give stroke pattern (only for "custom" style). "-" indicates a line, "." indicates a space. Note: the string should start with the "-" symbol.
color : int - color value
thickness : int - thickness
Auto Shapes inherit the standard Shape Methods and Properties. In addition they have methods and properties that allows the configuration points for the object to be reviewed and manipulated.

The Methods and Properties that are available for auto shape manipulation are described below.

Note: These Methods, Properties and Objects are only available within component scripting.

---

autoshapeName: String, Read Only - Returns the type of auto shape.
Currently it will be one of the following names:
Arrow
ArrowNotched
ArrowTwoWay
ButtonBevelled
ButtonRounded
Cube
Heart
Polygon
RectRounded
Star
Example
rr is a Rounded Rectangle auto shape in the current movie clip.
`trace(rr.autoshapeName);` will write "RectRounded" to the debug panel.

---

autoshapeHeight, autoshapeWidth: Numeric - returns / adjusts the height and width of the rectangle that bounds the auto shape.

Example

rr is a Rounded Rectangle auto shape in the current movie clip. This code could appear in the component Update Script panel:

```
trace(rr.autoshapeHeight);    // returns height of bounding rectangle in pixels
trace(rr.autoshapeWidth);     // returns width of bounding rectangle in pixels
```

---

getAutoshapePoints(): Method - Returns an array of points. The points represent the location of each of the green position handles. The exact meaning of the points depends on the type of the autoshape.

Example

rr is a Rounded Rectangle auto shape in the current movie clip.

```
p = rr.getAutoshapePoints();
for (var i in p) {
    trace(i add ": " add p[i].x add ", " add p[i].y); // list all of the configuration points for a rounded recta
}
```

---

setAutoshapePoints(): Method - Uses the supplied array to define the configuration points of the auto shape.

Example

rr is a Rounded Rectangle auto shape in the current movie clip. This script could appear in the Apply Before component sc

```
p = rr.getAutoshapePoints();  // get the currently defined points
// rounded rectangle only has one position handle
p[0].x = parameters.rx;       // take x position from parameters panel. 0,0 makes rectangle ellipsoid.
p[0].y = parameters.ry;
rr.setAutoshapePoints(p);     // set new guide point.
```

The getAutoshapePoints() and setAutoshapePoints() methods get and set the auto shape guide points. The use of the guide points is different depending on the type of auto shape. The use and setting of the guide points is described in the Auto Shape Guide Points section.

---

autoshapeCount: Numeric - returns / sets the first count number for the associated auto shape. Currently only Polygon and Star support this property which represents the number of sides (Polygon) or points (Star).

Note: autoshapeCount is an alias for the methods getAutoshapeCounts() and setAutoshapeCounts() where only the first element in the array is considered. Currently no auto shape requires more than one count value. However, this may change when more complex auto shapes are added.

Example

star is a Star auto shape in the current movie clip. This script could appear in the Apply Before component script tab to allow setting of the number of points.

```
star.autoshapeCount = parameters.points;     // set the number of points.
```

---

getAutoshapeCounts(): Method - Returns an array of counts. These counts represent the number of points or sides in an auto shape. Currently only Polygon and Star support this method. Other shapes will get a return value of undefined. Currently the array is length 1, however, this may change when more complex auto shapes are added.

autoshapeCount is an alias that refers to getAutoshapeCounts()[0].

Example

star is a Star auto shape in the current movie clip. This script could appear in the Update Script component script tab to allow review of the number of points.

```
c = star.getAutoshapeCounts();
trace("This star has " add c[0] add " points");
```

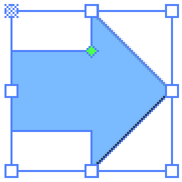
setAutoshapeCounts(): Method - Sets the auto shape point counts according to the contents of the supplied array.

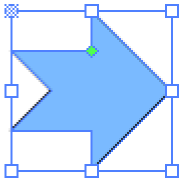
Example

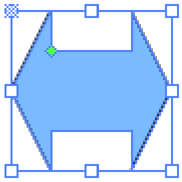
star is a Star auto shape in the current movie clip. This script could appear in the Apply Before component script tab to allow setting of the number of points.

```
c = star.getAutoshapeCounts();
c[0] = parameters.points;     // set the number of points in the 1st array element.
star.setAutoshapeCounts(c);
```

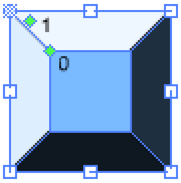
In the following tables w = width, h = height of the surrounding rectangle.
The points are specified with respect to the origin (0,0) which is in the center of the shape.
For any shape, top left hand corner is (-w/2, -h/2), bottom right hand corner is (w/2, h/2).

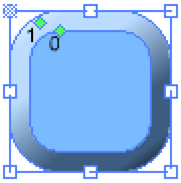
Common name	Arrow
autoshapeName	Arrow
Example	
Guide Points	1
p[0].x values	-w/2 to w/2
p[0].y values	-h/2 to 0

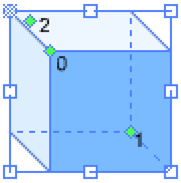
Common name	Notched Arrow
autoshapeName	ArrowNotched
Example	
Guide Points	1
p[0].x values	-w/2 to w/2
p[0].y values	-h/2 to 0

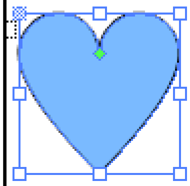
Common name	Two-way Arrow
autoshapeName	ArrowTwoWay
Example	
Guide Points	1
p[0].x values	-w/2 to 0
p[0].y values	-h/2 to 0

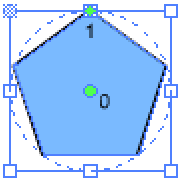
Common name	Bevelled Button
autoshapeName	ButtonBevelled

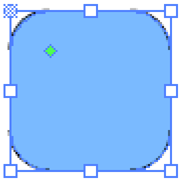
Example	
Guide Points	2
p[0].x values	-w/2 to 0
p[0].y values	-h/2 to 0
p[1].x values	-w/2 to w/2
p[1].y values	-h/2 to h/2

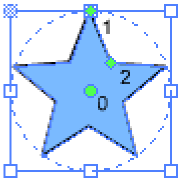
Common name	Rounded Button
autoshapeName	ButtonRounded
Example	
Guide Points	2
p[0].x values	-w/2 to -W0 where W0=(h>w)?0:w/2-h/2
p[0].y values	-h/2 to -H0 where H0=(w>h)?0:h/2-w/2
p[1].x values	-w/2 to w/2
p[1].y values	-h/2 to h/2

Common name	3D Cube
autoshapeName	Cube
Example	
Guide Points	1
p[0].x values	-w/2 to w/2
p[0].y values	-h/2 to h/2
p[1].x values	-w/2 to w/2
p[1].y values	-h/2 to h/2
p[2].x values	-w/2 to w/2
p[2].y values	-h/2 to h/2

Common name	Heart
autoshapeName	Heart
Example	
Guide Points	1
p[0].x values	-w/2 to w/2
p[0].y values	-h/2 to 0

Common name	Polygon
autoshapeName	Polygon
Example	
Guide Points	2
p[0].x, p[0].y	fixed at (0,0) Use left / right click to increase / decrease number of sides. The number of sides can be altered in component script via the **autoshapeCount** property.
p[1].x, p[1].y	Any values with the exception of 0,0 can be quoted. The angle with respect to the origin that the specified point makes defines the rotation angle. eg. (0,-2) or (0,-400) would both specify handle 1 as being above the origin in the position shown in the example. (2,0) or (100,0) would specify handle 1 to be the middle RHS.

Common name	Rounded Rect
autoshapeName	RectRounded
Example	
Guide Points	1
p[0].x values	-w/2 to 0
p[0].y values	-h/2 to 0

Common name	Star
autoshapeName	Star
Example	
Guide Points	3
p[0].x, p[0].y	fixed at (0,0) Use left / right click to increase / decrease number of points. The number of points can be altered in component script via the **autoshapeCount** property.
p[1].x, p[1].y	Any values with the exception of 0,0 can be quoted. The angle with respect to the origin that the specified point makes defines the rotation angle. eg. (0,-2) or (0,-400) would both specify handle 1 as being above the origin in the position shown in the example. (2,0) or (100,0) would specify handle 1 to be the middle RHS. **Note:** When the handle is dragged with the mouse, handle 2 is also rotated. This does NOT happen when

	the handle is moved using script. ie. handle 2 remains in the same position. If desired, p[2] should be recalculated using trigonometry to maintain the overall shape of the star.
p[2].x values	-w/2 to w/2
p[2].y values	-h/2 to h/2

4.5.5 Component Tutorials

The following tutorials can be loaded from the Samples area (File | Samples | Tutorials | filename.swi).

It is assumed that the user is familiar with Scripting and the drawing tools. If not, please view the Scripting Tutorials in the help manual.

.swi File	Tutorial Name	Purpose
rb.swi	Autoshape Button	Demonstrate how to create a button component based on the rounded button autoshape.

4.5.5.1 Autoshape Button

Description

This is a step-by-step tutorial demonstrating how to use an autoshape as the basis of creating a button component.

Aim

This Movie demonstrates:

- How to author a component.
- How to add Parameters via the Parameters Tab
- The use of the Update Script Tab to apply changes to the component after stage editing or reshaping.
- The use of the Apply (Before) Tab to apply parameter changes to the component.
- The use of the Shape Methods & Properties to set and retrieve the object color.
- The use of the Auto Shape Methods & Properties to set and apply the Guide Points
- How to set a referencePosition

.swi file

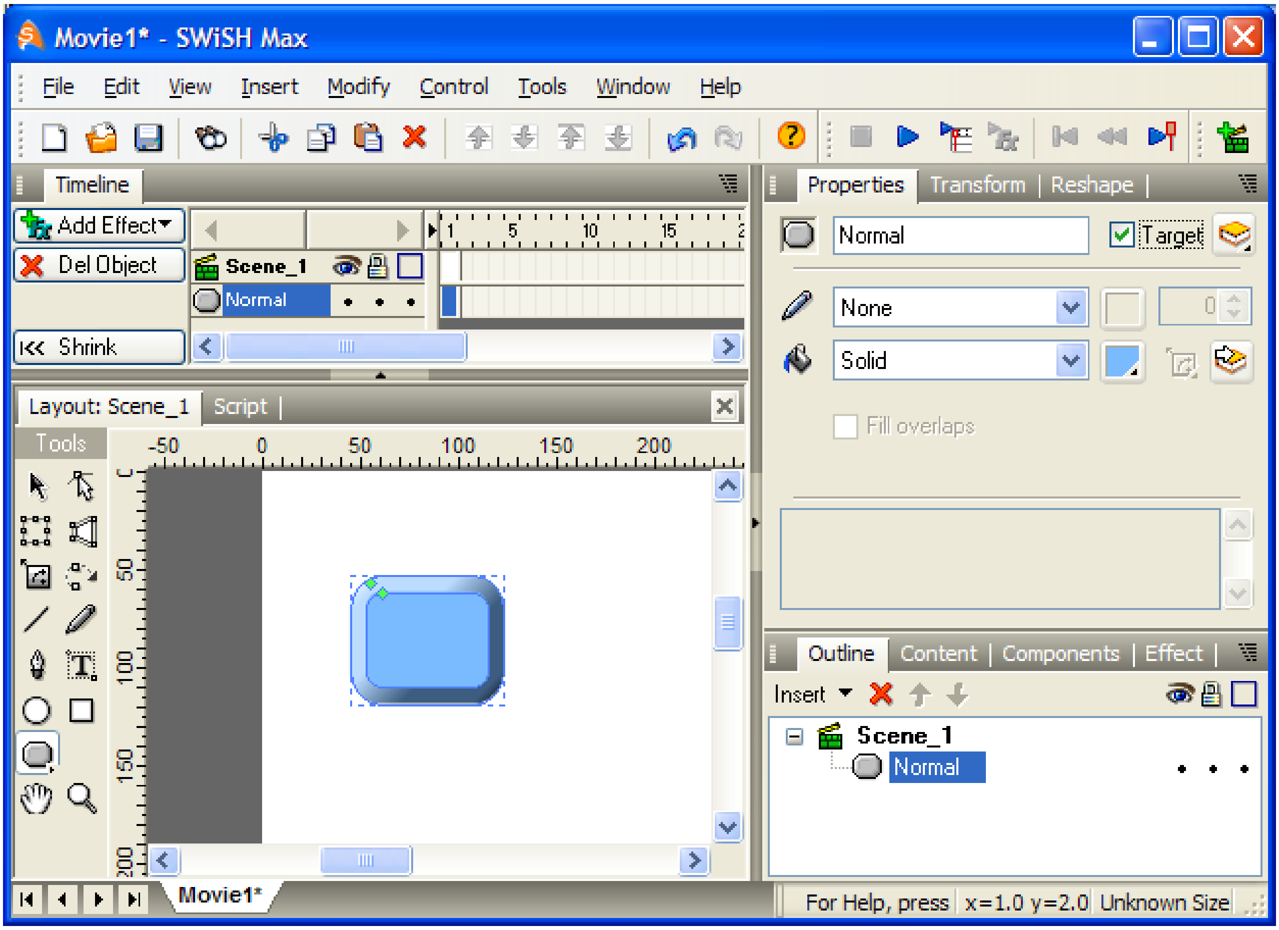
"rb.swi"

Step 1. Start with a new project (File | New) and save the file as "roundedbutton.swi"

Step 2. Use the Autoshape drawing tool to draw a rounded button.

Step 3. In the Properties Panel:

- Name the object "Normal"
- Tick the target checkbox.
- Select the line style to None.



Movie1* - SWiSH Max

Properties: Normal ☑ Target

Line style: None

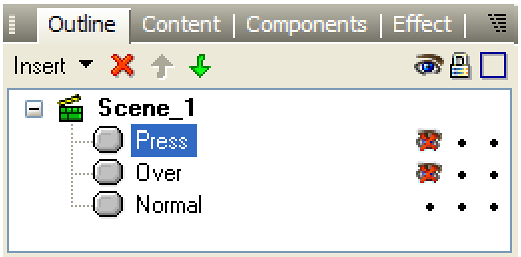
Fill: Solid

Step 4. Copy to the object to make layers for the Over and Press states. To do this:

- Right click on the object in the outline panel and select Copy.
- Then right click and select **Paste in Place** to create "Copy_of_Normal". Rename this object to be "Over". Click the corresponding eye icon to make the object normally invisible.
- Right click "Over" and select **Paste in Place** again to create :"Copy_of_Normal". Rename this object to be "Press". Click the corresponding eye icon to make the object normally invisible.

The reason that the Press and Over are made invisible is that we want the Normal object to be the one normally visible to the user when they place the component on the stage.

The outline panel should now look like the one below:



Step 5. Select the 3 objects in the outline panel, then right click and select **Grouping | Group as Movie Clip.** Name the movie clip "RoundedButton".

Step 6. With the rounded button movie clip selected, add the following script to the script panel using cut and paste:

```
onSelfEvent (load)
{
    // hide the layers that are not normally visible.
    Press._visible = false;
    Over._visible = false;
}
onSelfEvent (rollOver, dragOver)
{
    // Mouse is over, show over layer
    Over._visible = true;
}
onSelfEvent (rollOut, dragOut)
{
    // Mouse is no longer over, hide the over layer
    Over._visible = false;
}
onSelfEvent (press)
{
    // Mouse button has been pressed, show the press layer.
    Press._visible = true;
}
onSelfEvent (release)
{
    // Mouse button has been released, hide the press layer.
    Press._visible = false;
}
onSelfEvent (releaseOutside)
{
    // Mouse button is no longer pressed or over the object.
    Over._visible = false;
    Press._visible = false;
}
```

This script hides or makes visible the different layers Over and Press depending on the mouse action. The script assumes that the button will have a momentary action. ie. it shows the pressed state only when the mouse button is pressed. After adding the script re-select the layout panel if it was hidden by the script panel.

Step 7. Right click on the RoundedButton object in the outline panel and select **Author Component...**

Step 8. In the Parameters tab, click the "+ Add" to add the Parameters: ColorNormal, ColorPress and ColorOver. Enter values according to the table below:

Name	Prompt	Type	Control	Control Parameters	Assoc Prompерty
ColorNormal	Color of the button in its normal state	Color	color+alpha		
ColorPress	Color of the button when pressed	Color	color+alpha		
ColorOver	Color of the button during mouse over	Color	color+alpha		

Step 9. In the Update Script tab, enter the following script using cut and paste:

```
// This script is executed when the component is updated on the stage or the properties of any
// of the contained objects are altered.


// ====================================================
// The following script initializes the Parameters panel with the
// current color of the Normal, Press and Over objects.
// Read the solid color for the Normal layer. Note that the rounded button object has
// 1 contour, 1 fill. This was found by enabling the following trace statement:
// trace("CC" add Normal.contourCount add " fc:" add Normal.fillCount(0));
// the getFill() command and the fill object (f) are described in the
// Component Scripting section of the help file.
f = Normal.getFill(0,0);    // get the current fill for the Normal object
parameters.ColorNormal = f.color; // initialize parameter with that color.

f = Press.getFill(0,0);    // get the current fill for the Over object
parameters.ColorPress = f.color; // initialize parameter with that color.

f = Over.getFill(0,0);    // get the current fill for the Over object
parameters.ColorOver = f.color; // initialize parameter with that color.



// ====================================================
// The component attributes are set so that the user can modify the
// size / shape of the Normal object.
// If this happens, the changes need to be propagated to the Press and Over objects.

// Resize the Press and Over objects according to the current width and height of the Normal object.
Press.autoshapeWidth = Normal.autoshapeWidth;
Press.autoshapeHeight = Normal.autoshapeHeight;
Over.autoshapeWidth = Normal.autoshapeWidth;
Over.autoshapeHeight = Normal.autoshapeHeight;

// read how the autoshape guide points are set in the Normal object.
p = Normal.getAutoshapePoints();
Over.setAutoshapePoints(p);           // set the Over object to have the same autoshape guide points.
// For the Press object reflect the gradient selection point to give reverse gradient.
p[1].x = -p[1].x;
p[1].y = -p[1].y;
Press.setAutoshapePoints(p);  // set the Press object to have the same p[0] and an inverse p[1] guide point.



// ====================================================
// It is possible that the Normal Object was moved by the resizing process.
// Set the other objects to be overlayed to the same position then update the
// reference mode for all objects to be centered.

// force the Press and Over objects to have the same position.
Press._x = Normal._x;
Press._y = Normal._y;
Over._x = Normal._x;
Over._y = Normal._y;

// reset the reference point to center incase user has altered it.
Normal.referencePosition = "center";
Press.referencePosition = "center";
Over.referencePosition = "center";
this.referencePosition = "center";
```

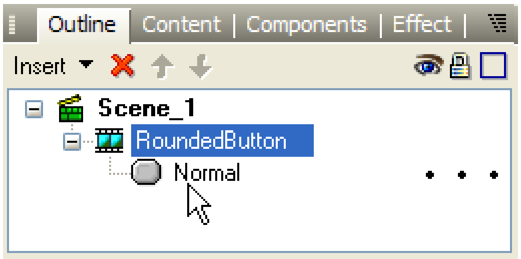
Step 10. In the Apply (before) tab add the following script using cut and paste:

```
// This script is executed whenever a parameter is updated.
// its purpose is to read the current colors from the parameters panel
// and set the corresponding object to that color.

f = Normal.getFill(0,0);    // get the current fill for the normal layer
f.color = parameters.ColorNormal;    // alter the color
Normal.setFill(f,0,0);    // save it.

f = Press.getFill(0,0);    // get the current fill for the press layer
f.color = parameters.ColorPress;    // alter the color
Press.setFill(f,0,0);    // save it.

f = Over.getFill(0,0);    // get the current fill for the over layer
f.color = parameters.ColorOver;    // alter the color
Over.setFill(f,0,0);    // save it.
```

After adding the script, press the OK button.

Step 11. Set Object properties to hide the Press and Over objects from the user.

- In the outline panel, expand the RoundedButton movie clip.
- Right click on the Normal object and select **Object Attributes..**.
- Tick the **Expose as child object** option.
- Right click on the RoundedButton movie clip and select **Object Attributes..**.
- Tick the **Conceal child objects** option.



Step 12. Test the button.

- If the Parameters panel is not visible, make it visible via the **Window** menu command.
- Select the RoundedButton object and use the parameters panel to choose different colors for ColorNormal, ColorPress and ColorOver.
- If you wish, select the object "Normal" and resize / adjust the shape of the button. The autoshape guide points can be used to adjust the corner radius and size of the inner shape as well as the direction and intensity of the gradient. Note that if the RoundedButton object is selected, entire object will be rescaled possibly resulting in distortion.
- Press the play button. The object should display different colors for mouse over and press actions.
- You can then save this .swi file to the components folder if you wish to add it to the list of standard components.

Analysis

Autoshapes in general and the rounded button in particular, provide a way to create useful buttons. The button is made up of three overlayed shape objects, Normal, Over and Press. Normal is the bottom layer and Press is the top layer. These shapes are created in steps 3 and 4. The shapes have their target checkbox set as their _visible property needs to be altered via script.

The shapes are grouped as a Movie Clip in step 5. This allows the 3 shapes to be handled as a single object.

Scripting is added in Step 6 to alter the visibility of the Over and Press shapes depending on mouse over and press actions. This script should be modified to perform the appropriate actions when the button is pressed or released.

Parameters to allow user configuration of the Normal, Over and Press colors are added in Step 8.

Step 9 shows the code that is used in the Update Script tab. This script is executed whenever the object is modified on the stage or via the properties panel.
This script is used to:

- Initialize the Parameters defined in Step 8 with the current shape colors.
- Modify the size / shape of the Press and Over shapes if the size / shape of the Normal shape changes.
- Re-adjust the position of the shapes so that they are all superimposed on each other and have their reference point set to center.

The script in the Apply (before) tab is listed in step 10. This script is activated when the user changes a parameter in the Parameters panel. The purpose of this script is to modify the shape colors to the ones currently set by the user.

Step 11 exposes the Normal shape and hides the other shapes within the component. Setting the attributes in this way allows the end user to modify the size and shape of the component my resizing or moving the guide points of the Normal object. When these changes occur, they are duplicated in the other hidden objects via the script defined in Step 9.

4.6 Scripting

SWiSH Max comes with a powerful scripting language. The language commands are defined in the Scripting Reference.

The scripting is similar to Adobe Flash Actionscript, but has SWiSH Max-specific extensions and differences .
The script can be compiled, depending on selected options, to produce script that is compatible with all Flash Players.

Tutorials

See the Scripting Tutorials section as an overview of write script and controls movies.

4.6.1 Changes in Scripting for SWiSH Max Version 2

Since the release of SWiSH Max Version 1 there have been a number of re-definitions to the Adobe™ Flash Player standard. The latest versions provide a significant improvements so SWiSH Max uses Flash Player version 9 (SWF9) as the default export format. SWiSH Max Version 1 used SWF4 as the default format as SWF4 was the most widely installed Flash Player at the time, and which would also play on any more recent player.

The support of multiple Flash Player standards means SWiSH Max has to impose restrictions onto script written for the earlier versions. These additional restrictions may cause old .swi files to fail or display errors in the Debug Panel when they are played. However, the changes required to fix the script are often simple and are often done automatically by SWiSH Max when the .swi file is loaded.

Flash Player History lists the most common problems that may cause an error when old .swi files are loaded by SWiSH Max Version 2. In some instances the points merely highlight the differences between the different SWF export versions and should be considered when writing script. The section Error Messages gives a detailed description of the most commonly encountered error messages.

4.6.1.1 Flash Player History / Bugs

Different versions of the flash player impose different rules for scripting. Errors that occur in a specific version may be due to enhanced error checking that has been introduced since the earlier version. The list below shows the common errors and changes between the versions.

- Later versions of Flash will produce unpredictable results if variables are not defined before use. SWiSH Max provides an Error message if a variable is not defined. In some instances this may also create an error like: `"cannot find _loadXXXXXXXX"`. This normally occurs when one .swf loads another .swf file. To fix the error make sure all .swf files are re-exported with SWiSH Max.
- SWF4 to SWF6 have case insensitive variable / object names. With SWF7 to SWF9 variable / object names are case sensitive, methods are case sensitive but properties are case insensitive.
- To support additional properties introduced by SWF6, SWiSH Max has the Expose SWF6 Properties export option as well as _text and _button properties. These changes may affect how dynamic text and button objects are accessed. See the TextField Access Matrix for a full description of how this affects the Text Field object.
- SWiSH Max Version 1 allowed the use of () as a short hand method of referencing eval(). ie. (objname)._x was previously allowed. This must be coded as eval(objname)._x
- eval() previously supported Slash Notation. This is no longer supported for SWF5 or greater.

- Reserved words are checked irrespective of the chosen SWF export level. eg. Date is a reserved word in SWF5+. An error will result if Date is used as a variable name in a swi file intended for SWF4 export.
- Checks are made for sections of code that are too long. The Flash Player has an internal limitation that loops and branch statements can only span a maximum of 32k byte codes. This is normally not a problem, but if you receive an error message like `"This movie has script that is too long or complex in ...",` Then you should break your code into smaller sections. Possibly by moving some code to user defined functions and by doing repetitive operations in loops. Note that comments do not affect the code size and should be left in place. See also Debug Panel.

Noted Bugs / Issues

Note: This is not a complete list. Please refer to the adobe website for full release note documentation.

SWF7 and earlier

- Spaces may be lost with html rendering of text.

For example:
<B>This is</B> very <U>interesting</U> text
<B>This is</B> <U>very interesting</U> text
This <I>is</I> <U>very</U> <B>interesting</B> text

when exported as SWF7 or earlier give incorrect results with spaces lost:

This is very interesting text
This isvery interesting text
This isveryinteresting text

When exporting as SWF8, the spaces are preserved.

SWF8 and possibly later

- Flash Player does not play streamed sounds on the first frame. This is by design to prevent the playback of a snippet of sound during a stop() action on the first frame of a movie. The workaround is to place a sound on the second frame of a movie. (84631)

All versions

- onSelfEvent(keyPress()) events may not get sent to all objects. Typically the object that is first in the tab order will be sent the event.

4.6.2 Introduction to Scripting

This section is designed for those not familiar with programming or Scripting languages. The following sections, and the tutorials, give background theory and some instruction on how to script.

- What is Scripting
- What is Possible with Scripting
- Variables
- Flow Control
- Objects

4.6.2.1 What is Scripting

SWiSH Max uses scripting to allow a user to program a Movie and items within a Movie.

Scripting can be used to define Actions that will occur at a specific Frame, when two Objects collide, or when some other external input is applied to the Movie.

Scripting can also be used to control the playing of sounds, the loading of other sections of the Movie as well as control the Movie playing within a Movie Clip.

Scripting can be used to interact with external scripts, such as PHP and ASP.

Scripting can be used to define physics properties of scripted objects, such as friction and acceleration. Physics properties are a SWiSH Max extension to Adobe Flash Actionscript.

4.6.2.2 What is Possible with Scripting

The following things can be achieved with Scripting:

- user input (forms)
- games
- math and physics
- web-based calculators
- dynamic presentations
- send and receive data from web pages
- run JavaScript functions embedded in an HTML page
- communicate with .php, .asp, and Perl scripts
- interface with a database through external scripts
- load .XML

... and much, much more.

4.6.2.3 Variables

All programming languages, make use of a concept called a 'variable'. A variable is analogous to the memory button of a calculator. It can be used to store the result of intermediate calculations and data. Unlike most calculators which only have 1 memory area, scripting can support multiple variables. The limit is defined by the size and power of the computer system running the Flash Player. To allow access to a specific variable, each variable is given a name. The name can include the characters, a...z, A...Z, 0...9, and _ (underscore). The . (dot) and [ ] characters have special meaning, described elsewhere.

Consider the following statement,

```
x = 3 + 1;
```

the result of the calculation (3 + 1) is assigned to the variable 'x'.

That variable can be used in subsequent calculations, such as:

```
y = x * x;      // y = 16
x = x + 1;      // x = 5, (previous value of x + 1).
```

Note that // is used to define a comment area. Comments are useful reminders of what the script does.

Variables can also be used to hold a text message (also called a 'string'), e.g. in this example below. 'message' is a string

```
message = "Warning do not press that button"
```

Note:

- Flash players 4, 5 and 6 have case insensitive variable names. eg. MyVariable and myvariable are the same variable.
- Flash players 7, 8 and 9 have case sensitive variable names, but case insensitive properties. eg. MyVariable and myvariable are different variables but MyVariable._x and MyVariable._X represent the same property.

See Naming Conventions concerning the naming of variables and objects.

4.6.2.4 Flow Control

All Scripting and programming languages execute commands in a defined order. This execution order is generally 'top to bottom'. SWiSH Max scripting also performs commands in a top-to-bottom manner. This general order of execution can be altered through the use of Flow Control commands.

Flow Control commands can be categorized into Looping, Conditional and Function call commands.

Looping commands repeat a section of code until a certain condition becomes false. Available looping commands are:

- do..while
- for
- while.
- for...in

Conditional commands allow specific sections of script to be executed if a certain condition is true or false. Available conditional commands are:

- if
- else
- else if.
- switch

Function call commands cause the execution flow to move to the named function, execute code within that function and return to original location after a return command is encountered in the named function. Available commands are:

- function
- return.

4.6.2.5 Objects

Scripting is an Object-Oriented language.

The term 'Object-Oriented' is primarily a programming concept that broadly means that an Object is any item that contains data and behaves in a defined way. An Object-Oriented language sorts similar Objects into a 'classes'.

The class to which the Object belongs defines:

- the type and meaning of the data that the Object contains (properties)
- the way in which the Object and data can be manipulated (methods)
- the way in which the Object behaves during certain conditions (events).

For example, consider a class that defines animals. That class could contain the animal's name, type, number of legs, noise and so on. These attributes would be properties. For example a pet cat Fleebag and pet dog Growler could both be considered Objects belonging to the class animal.

The class animal could be (partially) defined as follows:

Properties

- Name
- Type
- Noise
- Angry

Methods

- MakeNoise()
- MakeAngry()
- Walk()
- Feed()

Events

- OnAngry()
- OnHungry()
- OnTired()

When referring to properties and methods within an Object, the dot '.' character is used to delineate the object name and the contained property / method or event.

```
onHungry {
    pet1.Name = "Fleebag"; // assign the name "Fleebag" To pet1.
    pet1.Feed();       // feed the cat.
}
```

The method Feed() will be executed once pet1 becomes hungry.

Movies Clips Buttons and Text Objects can all be considered to be Scripting Objects. These Objects have pre-defined properties, methods and events associated with them.

4.6.2.5.1 Naming Conventions

As stated previously, the . (dot) character is used to delineate the Object name and the contained property/ method or event. An Object can also contain 'child' objects. Again, the . (dot) character is used to delineate between the Object and its child Object.

Assume an Object named **obj1** has a property **_x**. That property can be accessed as **obj1._x**
If **obj1** is a child object of **obj2,** then the reference to **obj1._x** is only valid from within **obj2**.
To access the same property from outside of **obj2**, the full name is required. This name would be **obj2.obj1._x**

Because of the hierarchical nature of Object names, the following synonyms _root, _parent and this are implemented to represent (in order) the top-most object (always the main Movie), the immediate parent and the current Object. These synonyms are described in further detail in the Scripting Reference Manual.

Up to Flash Player version 9 (the current version at time of release), the property names are case insensitive. ie. obj1._x is the same as obj1._X. This assumption may not apply to later versions. It is safer to code all properties as if they are case sensitive.

4.6.2.5.2 Properties

A 'property' is an item that defines something about the Object. Some properties are read only, whereas others are read/ write. The properties that an Object contains are defined according to the class it belongs to. This means that different types of Objects may have different properties.

For example, a Movie Clip and standard drawing Objects contain a number of properties that define the Object's size and position within the Movie.

Properties include:

- screen Position, _x, _y
- size, _width, _height
- scale, _xscale, _yscale
- rotation angle, _rotation
- opacity, _alpha.
- visibility, _visible
- name, _name

A text field Object contains all of the Movie Clip or Scripting Object properties, but also has its own additional properties:

- maxscroll
- scroll
- text

The properties that are available for each supported Object are defined in the Script Reference Dictionary.

4.6.2.5.3 Methods

A method is used to define ways in which an Object can be manipulated. The methods that an Object contains are defined according to the class it belongs to. This means that different types of Objects may have different methods.

For example, a Movie Clip contains a number of methods that can be used to control or manipulate the Movie Clip.

MovieClip Methods include:

- play()
- stop()
- duplicateMovieClip()
- gotoAndPlay().

A String Object contains various methods that allow various string manipulations to be performed.

The methods that are available for each supported Object are defined in the Script Reference Dictionary.

4.6.2.5.4 Events

An event is a defined occurrence that can apply to an Object. The events that are associated with an Object are defined according to the class the Object belongs to. This means that different types of Objects may have different events.

For example, a Movie Clip Object supports a number of different events.

These events include:

- onFrame ()
- onSelfEvent (load)
- onSelfEvent (enterFrame)
- onSelfEvent (mouseEvent).

An Event Handler routine can exist within the Object for each of the above events. When the defined event occurs, Script within the Event Handler function is executed.

The events that are available for each supported Object are defined in the Script Reference Dictionary.

4.6.2.5.5 Functions

Scripting Objects can also contain 'functions'. Functions are a way of creating re-useable sections of script. This can increase the modularity and readability of your Script.

Functions are an ideal way of expanding the methods that can be applied to a particular Object. Functions that are defined within an Object are local to that Object.

To create functions that are available to all Objects, either create them in the main Movie (they will then be available as _root.function()), or create a Movie Clip specifically to hold these functions (they will then be available as _root.movieclipname.function()).

For example, a function MoveToZero() can be defined to move an Object to position 0.

```
Function MoveToZero() {
   _x = 0;
   _y = 0;
}
```

The script command
`ball.MoveToZero();` will cause the ball object to move to 0,0

In this case MoveToZero() is effectively a local method of the ball Object.

4.6.2.5.6 Included Objects

SWiSH Max comes with the following pre-defined Objects:

Object Name	Description
Array	Individualy listed elements in a common array.
Button	A Button. This supports the additional Event Handling function on()
Color	Color object
Date	Allows access to local and universal time
Math	Used to provide mathematical functions
Movie Clip	A Movie Clip
Scripting Object	Any shape with the target checkbox ticked
Sound	Used to provide control for the sound object.
String	Used to handle and manipulate strings.

4.6.2.6 Recommendations

It is helpful to write script intuitively and consistently. The following recommendations regarding Naming Conventions, Comments, Handling Scope and Structure and Format will assist in creating script that is more robust, easier to understand, simpler to maintain and re-useable. The suggestions can be viewed in detail by following the associated hyperlinks.

4.6.2.6.1 Naming Conventions

- Names can only contain letters, digits, dollar signs ($), and underscores (_). Names cannot start with a digit. Names starting with dollar signs and underscores have special meanings, so you should not use them for your own names. This means all your names should begin with a letter.
- Names should be unique, within a given context (for example, within a single movie clip).
- Names should be descriptive, but as short of possible.
- Only use single-character variable names for local variables in loops.
- Do not use the same name with different case. For example, do not use both `myname` and `myName`. Treatment of case depends on the chosen flash player version. Consequently, different case does not guarantee different object or variables, nor does it guarantee the same object or variable.
- Avoid abbreviations in names. If used, make sure the same abbreviation always has the same meaning.
- Join words to create meaningful names.
  - For variables and function, the first letter should be lower case, and subsequent words should start with an upper case letter. For example: use `myLongVariable` rather than `mylongvariable`.
  - For prototypes (or classes) the first letter should be upper case. For example: `MyClass`.
  - For names of constants, use all upper case letters with underscores to separate the words. For example: `const MAX_WIDTH = 40`.
- Do not use reserved or key words, or reuse predefined class, method, or property names, for your own variables.
- Always use data types when declaring your local variables. This helps document what type of value the variable holds, and lets the compiler find some scripting errors for you. For example `var counter : Number = 0;`
- Use descriptively named constants for all numbers other than 1, 0, or -1.
- Boolean (or true/false) variable should begin with the word 'is'. For example, `isPlaying = true;`
- For functions or methods that return values, the function name should describe what the function returns.
- For functions or methods that perform some operator, the function name should describe what the function does. Such functions usually start with a verb.
- Append a suffix using an underscore to remind you of the type of an object. In addition to the easy identification of the datatype associated with the object, the suffix with a leading underscore will prevent unintentional clashes with existing property names within the parent object. Commonly used suffixes include:
  - "_txt" for a text field
  - "_mc" for a movie clip
  - "_btn" for a button
  - "_shape" for a shape
  - "_ext" for an external media object
  - or append the object type name (with leading lower case letter). Eg "_array" for an Array.

4.6.2.6.2 Comments

Comments document how, why and what your script does. This is helps you understand your code if you want to maintain it or reuse it later. Comments make no difference to the exported file size, so you can use them as much as needed without worrying about how it will affect your exported SWF file.

- You should add comments for everything that is not obvious from reading the code itself.
- Use block comments (/* and */) for multi-line comments. You can use these at the start of a block of code for a frame or function to explain what the block of code does. When used to describe a function, comments can be used to explain the intention of each of the parameters and the return value.
- Use single-line comments (//) for short comments. You can use these within a block of code to document the next few lines of script.
- Use trailing line comments (// at the end of a line). You can use these to document the single line of script on which they appear. These are especially useful during variable declarations to describe the data that a variable is intended to hold.
- Use blank lines to split your code into groups of related lines.

- If you find you need a lot of comments to explain how your code works, then that can be an indication that you need to simplify your code, or use more meaningful names

4.6.2.6.3 Handling Scope

Scope is where a given name can be used to access its associated the variable or function.

- Some variables are “Local Variables”, and their scope is limited to the function in which you declare them; if you try to access them from outside the function, then the variable will not be found. Local variables are declared with the ‘var’ statement within a function. If possible, local variables should be used in preference to Object or Global variables.
- Some variables are ‘MovieClip” or “Object” variables, and their scope is limited to the movie clip or object in which they are declared, for as long as that movie clip or object exists. The _root of a movie is a particular example of a MovieClip variable.
- Some variables are “Global Variables”, and their scope lasts as long as the movie is playing. They are members of the special _global object. Global variables provide a convenient way of passing data between objects and functions. However, care should be taken to clearly document how the global data is manipulated. This will help minimize the potential of unwanted "side effects".
- Use dot (.) or bracket ([ ]) notation to specify a ‘path’ to a variable. Be careful using ‘absolute’ path, ie paths that begin with _root, as this may have undesired affects if you load your SWF into a MovieClip in another movie, as _root always means the _root of the movie you are loading into, not the _root of the SWF being loaded. Instead, use ‘relative’ paths that start with this or _parent.
- If you need to load a SWF that inappropriately uses _root, then you may be able to use _lockroot to change which MovieClip _root refers to in the loaded SWF.
- Whenever possible, use the this keyword as a prefix for MovieClip variables for the current movie clip, even if your code works without it, never use any path prefix for local variables.

4.6.2.6.4 Structure and Format

- Add block comments to the start of frames and functions.
- Declare all MovieClip variables in an onSelfEvent(load) event, and give them initial values. Un-initialized variables behave differently according to the flash player version. For consistent behaviour, all variables should be initialized.
- Declare all local variables for a function at the start of the function, using data types, with a separate var statement for each variable. Ensure you initialise each local variable.
- Avoid the with statement, instead explicitly put paths on variables.
- Use functions for script reuse, resulting in smaller SWF files and fewer lines of script. Document each function with comments describing the function parameters and return values.
- Break large blocks of code into smaller concise functions. This will help avoid errors caused by the Code Section Limit.
- Use consistent indentation.
- Put a space after a keywords that is followed by an open parenthesis or brace. For example,

```
do {
    x = x * 2;
} while (x > y);
```

- Do not put a space between a method or function name and the open parentheses. For example: use `play()` not `play ()`
- Do not put a space between an object name and a the dot (.) and bracket ([ ]) operators. For example, `_root[clipName].stop()`
- Include a space after commas in a list of function or method arguments. For example: `duplicateMovieClip(“newClip”, depth);`
- Put a space between all operators (except for the dot operator) and their operands. For example: `var sum : Number = x + y;`
- Do not put a space between a unary operator and its operands. For example: use `i++` not `i ++`
- Do not put a space after an opening parenthesis or before a closing parenthesis.
- Put each statement on a separate line.
- Do not embed assignment statements. For example: use `b = c + d; a = b + e;` not `a = (b = c + d) + e;`
- If you need to break a long statement into two lines, break it after an operator.
- Use braces ({}) for if, for, while, do statements, even if they are not strictly necessary
- Place the opening brace on the same line as the statement keyword, and indent the lines within the

braces. For example:

```
while (condition) {
    // statements
}

if (condition) {
    // statements
} else {
    // statements
}

switch (condition) {
case CASE_1 :
    // statements
    // falls through
case CASE_2 :
    // statements
    break;
case CASE_3 :
    // statements
    break;
default :
    // statements
    break;
}
```

- Use parentheses to group conditions. For example: `if ((x < 0) && (y > 0)) { … }`

4.6.3 Tools

The main tool for adding script to the Movie is the Script Panel. In the default configuration, the script panel is docked with the Layout Panel. The Script Panel, and Script editor, is used to enter script commands via the 'Add Script' pull-down Menus (using the Assist Pane) or via direct text entry.

The Assist Pane should be used by authors unfamiliar with the available script commands or the properties associated with the different Scripting Objects. The Assist Pane allows selection of the available commands, properties and methods via a pull-down menu system.

To see what script is doing (or not doing), use the Debug Panel and Trace command. When the script is played by the internal player, a trace command will cause the associated text or variable to be displayed in the debug panel.

Add comments to the script program using the // and /* */ comment delimiters. The use of comments can greatly increase the readability of script code, making debugging and future modification significantly easier.

Comments are not exported and are only saved in the .swi file, so there is no speed penalty associated with their use.

Breaking code over multiple lines

Code can be run over multiple lines of the script editor.

e.g.

```
function(para1,
         para2,
         para3);
```

Text should be joined using '+' or 'add'.

e.g.

```
textVariable = "this data" +
               "this data" +
               "this data and" +
               "this data";
```

or

```
textVariable = "this data" add
                    "this data" add
                    "this data and" add
                    "this data";
```

In some cases 'add' is used in preference to '+' because '+' can be confused with addition. eg. "2" + "5" = "7" whilst "2" add "5" = "25"

4.6.3.1 Add Script

This page lists all the events, actions and operations which are inserted by the 'Add Script' tree.

Events			
	Frame		
		onFrame (...)	*Event*
		onSelfEvent (load)	*Event*
		onSelfEvent (enterFrame)	*Event*
	Button		
		on (press)	*Event*
		on (release)	*Event*
		on (rollOver)	*Event*
		on (rollOut)	*Event*
		on (dragOver)	*Event*
		on (dragOut)	*Event*
		on (releaseOutside)	*Event*
		on (keyPress(...))	*Event*
	Self		
		onSelfEvent (press)	*Event*
		onSelfEvent (release)	*Event*
		onSelfEvent (rollOver)	*Event*
		onSelfEvent (rollOut)	*Event*
		onSelfEvent (dragOver)	*Event*
		onSelfEvent (dragOut)	*Event*
		onSelfEvent (releaseOutside)	*Event*
		onSelfEvent (keyPress(...))	*Event*
	Text		
		onSelfEvent (changed)	*Event*
		on (changed)	*Event*
Define function(...)			
Frame			
	setLabel(...)		*Action*
	preloadContent()		*Action*
Movie Control			
	play()		*Action*
	stop()		*Action*
	Goto and play		
		gotoAndPlay(FRAME)	*Action*
		gotoAndPlay(LABEL)	*Action*

		nextFrameAndPlay()	*Action*
		prevFrameAndPlay()	*Action*
		nextSceneAndPlay()	*Action*
		prevSceneAndPlay()	*Action*
	Goto and stop		
		gotoAndStop(FRAME)	*Action*
		gotoAndStop(LABEL)	*Action*
		nextFrameAndStop()	*Action*
		prevFrameAndStop()	*Action*
		nextSceneAndStop()	*Action*
		prevSceneAndStop()	*Action*
Dynamic Movie Clips			
	duplicateMovieClip(...)		*Action*
	removeMovieClip()		*Action*
External files and data			
	Load/unload Movie Clip		*Action*
		loadMovie(...)	*Action*
		loadVariables(...)	*Action*
		unloadMovie(...)	*Action*
	Load/unload Level		
		loadMovieNum(...)	*Action*
		loadVariablesNum(...)	*Action*
		unloadMovieNum(...)	*Action*
Browser/Network			
	getURL(...)		*Action*
	fscommand(...)		*Action*
	javascript(...)		*Action*
	mailto(...)		*Action*
	playSound(...)		*Action*
	stopSound(...)		*Action*
	stopAllSounds()		*Action*
Mouse Dragging			
	startDrag(...)		*Action*
	stopDrag()		*Action*
Statements			
	var name		*Action*
	const name		*Action*
	name = expr		*Action*
	function(...)		*Action*
	call(...)		*Action*
	return ...		*Action*
	evaluate		*Action*
Conditional			
	if (...)		*Action*
	else		*Action*
	else if (...)		*Action*
	if (frameLoaded(...))		*Action*
	if (isNearThis(...))		*Action*
	if (_droptarget==...)		*Action*
	if (chance(...))		*Action*
	switch (...) {...}		*Action*
	case ...:"		*Action*
	default:		*Action*
	break		*Action*
Looping			
	while (...)		*Action*
	do		*Action*
	for (...;...;...)		*Action*

	for (... in ...)	*Action*
	break	*Action*
	continue	*Action*
Blocks		
	{..}	*Action*
	with (...)	*Action*
	tellTarget (...)	*Action*
	Exceptions (SWF7+)	*Action*
	try {...}	*Action*
	catch (...) {...}	*Action*
	finally {...}	*Action*
	throw ...	*Action*
Debugging		
	/* comments */	*Action*
	// comments"	*Action*
	trace(...)	*Action*

4.6.4 Events

Scripting is used to support Events and Actions. Script can be viewed and modified via the Script Panel.

All Actions are triggered in response to an Event. Any Scripting Object can have Events associated with it. When an Event occurs, the Event Handling Routine executes any Actions that are defined for that Event.

Events can occur in response to reaching a certain Frame, or in response to mouse or keyboard Actions.

An Event can trigger more than one Action. For example, when the mouse rolls over an Object, the Movie can be stopped (with a Stop Action) and the browser can be told to load an URL into another Frame (with the getURL Action).

There are three types of Events:

- Frame
- Button
- Self.

Input Text Objects also have the additional Events on (changed) and onSelfEvent (changed) available.

4.6.4.1 Frame Events

Frame Events occur when a Scene or Movie Clip reaches a specified Frame.

The available Frame Event Handlers are as follows:

- **onFrame**: the Event is triggered just before the specified Frame is displayed
- **onSelfEvent (load)**: the Event is triggered when a Movie Clip or Scene is first displayed. onSelfEvent (load) Events are handled before any onSelfEvent (enterFrame) or onFrame Events
- **onSelfEvent (enterFrame)**: the Event is triggered as each Frame is entered. onSelfEvent (enterFrame) Events are handled before any onFrame Events for a Frame.

Note: Event Methods also exist for onSelfEvent (enterFrame) and onSelfEvent (load).

For Actions, other than the Preload Content and Set Label Actions, the Event is triggered just before the specified Frame is displayed. See the Preload Content and Set Label Actions for information about how they work.

Insert a Frame Event either by:

- **using the Timeline**: selecting the Scene row and pressing Add Script. This will create an onFrame Event Handler for that Frame
- **using the Script Panel**: selecting a Scene and pressing Add Script, and selecting Frame and the required Frame Event.

4.6.4.2 Self Events

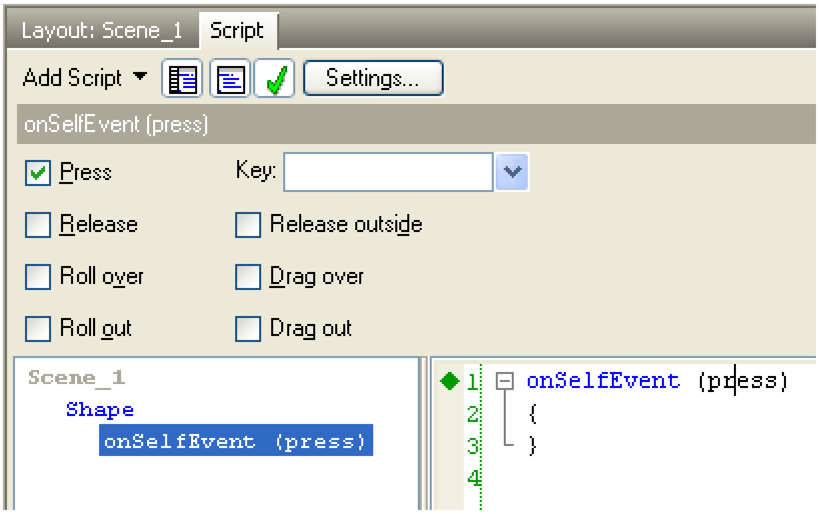
Self Events occur when the mouse interacts with an Object in a Scene or a defined key is pressed. Only Objects that are defined as Targets can use Self Events.

More than one Self Event can trigger the same Action, as shown in the example below.

The Actions in a Self Event Handler apply to the Object itself. For example, with Movie Clips, if a Self Event handler contains a stop() Action, then that Action stops the Movie Clip itself, not the main Movie. If the Actions is to apply to the Movie, use a Button Event instead.

Insert a Self Event either by going to the Script Panel, selecting an Object, and pressing Add Script | Events | Self | OnSelfEvents(Press). Click in the parameters list (i.e. in this case "Press") to see the Properties.

When a Self Event for an Object is selected, the Self Event Properties are shown above the Script Outline Pane in the Script Panel.



☑ Press Key:

□ Release □ Release outside

□ Roll over □ Drag over

□ Roll out □ Drag out

Scene_1
Shape
onSelfEvent (press)

```
onSelfEvent (press)
{
}
```

This contains a list of check boxes showing the Self Events. A combination of options are available for Self Events, such as Roll Over and Roll Out, so that the same list of Actions can be performed for different Events. The Events and their Event Handlers are as follows:

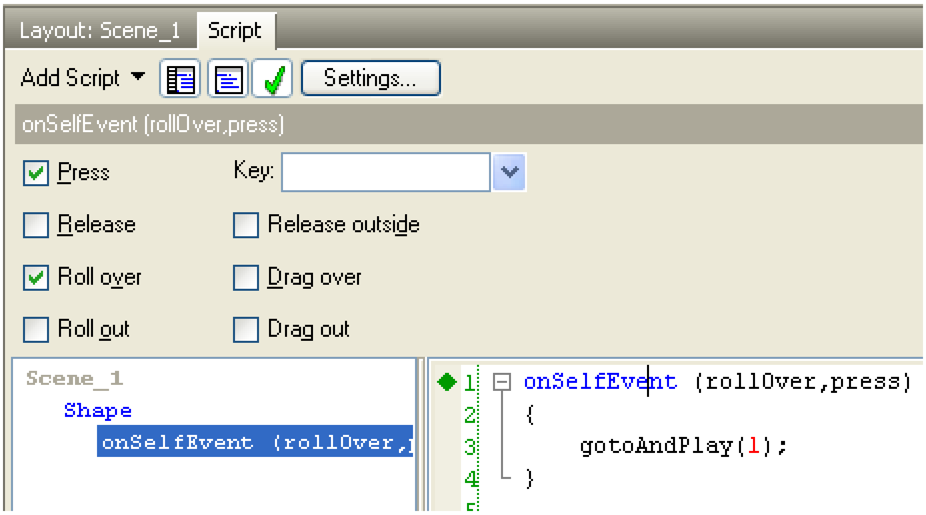
- **onSelfEvent (press)**: press the left mouse button while the cursor is over the Movie Clip
- **onSelfEvent (release)**: release the left mouse button while the cursor is over the Movie Clip
- **onSelfEvent (rollOver)**: move the mouse cursor from outside the Movie Clip to over it with no mouse button pressed
- **onSelfEvent (rollOut)**: move the mouse cursor from over the Movie Clip to outside it with no mouse button pressed
- **onSelfEvent (dragOver)**: move the mouse cursor from outside the Movie Clip to over it with the left mouse button pressed
- **onSelfEvent (dragOut)**: move the mouse cursor from over the Movie Clip to outside it with the left mouse button pressed
- **onSelfEvent (releaseOutside)**: release the left mouse button after moving the cursor outside the Movie Clip
- **onSelfEvent (keyPress)**: press the define Key. The required key can be entered in the Key field or

selected from the common keys drop-down Menu.

Note: Event Methods also exist for each of the above events.

Note: Due to limitations of the Flash Player, the Play Sound and Stop Sound Action cannot be triggered by Drag Over, Drag Out or Release Outside Events

In the example below, both Press and Roll Over Actions will send the Movie to Frame one.



onSelfEvent (rollOver,press)

☑ Press Key:

□ Release □ Release outside

☑ Roll over □ Drag over

□ Roll out □ Drag out

Scene_1
 Shape
  onSelfEvent (rollOver,p

```
onSelfEvent (rollOver,press)
{
    gotoAndPlay(1);
}
```

4.6.4.3 Button Events

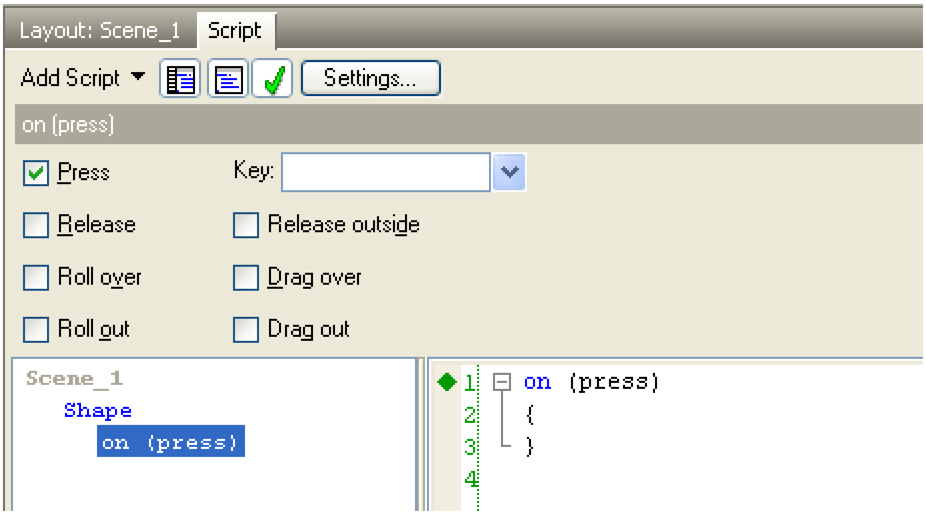
Button Events occur when the mouse interacts with an Object in a Scene or a defined key is pressed. Adding a Button Event to an Object makes it behave like a button and allows it to respond to the mouse and keyboard.

More than one Button Event can trigger the same Action, as shown in the example below.

The Actions in a Button Event Handler apply to the Scene or Movie Clip containing the Object. For example, if a Movie Clip has a Button Event Handler that contains a stop() Action, then that Action stops the Scene containing the Movie Clip, it does not stop the Movie Clip itself. To have the Actions apply to a Movie Clip, use a Movie Clip Event instead.

Insert a Button Event either by going to the Script Panel, selecting an Object, and pressing Add Event.

When a Button Event for an Object is selected, the Button Event Properties are shown above the Script Outline Pane in the Script Panel.



on (press)

☑ Press Key:
□ Release □ Release outside
□ Roll over □ Drag over
□ Roll out □ Drag out

```
on (press)
{
}
```

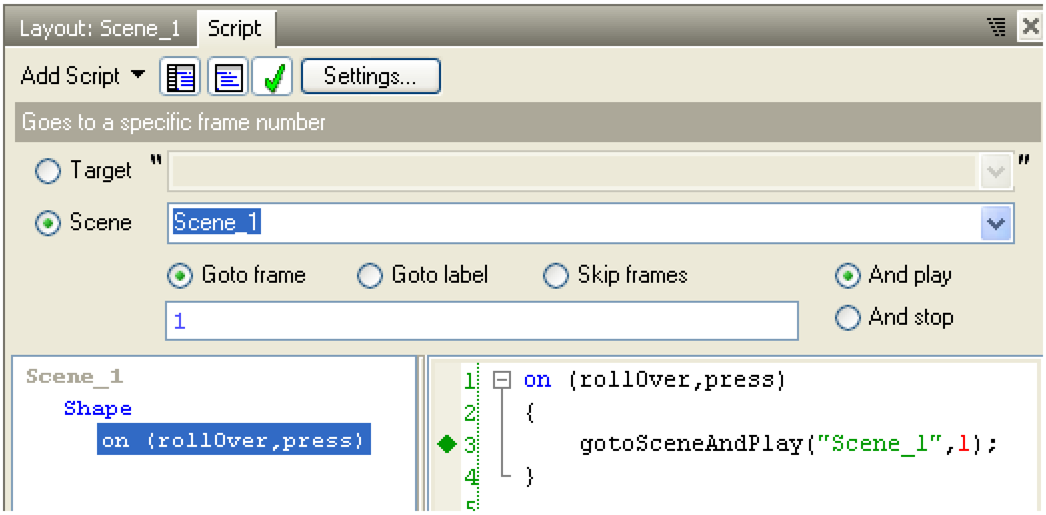
This contains a list of check boxes showing the Button Events. A combination of options are available for Button Events, such as Roll Over and Roll Out, so that the same list of Actions can be performed for different Events. The Events and their Event Handlers are as follows:

- **on (press)**: press the left mouse button while the cursor is over the button
- **on (release)**: release the left mouse button while the cursor is over the button
- **on (rollOver)**: move the mouse cursor from outside the button to over it with no mouse button pressed
- **on (rollOut)**: move the mouse cursor from over the button to outside it with no mouse button pressed
- **on (dragOver)**: move the mouse cursor from outside the button to over it with the left mouse button pressed
- **on (dragOut)**: move the mouse cursor from over the button to outside it with the left mouse button pressed
- **on (releaseOut)**: release the left mouse button after moving the cursor outside the button
- **on (keyPress)**: press the define Key. The required key can be entered in the Key field or selected from the common keys drop-down Menu.

Note: Event Methods also exist for each of the above events.

Note: Due to limitations of the Flash Player, the Play Sound and Stop Sound Action cannot be triggered by Drag Over, Drag Out or Release Outside Events. A simple way around this limitation is to place your sounds in a Movie Clip called "sound", then use Actions such as sound.gotoAndPlay(2) where Frame 2 of the Movie Clip will cause the sound to play.

In the example below, both Press and Roll Over Actions will send the Movie to Frame one.



Goes to a specific frame number

○ Target " "

● Scene Scene_1

● Goto frame ○ Goto label ○ Skip frames ● And play

1 ○ And stop

on (rollOver,press)

```
on (rollOver,press)
{
    gotoSceneAndPlay("Scene_1",1);
}
```

Note: Button Events are assumed to work on the _root or main Movie layer. Consequently Actions such as _X += 5; which may be expected to move the button, will actually move the entire Movie to the right by 5 pixels.

4.6.4.4 Text On Changed

Two additional events are available for input Text Objects. The Event Handlers are as follows:

- **on (changed)** and
- **onSelfEvent (changed)**

Although the events are triggered when the text field changes, they behave subtly differently. Code within the on (changed) event function will operate with the parent as the target object whereas onSelfEvent (changed) will operate directly on the current object.

Note:
The Event Method onChanged() can also be used.

Note:
There is a bug in Flash Player 6 where an OnChanged Event will cause it to crash if the Action triggered removes the Text Object. For example, changing the Scene with an OnChanged Event will cause the Flash Player to crash.

4.6.5 Actions

Actions are operations that are triggered by Events.

Actions can alter the playing of the Movie, start or stop sounds, load other Movies or web pages, or communicate with the hosting browser or player.

Actions appear as Script Commands within Event Handler Functions. These can be viewed/modified via the Script Panel.

The 'Add Script' menu, and Assist Pane, contains many useful and frequently used actions; however, even more actions can be added by typing directly into the Script panel. For more detailed descriptions of these and other available actions, please refer to the Script Reference section.

The Actions available using 'Add Script' are:

- Frame
  - setLabel(...)
  - preloadContent()
- Movie Control
  - play()
  - stop()
  - Goto and play
    - gotoAndPlay(FRAME)
    - gotoAndPlay(LABEL)
    - nextFrameAndPlay()
    - prevFrameAndPlay()
    - nextSceneAndPlay()
    - prevSceneAndPlay()
  - Goto and stop
    - gotoAndStop(FRAME)
    - gotoAndStop(LABEL)
    - nextFrameAndStop()
    - prevFrameAndStop()
    - nextSceneAndStop()
    - prevSceneAndStop()
- Dynamic Movie Clips
  - duplicateMovieClip(...)
  - removeMovieClip()
- External files and data
  - Load/unload Movie Clip
    - loadMovie(...)
    - loadVariables(...)
    - unloadMovie(...)
  - Load/unload Level
    - loadMovieNum(...)
    - loadVariablesNum(...)
    - unloadMovieNum(...)
- Browser/Network
  - getURL(...)
  - fscommand(...)
  - javascript(...)
  - mailto(...)
  - playSound(...)
  - stopSound(...)
  - stopAllSounds()
- Mouse Dragging
  - startDrag(...)
  - stopDrag()
- Statements
  - var name
  - const name
  - name = expr
  - function(...)
  - call(...)
  - return ...
  - evaluate
- Conditional
  - if (...)
  - else
  - else if (...)
  - if (frameLoaded(...))
  - if (isNearThis(...))
  - if (_droptarget==...)

if (chance(...))
switch (...) {...}
case ...:
default:
break

Looping

while (...)
do
for (...;...;...)
for (... in ...)
break
continue

Blocks

{..}
with (...)
tellTarget (...)

Exceptions (SWF7+)

try {...}
catch (...) {...}
finally {...}
throw ...

Debugging

/* comments */
// comments
trace(...)

4.6.5.1 Common Concepts

Some concepts are common to many script commands. These concepts are described in this section.

It may be easier to skip this section and rely on hyperlink references when necessary, if so continue from Set Label.

- Target
- Slash Notation
- User Defined Functions
- Evaluate Button

4.6.5.1.1 Target

Many actions make reference to a 'Target'. The target is generally the Movie, Movie Clip or Object that the Action applies to.

The target can often be selected via a pull-down list in the Script Panel. Alternatively it can be typed in long hand.

The Object Synonyms '_root', '_parent' and 'this' can be used when the target refers to a Scripting Object. Note that slash notation is also accepted and is converted to the Object synonyms.

4.6.5.1.2 Slash Notation

In many cases, Object names are referred to as being in 'slash' notation. This means that the full path to the Object is specified via slashes where / represents the top or 'root' level.

For example, consider that a main Movie contains a Movie Clip s1 and that Movie Clip (s1) contains another Movie Clip s2.

The full slash notation name of s2 is "/s1/s2".
This is also defined by the _target property.
Therefore ...

```
_root.s1.s2._target
```

...would return the string "/s1/s2".

4.6.5.1.3 User Defined Functions

Define a function for any Scripting Object via the Define function option in the 'Add Script' tree. User defined functions are used to execute a series of statements. The function can be passed up to four arguments and return a value.

User defined functions can be called from any script in the Movie by using the Function Action.

Functions can (and should) be used to improve the modularity and readability of the code. Sections of code that are often re-used should be placed into a function.

4.6.5.1.4 Evaluate Button

A large number of the commands described below make use of the evaluate button (e) in their scripting assist panel.
This button allows the user to toggle between a constant text entry or a value that is contained in a variable.

Example: a function fn() could take a string constant "my text" as a parameter, or a variable containing the required string as a parameter.

```
fn("my text");
```

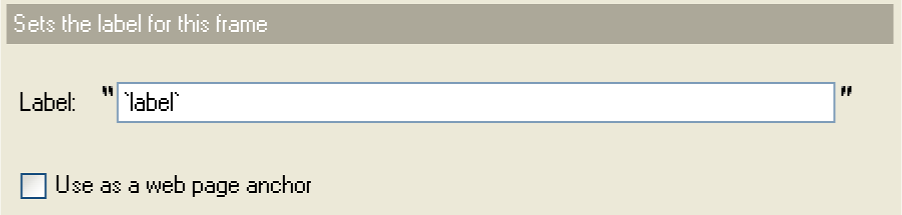
and

```
a = "my text";
fn(a);
```

would cause the same data to be passed to the function.

4.6.5.2 Set Label

Use this Action to label a Frame. Use the label to refer to the Frame by name in a Goto Frame or If Frame Loaded Action.



Sets the label for this frame

Label: "`label`"

□ Use as a web page anchor

The Set Label Action can ***only*** be used with a Frame Event. That is, it has to be applied to a particular Frame of a Scene or Movie Clip. As with the Preload Content Action, the Frame does not need to be played for the label to be set.

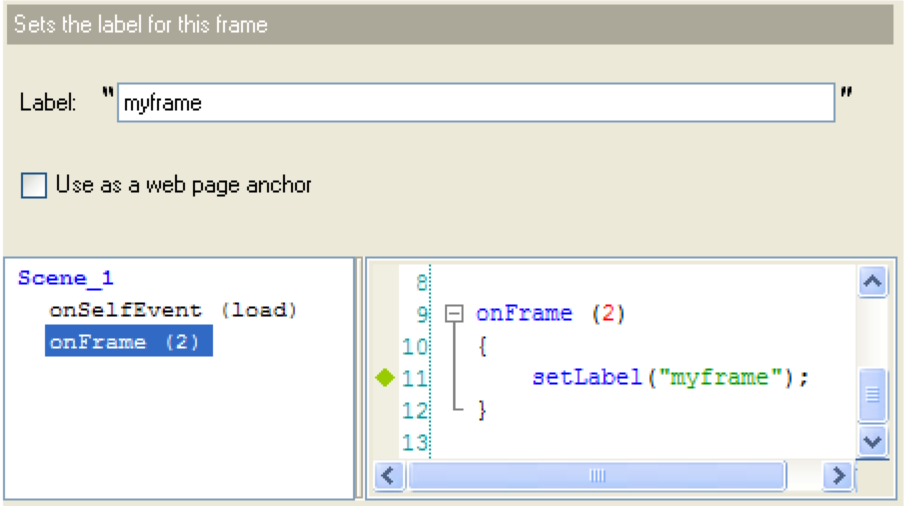
The advantage of using Frame labels is that Frames can be inserted or deleted without having to change any Goto Frame Actions.

Label
This is the name given to a particular Frame. Ensure labels are unique within a Movie Clip or the main Movie, even if they appear in different Scenes. The same label can be used in different Movie Clips.

Use as a web page anchor
Sets the label as a web page anchor in the Movie. Anchors simplify navigation in Flash Movies by letting viewers use the 'Forward' and 'Back' buttons in a browser to jump from frame to frame or scene to scene. Anchors only work with Flash Player 6.

Note: There is a known 'bug' in how Flash Player 6 when using Anchors. When jumping with Goto Frame, the browser history records ALL anchors from the start of the Movie to the destination Frame. Therefore, the 'Back' button will go to each anchor in the browser history rather than straight to the previous one

In the example below, Frame 2 has the label "myframe". A Goto Frame Action used elsewhere can jump to the Frame labeled "myframe". The use of labels allows the script to function correctly if additional frames were inserted before frame 2.



Sets the label for this frame

Label: "myframe"

□ Use as a web page anchor

```
Scene_1
    onSelfEvent (load)
    onFrame (2)
```

```
onFrame (2)
{
    setLabel("myframe");
}
```

4.6.5.3 Preload Content

This Action is used to mark the associated frame as the time when "Preload Content" should be loaded.

Any objects that have their preload option set to "At Preload Frame" will be loaded at this stage. The preload option can be found in the Export Panel of the Object.

Use this Action to instructs the SWiSH Max Movie to download (or define) a piece of content *before* it is used, for example to make sure a sound is ready to be played before a Play Sound Action, or an image has been downloaded before it is displayed.

To preload an object immediately before the Frame with the Preload Content Action, the preload option of the object has to be set as 'At Preload action'. The preload option can be found in the Export Panel, the properties dialog box of an image fill style and the sound options dialog box.

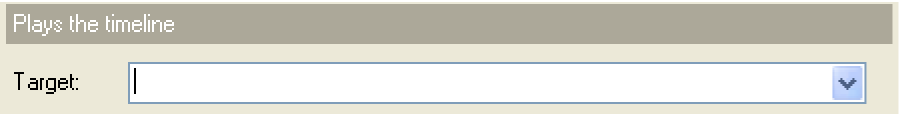
Note:

- There can be more than one Preload Content Action in a Movie. The Object to be preloaded will be loaded at the closest Frame with the Preload Content Action before its first use. For example, if the Object to be preloaded is first used at Frame 18 and two Preload Content Actions are at Frame 10 and Frame 20, the Object will be loaded at Frame 10
- The Preload Content action is *not required* to use content. For example, if a sound has not been preloaded when a Play Sound Action is encountered, the sound will be automatically downloaded

There are no options for the Preload Content Action.

4.6.5.4 Play

Use this Action to play a Movie or Movie Clip. If the Movie or Movie Clip is stopped, the Play Action will cause it to resume playing from the current Frame. If the Movie or Movie Clip is already playing, then the Play Action will have no effect.



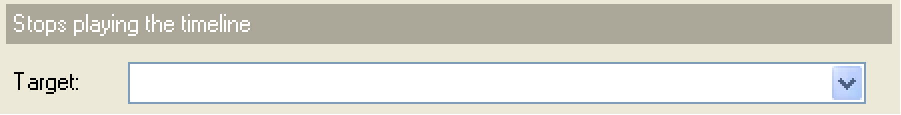
Plays the timeline

Target:

Target
The target Movie or Movie Clip to play.

4.6.5.5 Stop

Use this Action to stop a Movie or Movie Clip. If the Movie or Movie Clip is playing, the Stop Action will cause it to stop. If the Movie or Movie Clip is already stopped, then the Stop Action will have no effect.



Stops playing the timeline

Target:

Target
The target Movie or Movie Clip to stop.

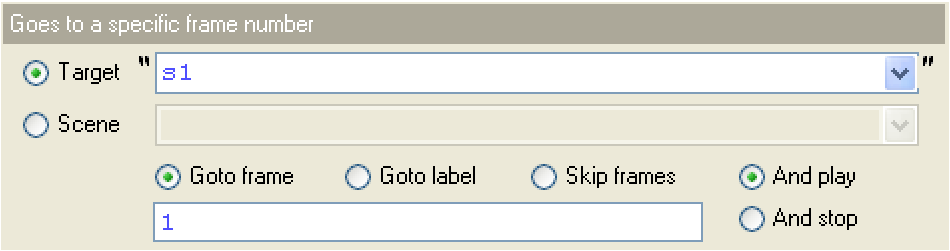
4.6.5.6 Goto And Play

This section discusses common goto and play Actions:

- gotoAndPlay(FRAME)
- gotoAndPlay(LABEL)
- nextFrameAndPlay()
- prevFrameAndPlay()
- nextSceneAndPlay()
- prevSceneAndPlay()
- gotoAndStop(FRAME)
- gotoAndStop(LABEL)
- nextFrameAndStop()
- prevFrameAndStop()
- nextSceneAndStop()

- prevSceneAndStop()

Use this Action to go to another Frame, Label, or Scene. The Frame or Label can be in the same Scene or another Scene



Goes to a specific frame number

● Target "s1"

○ Scene

● Goto frame ○ Goto label ○ Skip frames ● And play

1 ○ And stop

When the Goto And Play Action is used with a Frame Event, the Goto And Play happens before the Frame containing the Action is displayed.

Note: A Goto And Play Action will have no effect until the Frame it is to go to is loaded

Target
The target Movie or Movie Clip that the action applies to.
It is possible to start the Movie in Movie Clip s1 playing from Frame 10 via the command: _root.s1.gotoAndPlay(10); in which case, _root.s1 would be the target.
If this radio button is selected, then the Scene field is disabled as scenes only apply to the _root movie.

Scene
The Scene radio button only appears when the Goto frame option is selected. If the Goto Frame Action is targeting a Movie Clip, then a Scene can not be selected, as Movie Clips do not have Scenes within them.

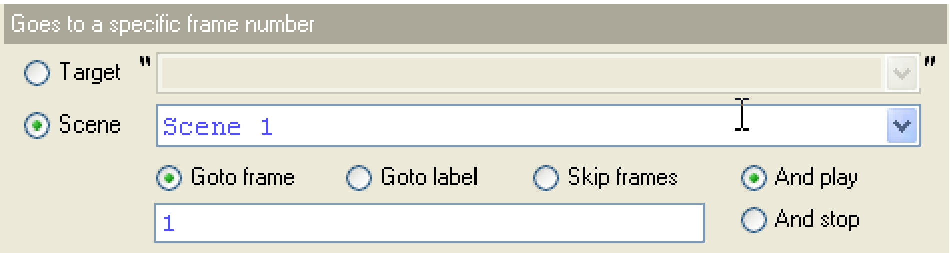
If this radio button is selected then the scene that the frame applies can be selected from the drop down menu. The available scenes are:
<first scene>
<last scene>
<next scene>
Scene_name

If this radio button is selected then the target field is disabled as the _root target is implied.

Specify the Frame to go to in one of three different ways:

- by Frame number within a Scene
- by Frame label
- skip a number of Frames.

Frame



Goes to a specific frame number

○ Target ""

● Scene Scene 1

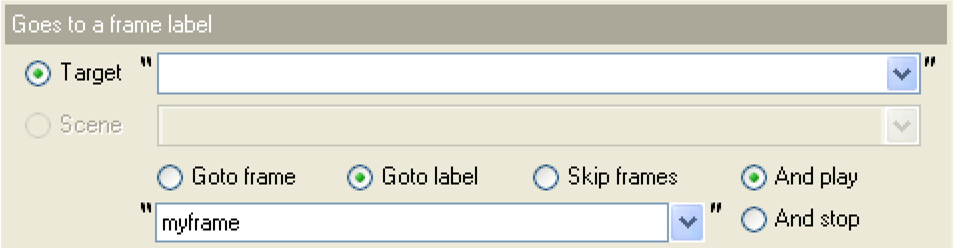
● Goto frame ○ Goto label ○ Skip frames ● And play

1 ○ And stop

Select the Scene or Movie Clip containing the Frame to go to. Enter the Frame number (starting from '1') in the Script Panel. Select the Scene directly by name, or use one of the five special Scene names:

- <current scene>
- <first scene>
- <last scene>
- <previous scene>
- <next scene>

Label



Goes to a frame label

● Target " "

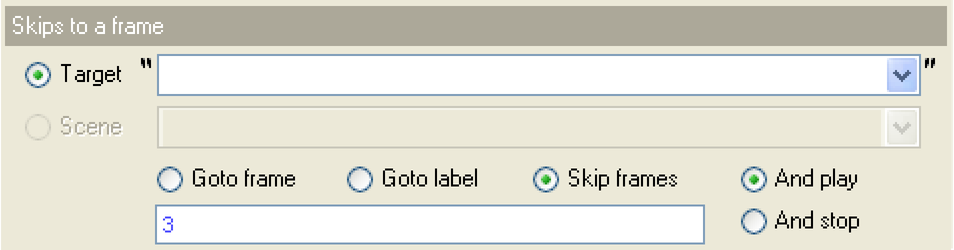
○ Scene

○ Goto frame ● Goto label ○ Skip frames ● And play

"myframe" ○ And stop

Select the label of the Frame to go to. The Frame can be anywhere in the current Movie or Movie Clip. As a frame label implies a specific scene, the Scene radio button is disabled. See the Set Label Action.

Skip



Skips to a frame

● Target " "

○ Scene

○ Goto frame ○ Goto label ● Skip frames ● And play

3 ○ And stop

Specifies a positive or negative number of Frames to skip. For example, if set to 1 the next the Movie will go to the next Frame,
while if set to -1 the Movie will go to the previous Frame. As skipping implies the current location, the Scene radio button is disabled.

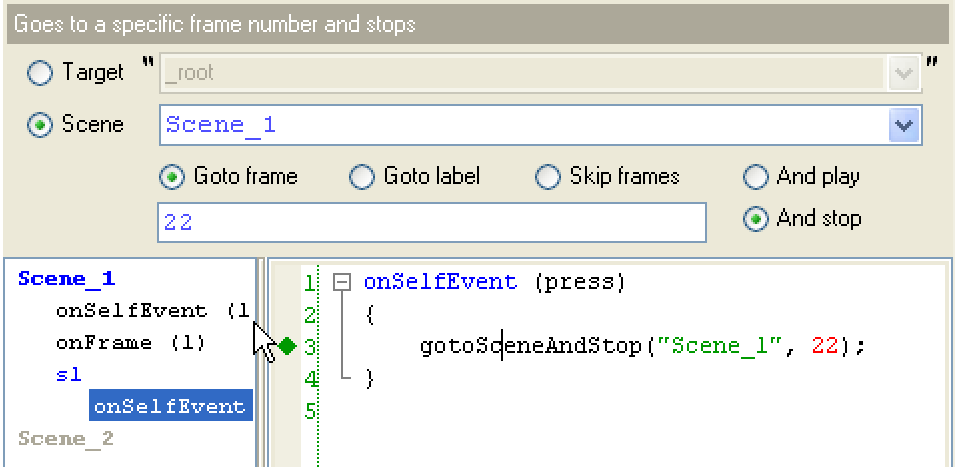
Play

When the Play option is selected (the default setting), the Movie or Movie Clip will continue playing from the Frame specified.

Stop

When the Stop option is selected, the Movie or Movie Clip will display the Frame specified and then stop.

In the example below, the Movie will jump to Frame 22 of "Scene_1" when the mouse button is pressed inside the Object s1. The Movie will stop playing at the Frame specified.



Goes to a specific frame number and stops

○ Target " _root "

● Scene Scene_1

● Goto frame ○ Goto label ○ Skip frames ○ And play

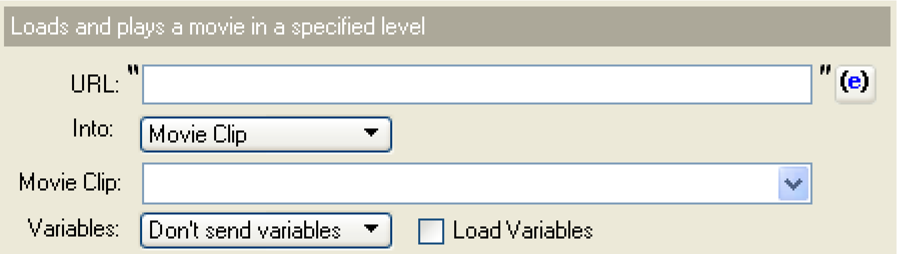
22 ● And stop

Scene_1
onSelfEvent (1
onFrame (1)
s1
onSelfEvent
Scene_2

```
onSelfEvent (press)
{
    gotoSceneAndStop("Scene_1", 22);
}
```

4.6.5.7 Load Movie

Use this Action to load another .swf file into the current Flash Player.



Loads and plays a movie in a specified level

URL: " " (e)

Into: Movie Clip

Movie Clip:

Variables: Don't send variables □ Load Variables

When a .swf file is ***loaded***, the [0,0] coordinate of the loaded .swf file is aligned with the [0,0] coordinate of the Movie or Movie Clip into which is it loaded. The [0,0] coordinate of a .swf file is usually the top-left corner. The [0,0] coordinate inside a Movie Clip is usually the center. This means if a .swf file is loaded into a level, the top-left corner of the loaded .swf file will be aligned with the top-left corner of the Movie. If a .swf file is loaded into a Movie Clip, then the top-left corner of the loaded .swf file will be aligned with the *center* of the Movie Clip.

If a .swf file is ***exported*** from SWiSH Max, to specify that the [0,0] coordinate of the Movie are in the center, rather than the top left, change the "Offset to suit use as a Movie Clip" option in the Movie Export Panel.

URL

Specify an absolute or relative URL of the .swf file to load. If an absolute URL is not specified for the .swf file, the Flash Player will assume the relative URLs are based on the URL for the currently open .swf file. When running in a browser, the base URL can be overridden using the 'Base' option on the HTML Export Panel. When playing within SWiSH Max, or using the File | Test | SWF In Player or File | Test | HTML + SWF In Browser commands, the default folder is the one selected in the Tools | Preferences dialog box. To test the Load Movie Action, ensure the .swf files can be found in the selected folder.

Note: When used in external Flash Player 6+ player, browser or projector .exe, use loadMovie to load either a SWF file of a non-progressive JPEG file.

Load Into...

Movie Clip

Loads the .swf file into the current Movie Clip or Scene. The .swf file replaces the previous content of the Movie Clip, however, the position and size of the Movie Clip, and any Effects applied to it will apply to the loaded .swf instead.

Level
Specifies the level to load the .swf file into. Movies loaded into higher levels appear in front of those in lower levels. The Movie that was loaded first is always loaded at the bottom level, level 0. The Movie at level 0 sets the Frame Rate, background color, and Frame size for all other loaded Movies. Movies are then stacked in numerically higher levels above the Movie at level 0. Replace a Movie at a given level by simply loading another Movie into that level. Movies can be 'Chained' together by loading another Movie over the current Movie at level 0.

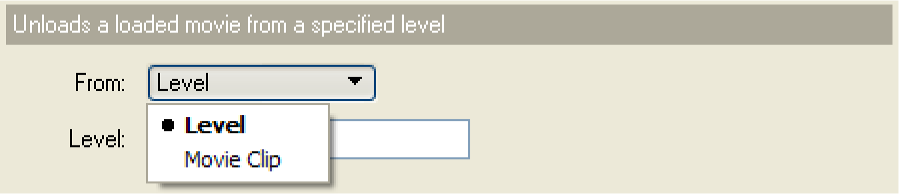
Variables
Specifies the HTTP method of sending variables to the Movie to be loaded.
Options are:

- **Don't send variables**: no variables are sent
- **Send using get**: the GET method appends the variables to the end of the URL, and is used for small numbers of variables
- **Send using post**: the POST method sends the variables in a separate HTTP header and is used for long strings of variables

Load Variables Only
Reads variables from the URL specified, either from a text file or a script such as CGI, ASP, PHP or Perl scripts. The content of the Movie Clip or Level specified are not replaced, only the value of the variables.

4.6.5.8 Unload Movie

Use this Action to unload a .swf file that was previously loaded in a specified level.



Unloads a loaded movie from a specified level

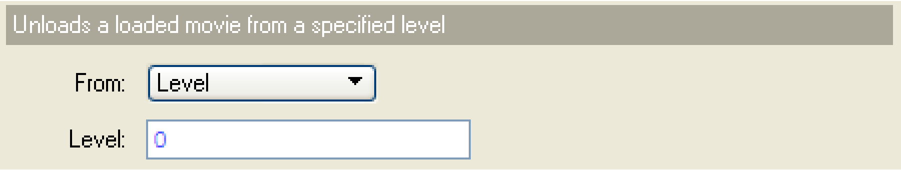
From: Level

Level:

● Level
Movie Clip

Source
This variable determines whether the Movie should be unloaded from a particular level or Movie Clip. See the Load Movie Action for more information.

Level (unloadMovieNum)
Specifies the Movie level to unloaded. Any .swf Movie playing in that level is cleared.

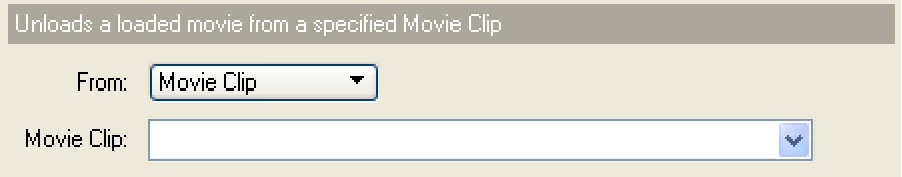


Unloads a loaded movie from a specified level

From: Level

Level: 0

Movie Clip (unloadMovie)
Specifies the Movie Clip to unloaded. Any .swf Movie playing within that Movie Clip is cleared.



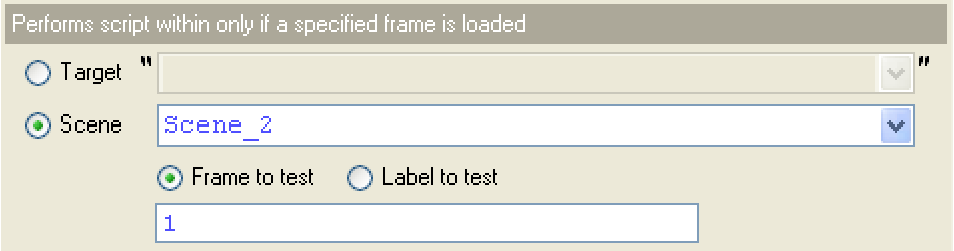
Unloads a loaded movie from a specified Movie Clip

From: Movie Clip

Movie Clip:

4.6.5.9 If Scene Frame Loaded

Use this Action to perform Actions only if a given Frame has been loaded. This can be useful when creating a preloader.



Performs script within only if a specified frame is loaded

○ Target " "

● Scene Scene_2

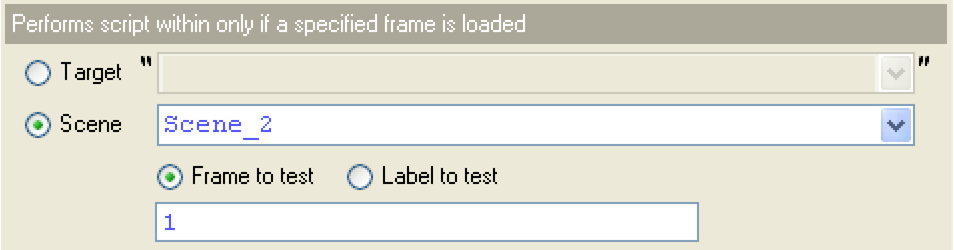
● Frame to test ○ Label to test

1

You can specify the Frame you want to test for in one of two ways: by Frame number within a Scene, or by Frame label.

Target
The target Movie, Movie Clip or Object that the Action applies to. Note that the target name must use slash notation or the _target property.

Frame



Performs script within only if a specified frame is loaded

○ Target " "

● Scene Scene_2

● Frame to test ○ Label to test

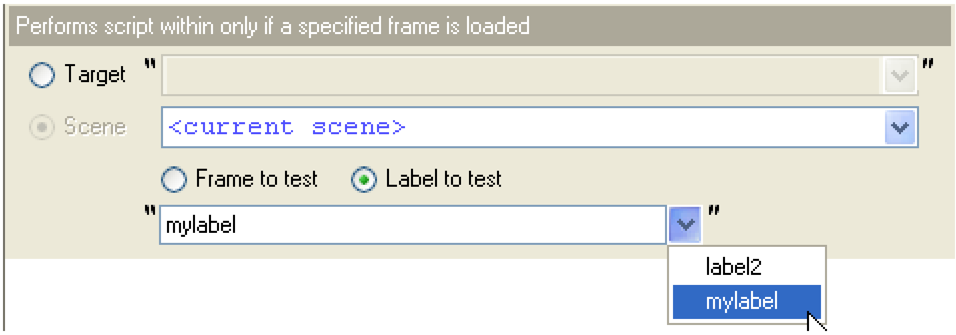
1

Select the Scene containing the Frame you want to go to. Enter the Frame number (starting from zero) in the 'Action' Panel. You can select the Scene directly by name, or use one of the five special Scene names:

- <current scene>
- <first scene>
- <last scene>
- <previous scene>
- <next scene>

Scene
The Scene selection only appears when you select the Frame option (see above).

Label to test


Performs script within only if a specified frame is loaded

○ Target " "

● Scene <current scene>

○ Frame to test ● Label to test

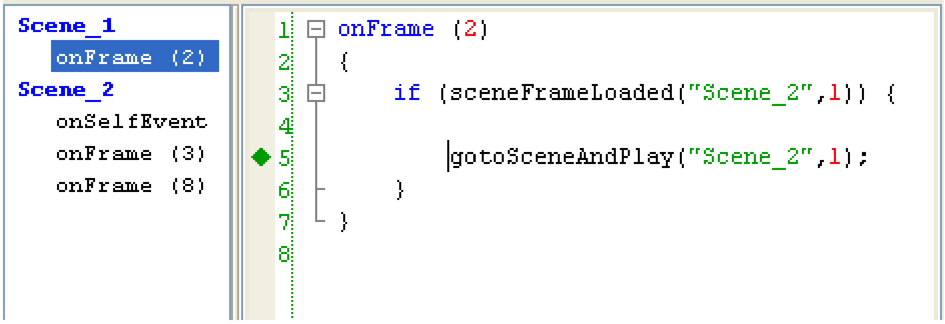
"mylabel"

Specifies the frame based on a defined label. A list of available labels is shown by pressing the associated down arrow button. In this case mylabel and label2 are available in the current scene.

In the example below, the Movie is instructed to jump to Frame 1 of Scene_2, only when Frame 1 of Scene_2 has been downloaded.

Note: In this case, an If Frame Loaded Action is not strictly necessary, as the Goto And Play Action will not do anything if the Frame it is to go to is not yet loaded

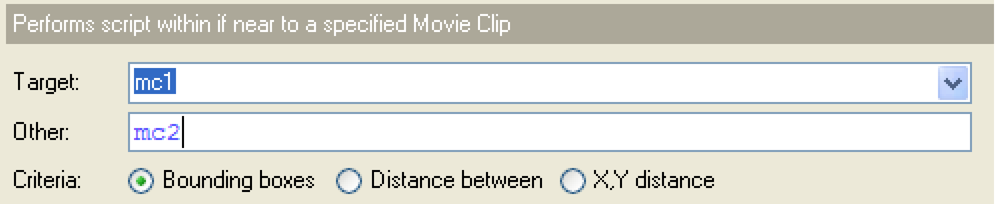
Note: Due to a bug in Flash Player, even if a movie is fully downloaded and in your temporary internet files cache, it will NOT be fully loaded by the time frame 1 of the movie plays. Flash Player will only load approximately 8K of your movie by the time the first frame plays. For this reason, if you are testing for frame loaded in order to make a preloader, start your preloader loop on frame 2 of the movie so that you can correctly bypass the preloader if the movie is already downloaded.



```
onFrame (2)
{
    if (sceneFrameLoaded("Scene_2",1)) {

        gotoSceneAndPlay("Scene_2",1);
    }
}
```

4.6.5.10 If Is Near This

Use this Action to perform Actions if the current Movie Clip is near another Movie Clip.


Performs script within if near to a specified Movie Clip

Target: mc1

Other: mc2

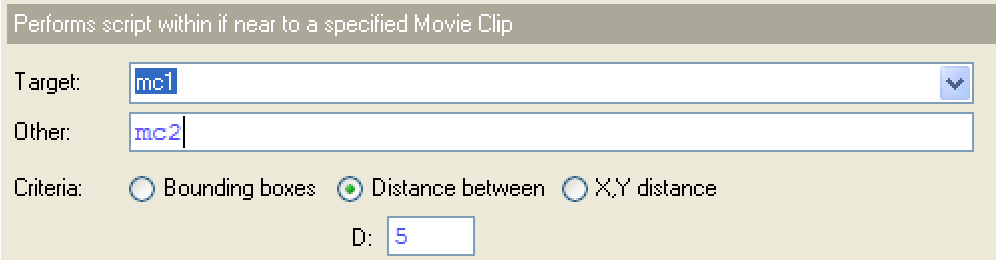
Criteria: ● Bounding boxes ○ Distance between ○ X,Y distance

Target
The target Movie or Movie Clip that we are testing for proximity.

Other
The target Movie or Movie Clip that the main target's location will be tested against.

Bounding boxes hit
The Targets are considered to be 'near' if the bounding boxes of both targets overlap.

Distance between
Select the allowable distance in the X direction for the other Movie Clip to be considered near.


Performs script within if near to a specified Movie Clip

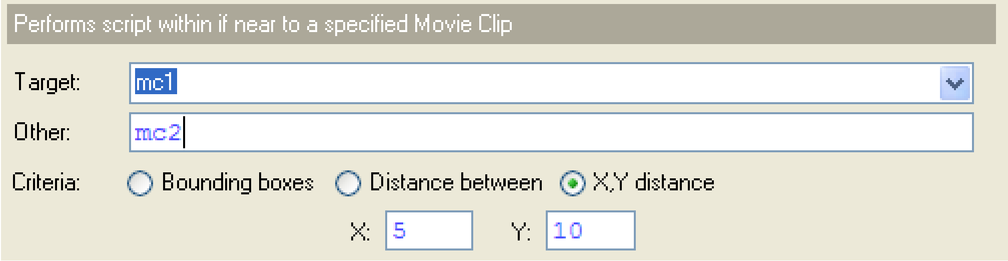
Target: mc1

Other: mc2

Criteria: ○ Bounding boxes ● Distance between ○ X,Y distance

D: 5

X, Y distance
Select the allowable distance in the X and Y directions for the other Movie Clip to be considered near.


Performs script within if near to a specified Movie Clip

Target: mc1

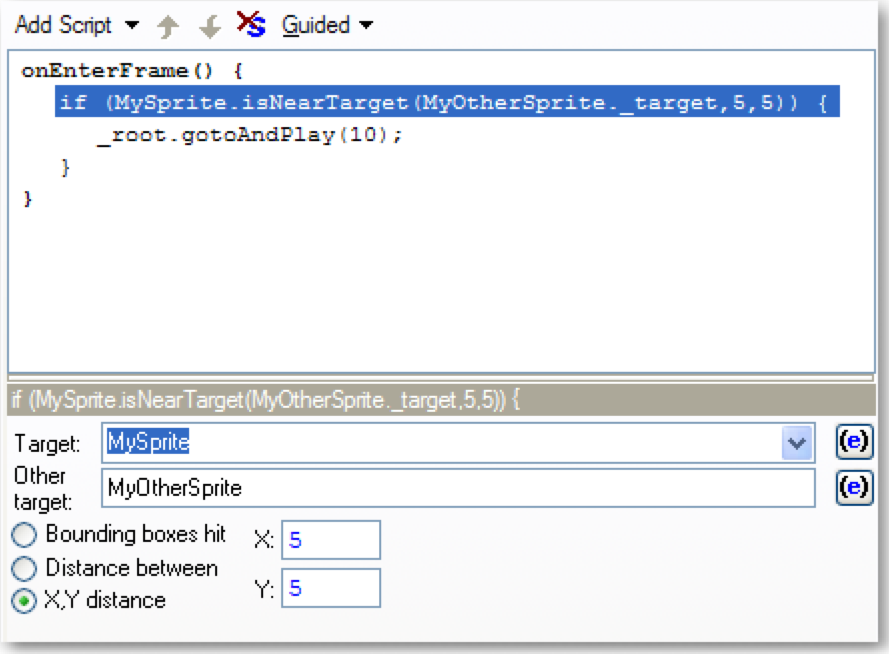
Other: mc2

Criteria: ○ Bounding boxes ○ Distance between ● X,Y distance

X: 5 Y: 10

Note: The distance is taken from the X and Y origin coordinate of the Movie Clips - this is usually the center of the Movie Clip; however, the origin (or Anchor point) can be changed in the 'Transform' panel.

In the example below, the root timeline will jump to Frame 10 if the origin of the Movie Clip called "MyMovie Clip" is within 5 pixels of the origin of the Movie Clip called "MyOtherMovie Clip".



```
onEnterFrame() {
    if (MySprite.isNearTarget(MyOtherSprite._target,5,5)) {
        _root.gotoAndPlay(10);
    }
}
```

if (MySprite.isNearTarget(MyOtherSprite._target,5,5)) {

Target: MySprite

Other target: MyOtherSprite

○ Bounding boxes hit

○ Distance between

● X,Y distance

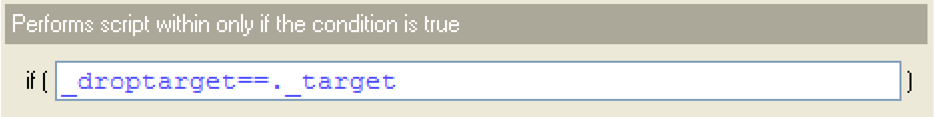
X: 5

Y: 5

This Action can contain the other Conditional Actions including If, While and Else.

4.6.5.11 If Drop Target

Use this Action to perform Actions if the current Movie Clip is dropped on another Movie Clip when dragging (see the Start Drag and Stop Drag actions)



Performs script within only if the condition is true

if (_droptarget==._target)

It is necessary to modify the if statement so that the .target side represents the target name.

eg. _droptarget==_root.MyClip._target

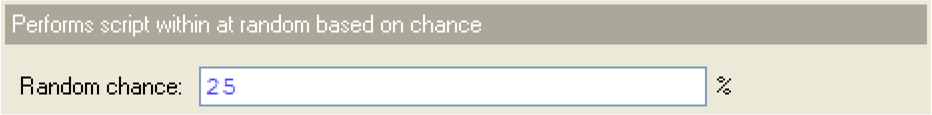
the script below will cause the Movie Clip to jump to Frame 10 if dropped on the Movie Clip called MyClip.

```
onrelease()
{
    if(_droptarget==root.MyClip._target)
    {
        _root.MyClip.gotoAndPlay(10);
    }
}
```

The Action after the if statement can contain the other Conditional Actions including If, While and Else.

4.6.5.12 If Chance

Use this Action to perform Actions at random. Specify the chance of the Actions being taken as a percentage probability.



Performs script within at random based on chance

Random chance: 25 %

Random Chance

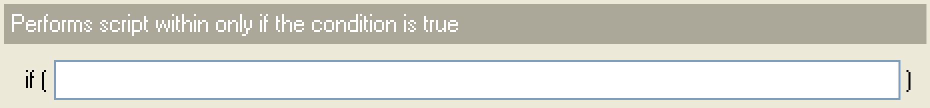
Select the chance that the Actions should execute as a percentage probability.

eg. enter 25 and the actions within the if statement will be executed, on average, 25% of the time.

The Action within the if statement can contain the other Conditional Actions including If, While and Else.

4.6.5.13 If

The IF action is used to test an expression and determine if it evaluates to true or false. If the expression evaluates to true, then a subsequent statement or set of statements are executed. If the expression is false, the statements are ignored.



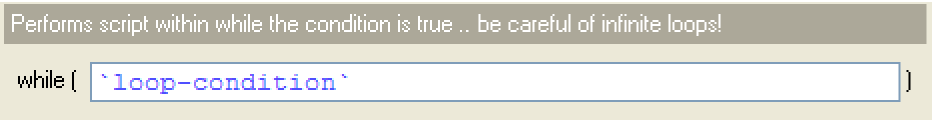
Performs script within only if the condition is true

if ( )

4.6.5.14 While

The While action begins a loop that evaluates a condition and executes the subsequent statement or set of statements while the condition evaluates to true. When the condition evaluates to false, the statements are ignored and the loop is aborted.

Edit the default text 'loop-condition' to be the condition to evaluate.



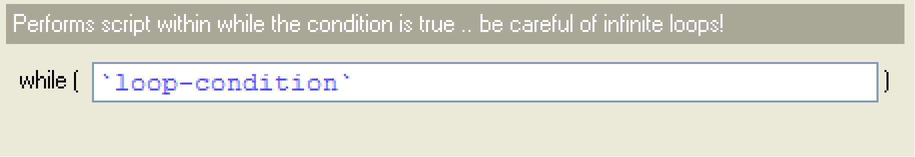
Performs script within while the condition is true .. be careful of infinite loops!

while (`loop-condition`)

To test the condition at the end of the loop use the Do...While loop instead. This loop will execute the statement or set of statements at least once before testing the if the condition is true or false.

4.6.5.15 Do While

The Do...While action is a conditional loop. Just like the While action, this action will evaluate a condition and execute a statement or set of statements if the condition equates to true. Unlike the While action, the Do..While action will execute the statement(s) at least once before it evaluates the condition.



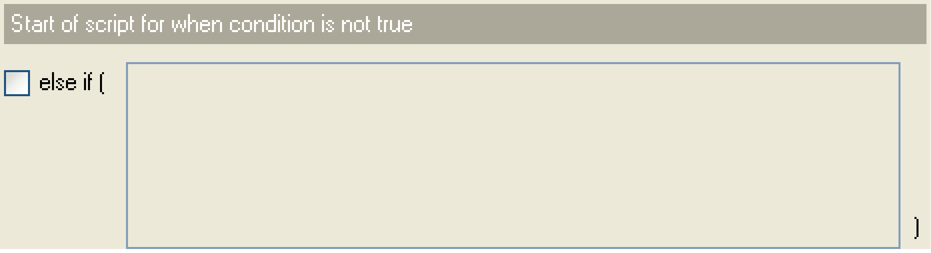
Performs script within while the condition is true .. be careful of infinite loops!

while (`loop-condition`)

Edit the while expression to be the condition that you wish to evaluate.

4.6.5.16 Else

The Else action is used to execute an alternate statement or set of statements if the intial IF condition equates to false.



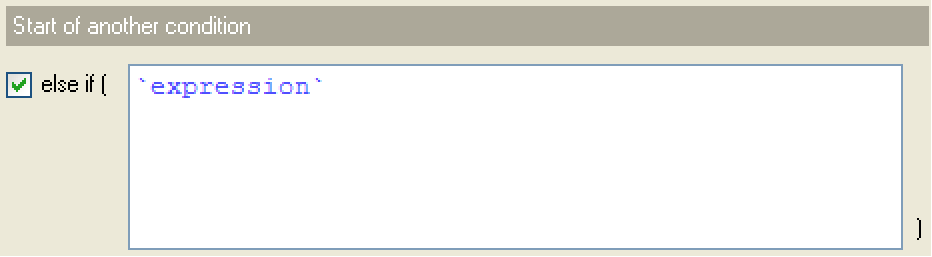
Start of script for when condition is not true

□ else if ( )

Else If
This option converts an Else action into an Else If action, which is used to evaluate another condition if the initial IF condition equates to false. Enter the condition of the if statement into the dialog box.

4.6.5.17 Else If

The Else If action is used to evaluate another condition if the initial IF condition equates to false.



Start of another condition

☑ else if (`expression`)

Edit the field to replace 'expression' with the required conditional statement.

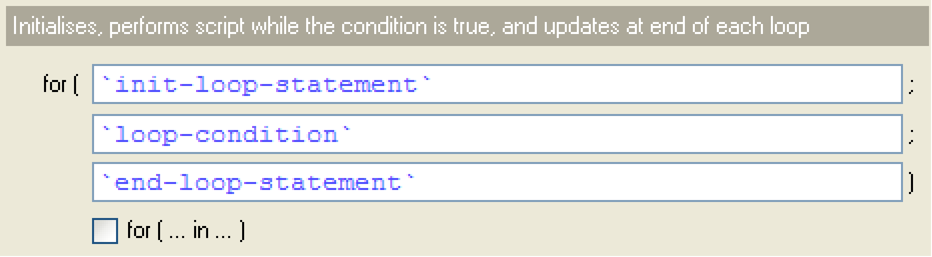
Else If
If this option is deselected, the condition is ignored and the subsequent statement or set of statements will be executed if the initial condition equates to false.

4.6.5.18 For

The For action is a conditional loop statement. The For loop consists of three parts:

- an initial condition or argument is defined
- the expression that will be evaluated is defined
- a statement that will be executed at the end of the loop is defined (this is typically used to increase the initial condition).

The expression is tested and if it evaluates to true, the subsequent statement or set of statements will be executed. After the statement(s) are executed, the statement defined for the end of the loop will be executed and the loop tests the condition again. If, at any time, the condition evaluates to false, then the loop is aborted.



Initialises, performs script while the condition is true, and updates at end of each loop

```
for ( `init-loop-statement` ;
      `loop-condition` ;
      `end-loop-statement` )
```

□ for (... in ...)

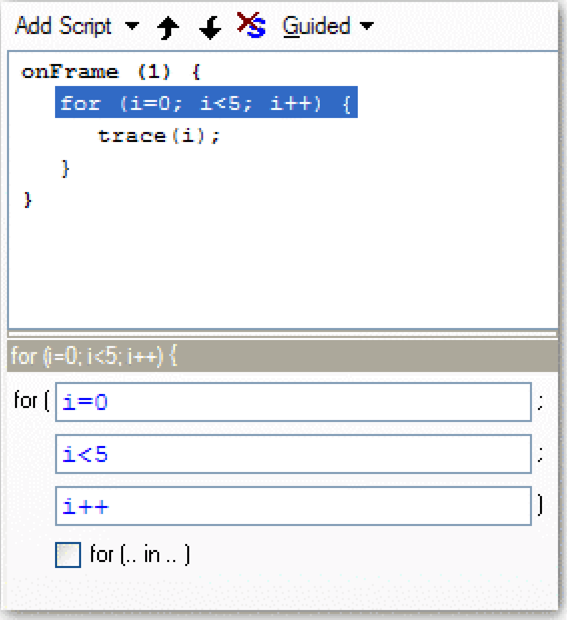
Edit the 'init-loop-statement', 'loop-condition' and 'end-loop-statement' to your requirements.

Note that the fields above are all optional. If the loop-condition statement is omitted, the loop will loop continuously.

For (.. in ..)
Makes this a For..In loop.

Example

```
for (i = 0; i <= 5 ; i++) {
    trace(i);
}
// displays "0,1,2,3,4" in the 'Debug' window.
```



```
onFrame (1) {
    for (i=0; i<5; i++) {
        trace(i);
    }
}
```

for (i=0; i<5; i++) {

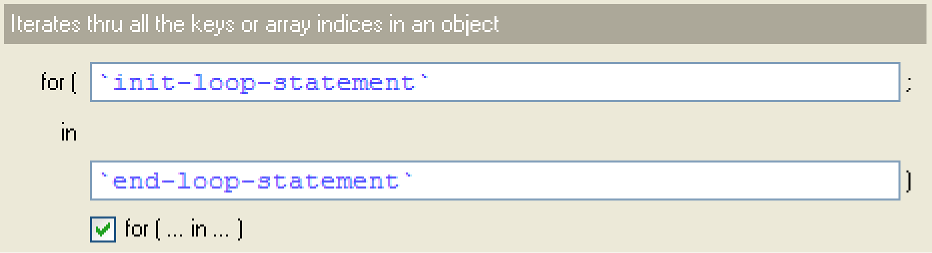
for (i=0 ;
i<5 ;
i++)
☐ for (.. in ..)

The initial condition is defined as "i=0". Next, the expression "i<5" is tested and evaluates to true ("i" is indeed less than 5 at this point in the loop). Next, the statement "trace(i)" is executed - followed by the end statement "i++" (which increases the value of "i" by one).

At this point in the loop i=1, and the expression "i<5" is once again tested. This process will continue until "i" is no longer less than 5.

4.6.5.19 For .. In

For..In is a looping action. Its purpose is to cycle through the individual properties of an Object (such as a Movie Clip) or through the individual elements within an Array and execute a statement or set of statements for each.



Iterates thru all the keys or array indices in an object

for (`init-loop-statement` ;
in
`end-loop-statement`)
☑ for (... in ...)

For (.. in ..)

Deselecting this option will convert a For..In loop to a For loop.

Example
The following example uses the for in loop to cycle through each of the properties that exist within the user defined object 'details'.

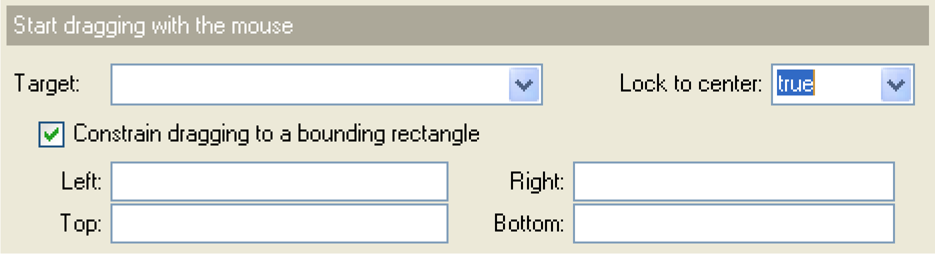
```
    var details:Object = {nameFirst:"Andrew", nameLast:"Wilson", phone:"555 1234"};

    for (var objProperty in details)
    {
        trace("details." add objProperty add " = " add details[objProperty]);
    }
```

This is the output from the trace statement:

```
details.phone = 555 1234
details.nameLast = Wilson
details.nameFirst = Andrew
```

4.6.5.20 Start Drag

Use this Action to start dragging a Movie Clip with the mouse. To stop dragging, use the Stop Drag Action. This Action is usually placed in a Press Event.



Start dragging with the mouse

Target:

Lock to center: true

☑ Constrain dragging to a bounding rectangle

Left:

Right:

Top:

Bottom:

Target
The target to apply the Start Drag Action to.

Lock to center:
When locked, the Movie Clip center position is locked to the mouse pointer. If not locked, the Movie Clip will retain its position with respect to the mouse at the time of the initial mouse click.

Constrain dragging to a bounding rectangle
When constrained, the Movie Clip stays within the specified rectangle.

Note: If a Press or Release Event is placed on the Movie Clip itself, it will be exported as a button and no longer work as a Movie Clip. Also the dragging Actions will then apply to the Movie Clip or Movie that contain the Movie Clip (rather than the Movie Clip itself). Instead, put the mouse Events on a button or Object *inside* the Movie Clip

4.6.5.21 Stop Drag

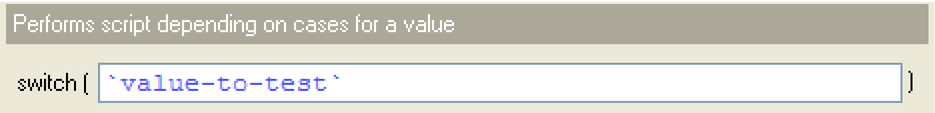
Use this Action to finished dragging a Movie Clip using the Start Drag Action. This Action is usually placed in a Release Event.

There are no options for the Stop Drag Action.

Note: If a Press or Release Event is placed on the Movie Clip itself, it will be exported as a button an no longer work as a Movie Clip. Also the dragging Actions will then apply to the Movie Clip or Movie that contain the Movie Clip (rather than the Movie Clip itself). Instead, put the mouse Events on a button or Object *inside* the Movie Clip

4.6.5.22 Switch

Use this Action to mark the start of an alternative set of Actions to perform if one of the specified cases is true (or if the default case is specified if no other case is true).



Performs script depending on cases for a value

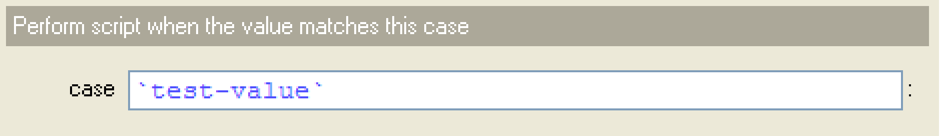
switch (`value-to-test`)

Edit 'value-to-test' with the name of the variable that will hold the test value.

See Switch in the Script Reference for more details.

4.6.5.23 Case

The Case action is a condition used within a Switch action. A statement or set of statements will be executed if this case is the same as the argument provided within the Switch action. A Case can be compared to the IF conditional action. A single Case would be the same as using a single IF conditional, and multiple Cases would be the same as adding multiple Else IF conditional actions.



Perform script when the value matches this case

case `test-value` :

Edit 'test-value' to be the value to be tested.

Example:

```
Function testSwitch() {
    switch (i) {
        Case 1:
        trace("i is equal to 1");
        break;
        Case 2:
        trace("i is equal to 2");
        break;
        Case 3:
        trace("i is equal to 3");
        break;
        default:
        trace("i is not 1, 2 or 3");
        break;
    }
}
onFrame (1) {
    i = 2;
    testSwitch();
}
// displays "i is equal to 2" In the Debug window.
```

This could be compared to:

```
if (i == 1) {
    trace("i is equal to 1");
} else if (i == 2) {
    trace("i is equal to 2");
} else if (i == 3) {
    trace("i is equal to 3");
} else {
    trace("i is not 1, 2 or 3");
}
```

4.6.5.24 Default

The Default action defines a statement or set of statements if an argument can find no matches within a Switch conditional. It would be the same as using an Else action with an IF condtitional.

Example:

```
function testSwitch() {
    switch (i) {
        case 1:
        trace("i is equal to 1");
        break;
        case 2:
        trace("i is equal to 2");
        break;
        case 3:
        trace("i is equal to 3");
        break;
        default:
        trace("Cannot find a match for the variable 'i'");
    }
}
onFrame (1) {
    i = 4;
    testSwitch();
}
// Since the variable "i" is not equal to any of the defined Cases within the Switch action,
"Cannot find a match for the variable 'i'" will be displayed in the debug window.
```

This could compared to:

```
if (i == 1) {
    trace("i is equal to 1");
} else if (i == 2) {
    trace("i is equal to 2");
} else if (i == 3) {
    trace("i is equal to 3");
} else {
    trace("Cannot find a match for the variable 'i'");
}
```

4.6.5.25 Break

Break is used within a conditional loop or within a block of statements associated with a particular case in a switch action. When used within a loop, the break action tells the script to abort the loop and execute the statement immediately following the loop. When used within a switch action, the break action tells the script to skip the remaining statements in that case and jump to the next statement after the enclosing switch action.

There are no options for the Break action.

4.6.5.26 Continue

Continue is used in conditional loops and behaves differently based on which type of loop is used.

If continue is used within a while loop, it tells the script to skip the rest of the loop and return to the beginning of the loop - where the condition is tested.

If continue is used within a do..while loop, it tells the script to skip the rest of the loop and jump to the bottom of the loop, where the condition is tested.

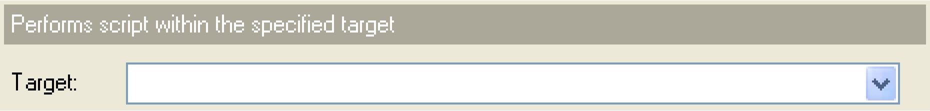
If continue is used within a for loop, it tells the script to skip the rest of the loop and jump to the evaluation of the loop's post-expression.

If continue is used within a for...in loop, it tells the script to skip the rest of the loop and jump to the top of the loop and continue with the next value in the enumeration.

There are no options for the Continue action.

4.6.5.27 tellTarget

Use this Action to direct all its child Actions to the named Movie Clip/Movie rather than the current Movie Clip/Movie.



Target
Defines the target the Actions apply to.

The tellTarget Action is useful when there are Movie Clips or loaded Movies at other levels, as it allows Movie Clips to do things to each other. It also lets the Movie Clip do things to the main Movie and vice versa.

Note: Movie Clips that are inside a Button State cannot act as targets, even if they have a name. Also Movie Clips that are within an unnamed Movie Clip or a Group can only be referred to from within the Movie Clip itself (i.e. from one of the Objects inside the Movie Clip)

Targets can be nested inside other targets. In this case, specify a target by either an absolute or relative target path, both slash and object synonym notation is supported:

- An ***absolute*** target path starts with a '/' or _root.
- A ***relative*** target path starts with either the name of a child Movie Clip or 'this.<child Movie Clip name>'. The parent Movie Clip (or Scene) can be referred to via '../' or '_parent.'.

For example, assume a Scene has the following Movie Clips:

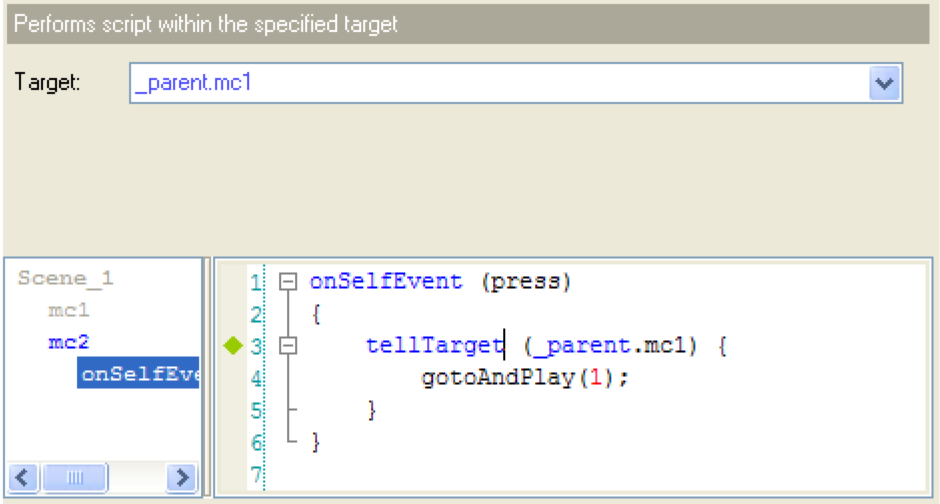
```
-Scene 1
        -Movie Clip Tom
                -Movie Clip Richard
        -Movie Clip Harry
```

Scene 1 contains two Movie Clips, 'Tom' and 'Harry'. Movie Clip Tom contains another Movie Clip, 'Richard'.

From within Movie Clip Tom, use the following to refer to the other Movie Clips and Scenes:

'/' and _root.	refers to Scene 1 (main Movie)
'../' and _parent.	refers to Scene 1 (parent of the current Movie Clip)
'../Harry' and _parent.Harry	refers to the sibling Movie Clip called Harry (that is, Harry is within the parent of Tom)
'/Harry' and _root.Harry	refers to the Movie Clip within the main Movie called Harry
'Richard' and this.Richard	refers to the child Movie Clip Richard
'/Tom/Richard' and _root.Tom.Richard	refers to the child Movie Clip Richard, which is a child of Tom, which is in the main Movie.

In the example below, the tellTarget Action will send "My Movie Clip" to Frame 1 when Scene_1 arrives at frame 10.



Performs script within the specified target

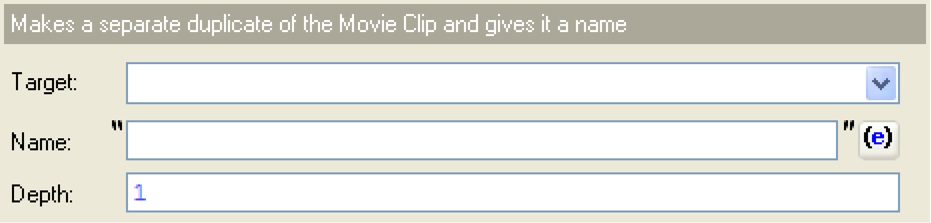
Target: _parent.mc1

```
onSelfEvent (press)
{
    tellTarget (_parent.mc1) {
        gotoAndPlay(1);
    }
}
```

This is identical to _parent.mc1.gotoAndPlay(1);

4.6.5.28 Duplicate Movie Clip

Use this Action to duplicate a Movie Clip in the Movie.



Makes a separate duplicate of the Movie Clip and gives it a name

Target:

Name: " "

Depth: 1

Target
The target Movie or Movie Clip to be duplicated.

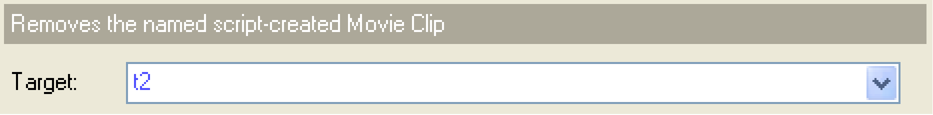
Name
The name of the new Movie Clip to be created.

Depth
Depth should be >= 0. Duplicate Movie Clips at higher depth appear in front of duplicate Movie Clips at lower depth. Replace a duplicate Movie Clip by specifying the same depth as another Movie Clip. All duplicate Movie Clips appear in front of all static Objects.

4.6.5.29 Remove Movie Clip

Use this Action to delete a Movie Clip created using the Duplicate Movie Clip Action.

Note: Only Movie Clips created with the Duplicate Movie Clip Action can be deleted using Remove Movie Clip



Removes the named script-created Movie Clip

Target: t2

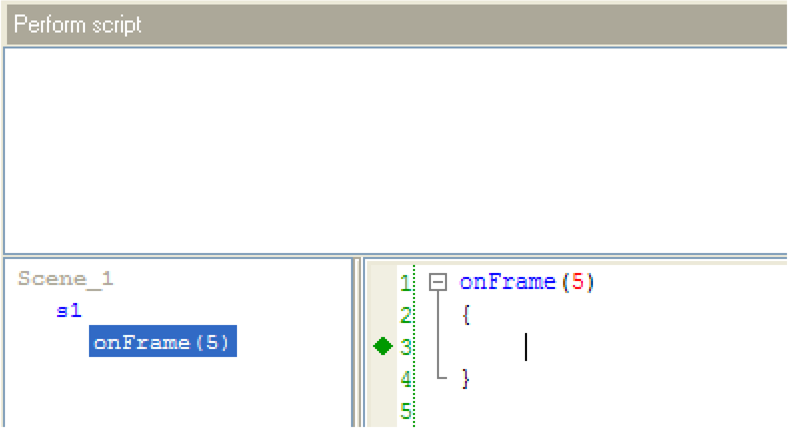
Target
The target Movie or Movie Clip to be deleted.

4.6.5.30 Evaluate

Use this Action when you evaluate an expression.

This can also be used to enter single lines of script without going through the full menu-pull-down procedure.

e.g. you can type in "this._X += 5" without using the Assign Action and its multiple parameters.



Perform script

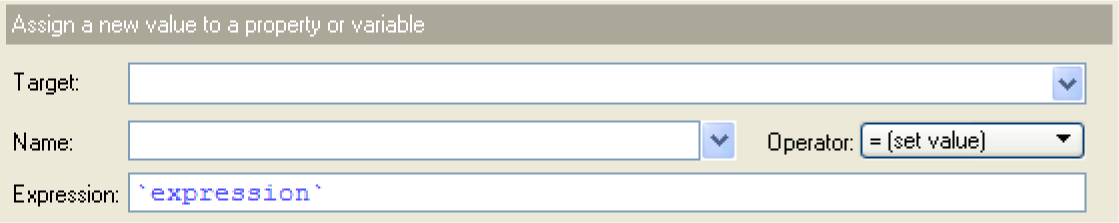
Scene_1
s1
onFrame (5)

```
onFrame (5)
{

}
```

However once the typed script is recognized, an appropriate panel will be displayed.

4.6.5.31 Name = Expr

Use this Action to change a properties of an Object, Movie Clip or the whole Movie.



Assign a new value to a property or variable

Target:

Name:

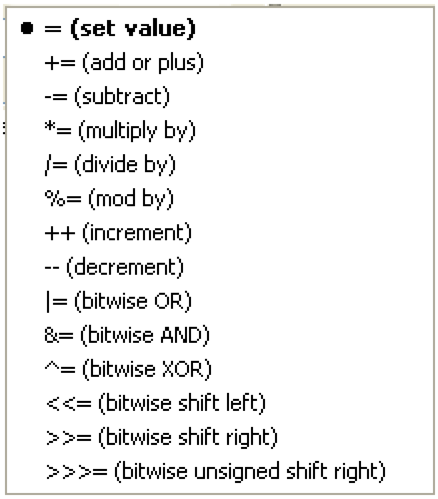
Operator: = (set value)

Expression: `expression`

Target
The target Movie or Movie Clip.

Name
Select the variable or property you wish to set.

Operator
For any of the properties or variables, you can either set it to a new value (=), add (+=) or subtract (-=) a value, or multiply (*=) or divide (/=) by a constant value or by the corresponding value of another Movie Clip.

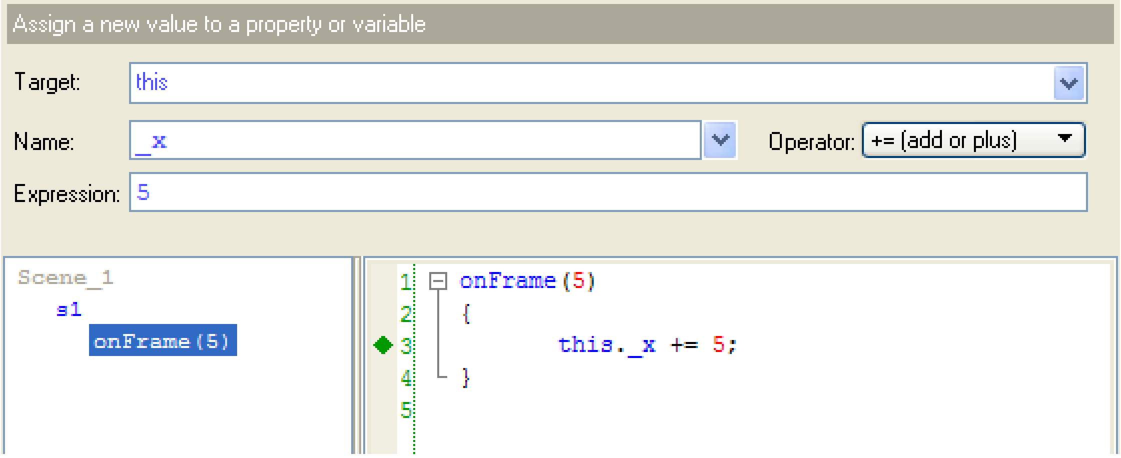


- ● **= (set value)**
- += (add or plus)
- -= (subtract)
- *= (multiply by)
- /= (divide by)
- %= (mod by)
- ++ (increment)
- -- (decrement)
- |= (bitwise OR)
- &= (bitwise AND)
- ^= (bitwise XOR)
- <<= (bitwise shift left)
- >>= (bitwise shift right)
- >>>= (bitwise unsigned shift right)

Note: for any operator `<op>`, `a <op>= b;` is identical to `a = a <op> b;`
eg. `a += 5; is idential to a = a+5;`

Value
The variable or property can be assigned any value or expression equating to a value.

In the following example, at Frame 5 the target Movie Clip "s1" will jump 5 pixels in the X direction.



Assign a new value to a property or variable

Target: this

Name: _x

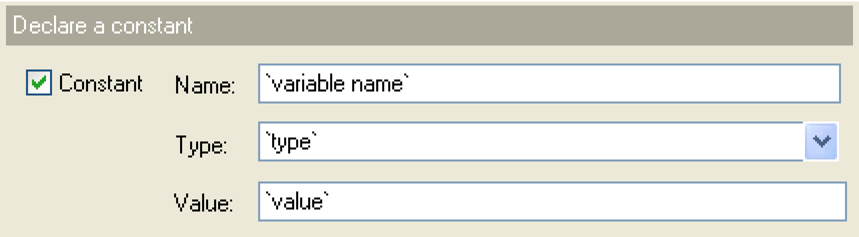
Operator: += (add or plus)

Expression: 5

```
onFrame(5)
{
        this._x += 5;
}
```

4.6.5.32 const,var

These actions are used to define local variables and assign initial values to those variables. Using the const action creates a variable that cannot be modified. The use of constants can improve code readability by giving significant constants a name.



Declare a constant

☑ Constant

Name: `variable name`

Type: `type`

Value: `value`

The above assist panel appears when const is selected. Un-checking the Constant checkbox changes the assist panel to the var assist panel. The var assist panel will insert the keyword 'var' into your script in place of 'const'.

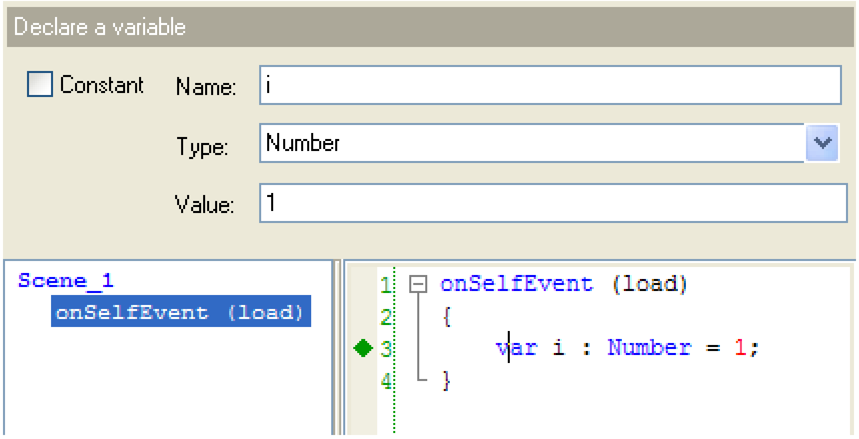
The following code is inserted into your script:

```
const `variable name` : `type` = `value`;
```

Name
This is the user defined name for the variable.

Type
Is optional and defines the data type of the variable. The pull down button shows the available standard data types.

Value
Defines the initial value of the constant or variable. This value is optional for variables.



Declare a variable

□ Constant

Name: i

Type: Number

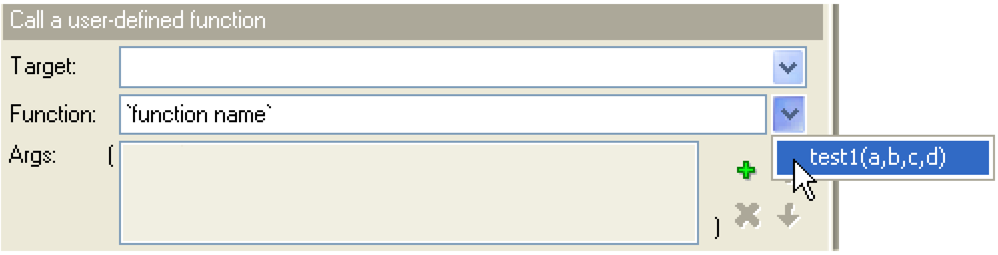
Value: 1

Scene_1
onSelfEvent (load)

```
onSelfEvent (load)
{
    var i : Number = 1;
}
```

4.6.5.33 Function

Use this Action when you want to call a User Defined Function.



Call a user-defined function

Target:

Function: `function name`

Args: ( )

test1(a,b,c,d)

Target
Select the Movie Clip or Movie who's function you wish to call.

Function
The function to call. A list of available functions can be seen by pressing the arrow button.

Arguments
The arguments to be used by the function.

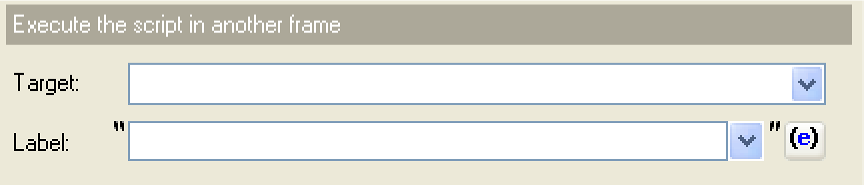
Use the ✚ button to add arguments, the ✖ button to delete them and the ⇧ and ⇩ to move the arguments within the list.

See Also

Return, User Defined Functions

4.6.5.34 Call

Allows you to Call the Actions in a labeled frame of the nominated clip. Lets you put a library of Action script in a Movie Clip and call it from elsewhere.



Execute the script in another frame

Target:

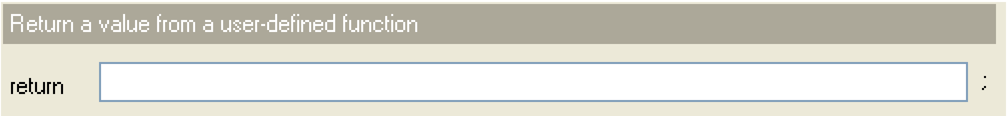
Label: " " (e)

Target
The target Movie or Movie Clip to be called.

Label
Select the label of the Frame you want to call. The Frame can be anywhere in the current Movie or Movie Clip. You do not need to select the Scene containing the labeled Frame. See the Set Label Action.

4.6.5.35 Return

Use this Action when you want to return a value from a Function.



Return a value from a user-defined function

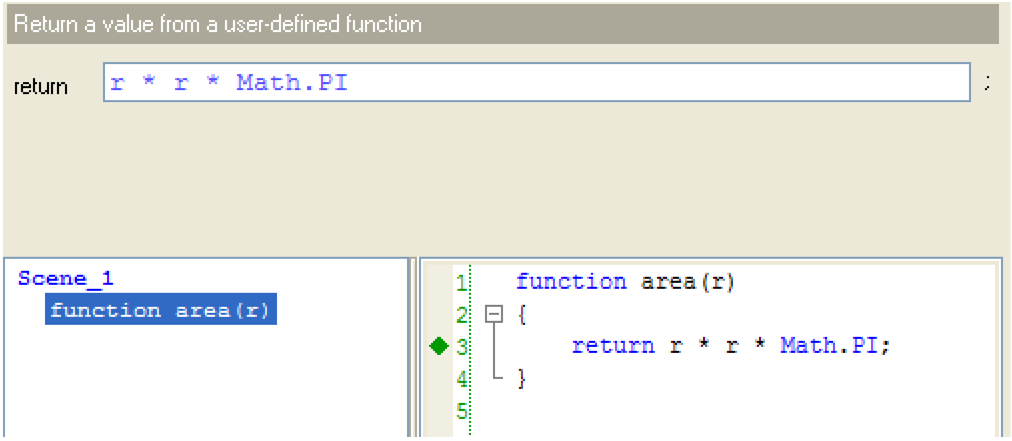
return ;

Return
The string or expression for the value you wish to be returned.

Example
If your function was intended to return the area of a circle, you would enter:

r * r * Math.PI

(assuming that r was the radius parameter of the function).



Return a value from a user-defined function

return r * r * Math.PI ;

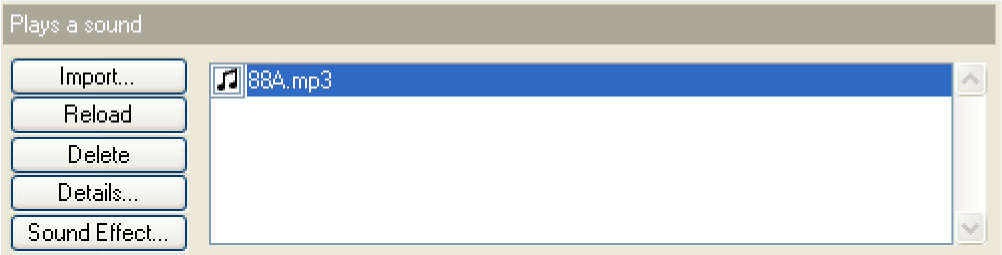
Scene_1
function area(r)

```
function area(r)
{
    return r * r * Math.PI;
}
```

See Also
Function

4.6.5.36 Play Sound

Use this Action to play a sound.



Only sounds that have been imported can be selected. Use the 'Import...' or 'Reload' buttons to import/reload a .wav or .mp3 sound file.

Note:

- Due to a limitation in the Flash Player, sounds will only play in response to a Frame Event, or the Mouse Events Press, Release, Roll Over or Roll Out. Sounds cannot be played in response to the Mouse Events Drag Over, Drag Out or Release Outside
- Due to a limitation in the Flash Player, MP3 sounds must have sample rates of 11025Hz, 22050Hz or 44100Hz
- The Play Sound action will ALWAYS play, even inside a conditional statement. This means conditional statements can not be used (such as if, else if, etc.) to trigger the Play Sound action, because it will play regardless of the conditions. To get around this limitation, put a gotoAndPlay event inside the Conditional statement, and send the timeline to a frame that contains the Play Sound action (either inside a Movie Clip, a level, or the _root).

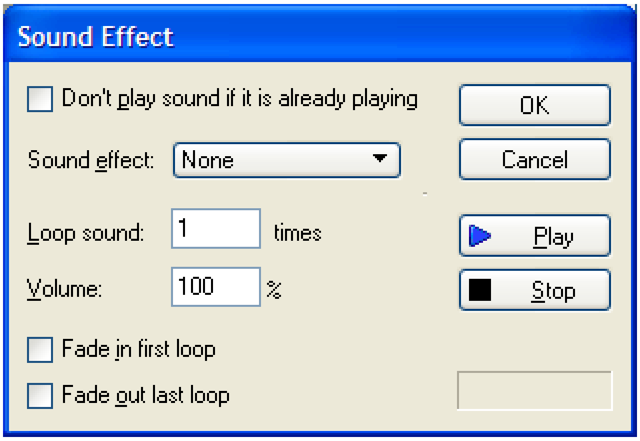
Import...
Imports a sound file into the Movie using the 'Import...' button. Supports importing Windows .wav files or .mp3 sound files.

Reload
Reloads a previously loaded sound file by selecting the sound in the 'Content' tree and using the 'Reload' button. This is useful if the sound file on disk had changed since the last time it was loaded.

Delete
Deletes a sound file from the Movie by selecting it in the 'Content' tree and using the 'Delete' button. This does not delete the sound from the disk drive, it only removes it from within the .swi file so it can't be played.

Details...
Also available from the 'Content' tree, the 'Details' dialog allows modification of the properties of a sound file including Compression, Channels and Sample Rate. See the section on Audio Properties for details.

Sound Effect
The 'Sound Effect' dialog box allows modification of how the sound file is played for this Action.



□ Don't play sound if it is already playing

Sound effect: None

Loop sound: 1 times

Volume: 100 %

□ Fade in first loop

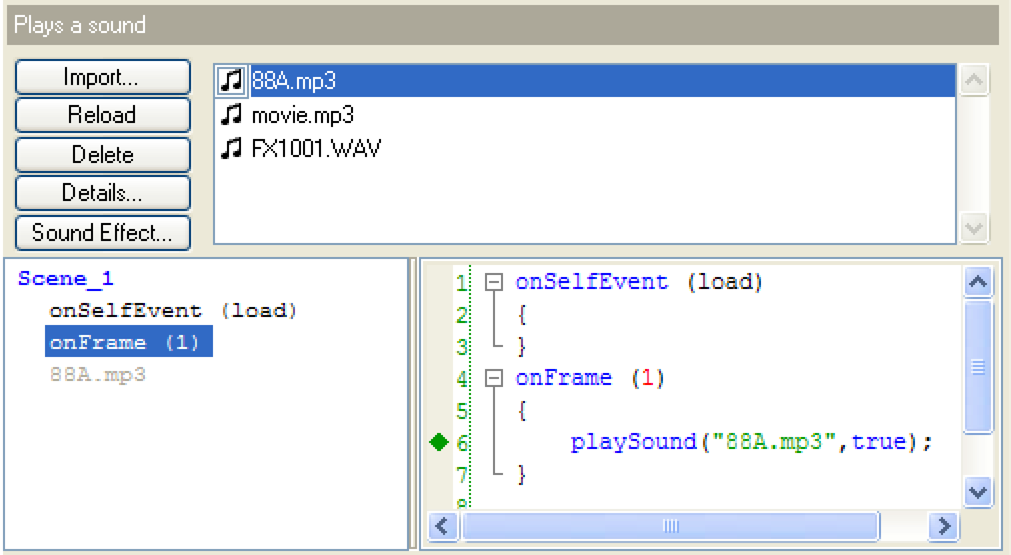
□ Fade out last loop

- **Don't play sound if it is already playing:** stops a sound being triggered multiple times
- **Sound effect:** specifies simple sound Effects, including fading in or out and panning effects
- **Loop sound:** specifies how many times the sound will be played. In the example above, the sound called "88A.mp3" is played 1 time. Due to a limitation in the Flash Player, there is no infinite loop option. If the sound is to loop for a very long time, enter a large number, such as 50,000
- **Volume:** specifies the volume of the sound for playback as a percentage of the original volume
- **Fade in first loop:** fades in the first loop played for the sounds file
- **Fade out last loop:** fades out the last loop played for the sound file
- **Play:** plays the sound with the current setting. The 'Play' button also updates the number of bytes for the sound file, as shown under the 'Stop' button
- **Stop:** stops the sound currently playing. This is useful for a long sound, or a sound with a large number of loops

Note: Setting the 'Loop Sound' option to "0" (zero) will only play the sound file one time. i.e. it will loop 'zero' times, so only playing through one full time. Also, changing the number of loops does not increase or decrease the file size of the exported .SWF file.

Content tree

The 'Content' tree displays the sound Objects that are available for playing in this Movie. In the example below, the sound called "88A.mp3" is played when the movie reaches Frame 1.



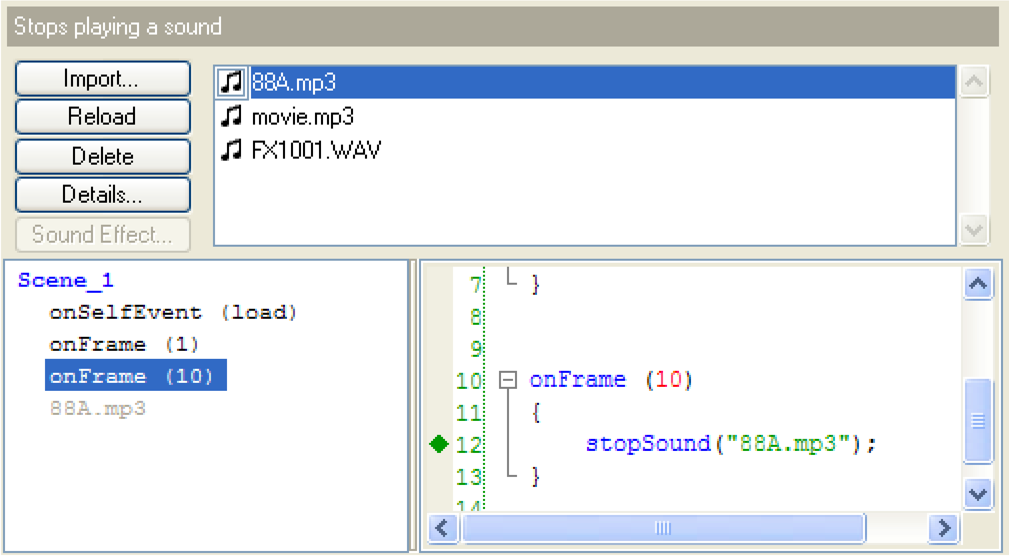
88A.mp3
movie.mp3
FX1001.WAV

Scene_1
  onSelfEvent (load)
  onFrame (1)
  88A.mp3

```
onSelfEvent (load)
{
}
onFrame (1)
{
    playSound("88A.mp3",true);
}
```

4.6.5.37 Stop Sound

Use this Action to stop playing the selected sound.



```
Scene_1
    onSelfEvent (load)
    onFrame (1)
    onFrame (10)
    88A.mp3
```

```
    }


onFrame (10)
{
    stopSound("88A.mp3");
}
```

The options and limitations are the same as for the Play Sound Action. However, instead of playing the selected sound, it is stopped.

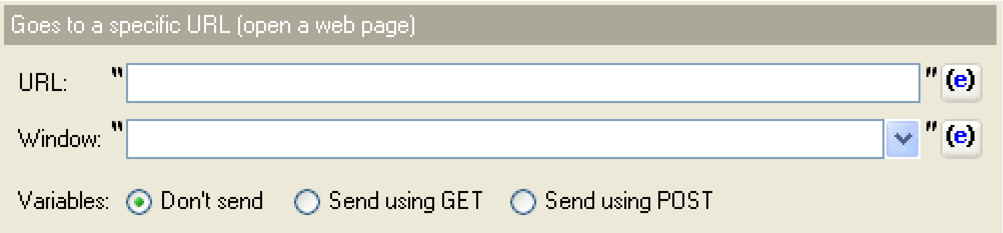
4.6.5.38 Stop All Sounds

Use this Action to stop playing all sounds that are currently playing for a Movie or Movie Clip.

There are no options for this Action.

4.6.5.39 Get URL

Use this Action to load an URL into the specified Target Frame of the browser (or whatever program is playing the Movie).



URL: " "

Window: " "

Variables: ● Don't send ○ Send using GET ○ Send using POST

Note: When playing within SWiSH Max, Get URL Actions are not performed

URL
You can specify an absolute or relative URL of the file to load. If you do not specify an absolute URL for the file, the Flash Player will assume the relative URLs are based on the URL for the currently open .swf file. When running in a browser, you can override the base URL using the 'Base' option on the HTML Export Panel.

Note: When you specify a URL to a separate web page, you need to include "http://" before the "www." in all external links.

Window
You can specify an HTML Target Frame to load the file into. This is not the same as a Frame in the SWiSH Max Movie, but rather an HTML Frame. You can leave this blank if you do not understand HTML Frames, or simply want to replace the page that contains the Movie with the specified URL.

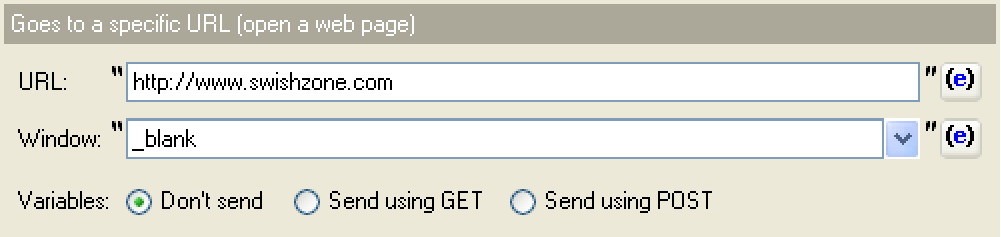
The default window options available are:

- _self = This will open the specified link within the same frame (or the same window if no frames are present).
- _blank = This will open a new browser window and load the link specified.
- _parent = This will open the link within the current browser window regardless of frame settings.
- _top = This will open the specified link in the top-level frame (can also be used to escape the current frameset).

Variables

- Don't Send = This will stop the getURL action from sending any variable information to the link specified.
- GET = This method is used to add the variable information at the end of the URL (best for minimal variable information)
- POST = This method will send the variable information in a separate HTTP header (best for long strings of variables)

In the example below, the browser will load the URL "http://www.swishzone.com" in a new window when the mouse button is pressed inside the current Object.



Goes to a specific URL (open a web page)

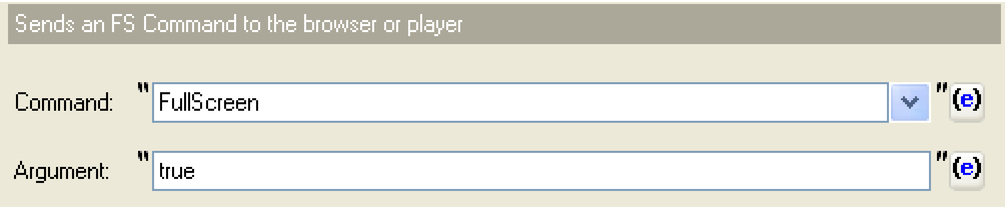
URL: "http://www.swishzone.com" (e)

Window: "_blank" (e)

Variables: ● Don't send ○ Send using GET ○ Send using POST

4.6.5.40 FS Command

Use this Action to send commands to the browser or player.



Sends an FS Command to the browser or player

Command: "FullScreen" (e)

Argument: "true" (e)

Command
You can specify the command to enter.

Argument
This provides extra information to the script for the given command.

Note: When playing within SWiSH Max, FS Command Actions are not performed

Using FSCommand with the stand-alone Flash Player or Projector
If the .swf Movie is playing within a stand-alone Flash Player or a projector, you can only use one of the predefined FS commands. Those predefined FS commands appear in the drop-down list on the FS Command Action page.

AllowScale
True: displays the .swf file scaled (not always 100%)
False: .swf file is always 100%

ShowMenu
True: displays the full Flash Menu on right mouse click (Windows). Displays the full Flash Menu on control + mouse click (Mac)
False: displays "About Shockwave Flash" only (Windows). Greys out the Menu (Mac).

FullScreen
True: .swf file is displayed full-screen size
False: .swf file is displayed at original size

Exec
Program Name: executes the specified program. This command is valid for stand-alone projectors only.

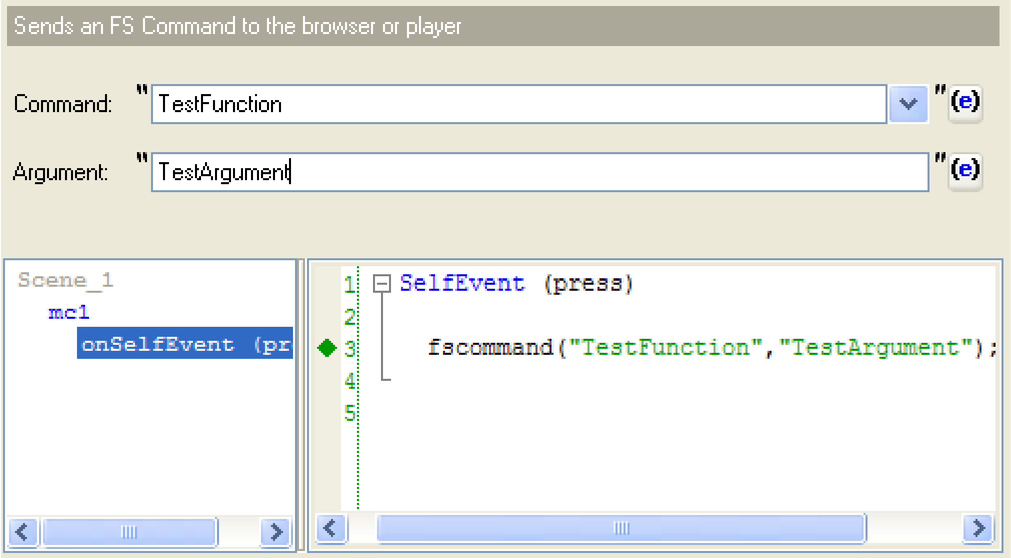
Quit
Quits the .swf file.

Using FS Command within an HTML document in a browser

If the Movie is playing within an .htm document in a browser, there are no predefined commands. Instead, the FS Command Action triggers some JavaScript/VBScript within the HTML page. This script has to test for the command name, and should then run appropriate script commands. The argument value is also available to the script to use as it is required.

When you use the 'Export HTML' command, SWiSH Max will add the necessary basic script to the .htm document when you export the Movie. All you have to do is to implement the function at the place marked "// ADD YOUR CODE HERE" in the exported .htm document.

In the example below, the Movie will call the embedded script function TestFunction in the .htm document when the mouse button is pressed inside the current object.



Sends an FS Command to the browser or player

Command: "TestFunction"

Argument: "TestArgument"

```
SelfEvent (press)

    fscommand("TestFunction","TestArgument");
```

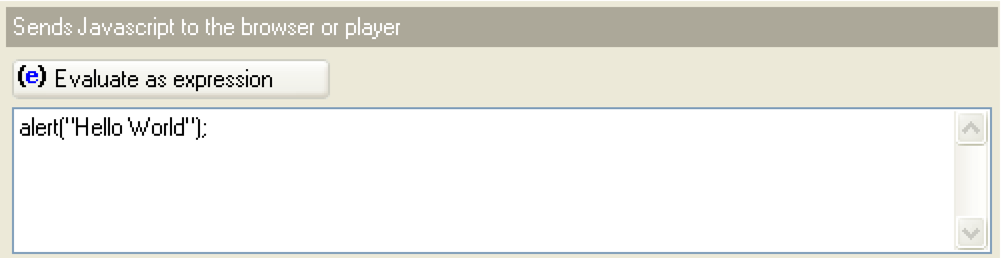
SWiSH Max will add the following script to the beginning of the .htm document when you export the Movie. Then you can replace "*// ADD YOUR CODE HERE*" with your own code.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
```

```
// Detect Browser
var InternetExplorer = navigator.appName.indexOf("Microsoft") != -1;
// FSCommand handler for Netscape
function Movie1_DoFSCommand(command, args) {
  var Movie1Obj = InternetExplorer ? Movie1 : document.Movie1;
  if (command=="TestFunction") {
    // ADD YOUR CODE HERE
  }
}
//-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
// FSCommand handler for VBScript and ActiveX
Sub Movie1_FSCommand(ByVal command, ByVal args)
  call Movie1_DoFSCommand(command, args)
end sub
//-->
</SCRIPT>
```

4.6.5.41 JavaScript

Use this Action to directly execute some JavaScript.

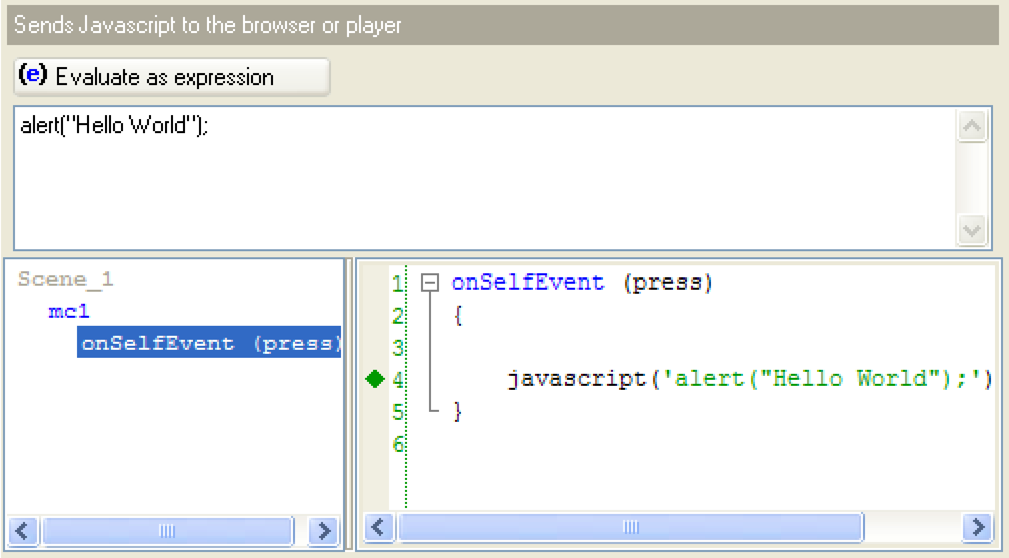

Sends Javascript to the browser or player
(e) Evaluate as expression

```
alert("Hello World");
```

The JavaScript can have multiple lines. SWiSH Max will automatically 'escape' the blanks and newlines etc, prefixes "javascript:", and appends ";void(0);" before sending it to the browser. This means you only have to include the actual JavaScript code itself.

Note: When playing within SWiSH Max or within a stand-alone player, JavaScript Actions are not performed

In the example below, the browser will call the specified JavaScript code to display the message "Hello World" in a dialog box when the mouse button is pressed inside the current Object.



Sends Javascript to the browser or player

(e) Evaluate as expression

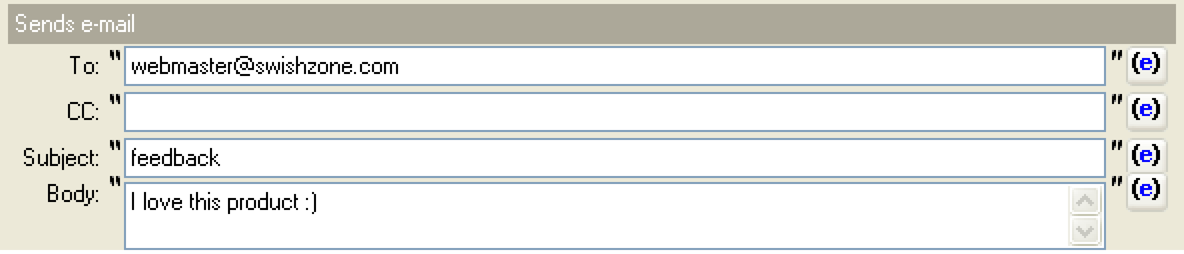
alert("Hello World");

Scene_1
mc1
onSelfEvent (press)

```
onSelfEvent (press)
{
    javascript('alert("Hello World");')
}
```

4.6.5.42 Mailto

Use this Action to send email.



Sends e-mail

To:	webmaster@swishzone.com
CC:	
Subject:	feedback
Body:	I love this product :)

When the Mailto Action is executed, the Flash Player will start the client's default email program (such as Outlook Express). The fields you have specified will be already filled in. The user can edit any of these or fill in any that are blank and then send the message.

To
You can put in the email address of the recipient.

CC
You can put in the email address of the recipient of a copy of the message.

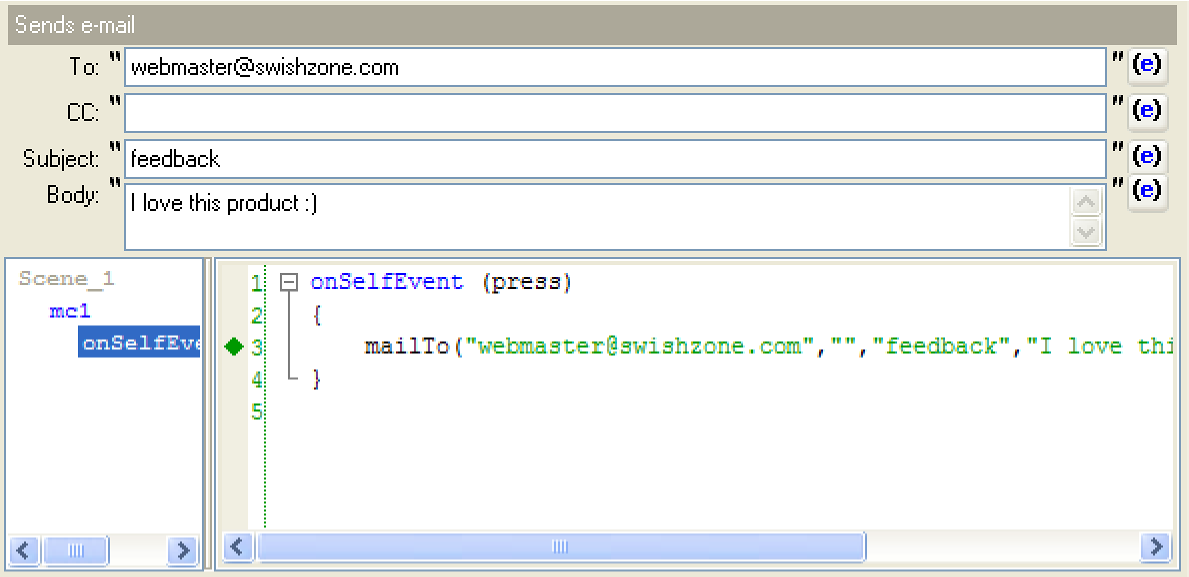
Subject
You can supply a subject line for the message.

Message body
You can supply text for the message. Newlines can be embedded for formatting.

Note: If you have a constant string expression (something SWiSH Max can evaluate as a string when it compiles the script), then SWiSH Maxwill automatically 'escape' any special characters like newlines for you. eg newlines become "%0a"

However, if you are building up the string dynamically so SWiSH Max cannot tell what it will be when it compiles your script, thne you need to do the escaping yourself. So instead of building a message body, say, with lines separated by newlines ("\n"), you need to use the escaped version ("%0a")

In the example below, the Movie will allow the user to send an email to webmaster@swishzone.com and a copy to sales@swishzone.com.



Sends e-mail

To: "webmaster@swishzone.com"

CC: ""

Subject: "feedback"

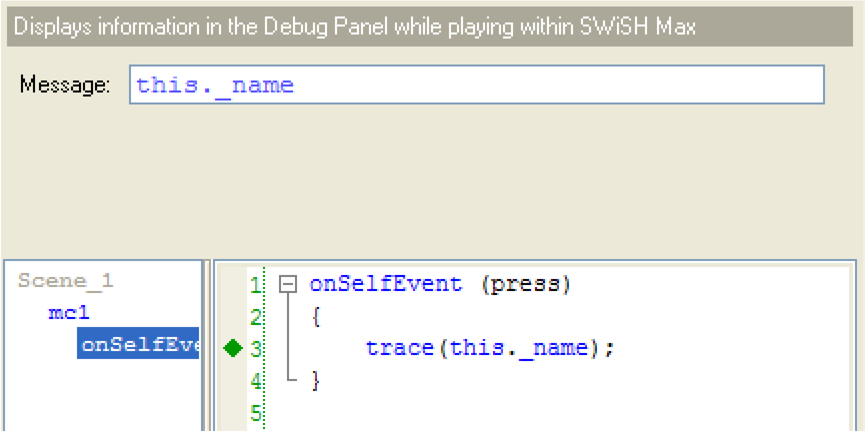
Body: "I love this product :)"

```
onSelfEvent (press)
{
    mailTo("webmaster@swishzone.com","","feedback","I love thi
}
```

4.6.5.43 Trace

Use this Action to display the results of an expression in the Debug Panel when playing your Movie.

In the example below, the name of the Movie Clip will be displayed in the 'Debug' Panel when the Movie Clip is pressed.



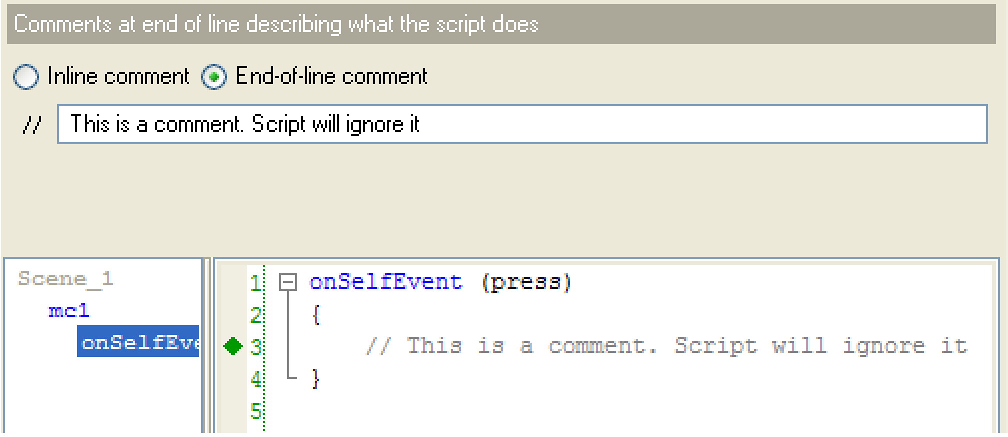
Displays information in the Debug Panel while playing within SWiSH Max

Message: this._name

```
onSelfEvent (press)
{
    trace(this._name);
}
```

4.6.5.44 Comment

Use this Action to add comments to your code. Comments do nothing at all to the programming, they just help document what is going on. As comments are not included in the exported .swf file, they do not effect the speed of loading or size of your website. The use of comments will assist greatly in the long term maintenance of your script.



Comments at end of line describing what the script does

○ Inline comment ● End-of-line comment

// This is a comment. Script will ignore it

```
onSelfEvent (press)
{
    // This is a comment. Script will ignore it
}
```

The radio button **Inline Comment** will change the comment to a comment that starts with
/* and ends with ***/**
The use of an end of comment delimiter allows such comments to span multiple lines. This style of comment is useful for adding large comments such as header blocks that describe the following block of code.

The radio button **End-of-line Comment** changes the comment delimiters so that it starts with **//** and ends at the end of the line. If the comment is to extend over multiple lines, each new line must start with **//**. This style of comment is useful for adding a comment about a specific statement.
For example:

```
this._x += 5; // move movie clip right by 5 pixels.
```

4.6.5.45 with

This action allows commands within the block of code to refer to the object named in the with statement.

For example:

```
onSelfEvent (load)
{
    var phonenumbers : Object = {nameFirst:"Jon", nameLast:"Smith", phone:"555 1245"};
    with(phonenumbers)
    {
        trace(nameFirst);
        trace(nameLast);
        trace(phone);
    }
}
```

Will cause the following output to be displayed in the debug window:

```
Jon
Smith
555 1245
```

This code is equivalent to:

```
onSelfEvent (load)
{
    var phonenumbers : Object = {nameFirst:"Jon", nameLast:"Smith", phone:"555 1245"};
    trace(phonenumbers.nameFirst);
    trace(phonenumbers.nameLast);
    trace(phonenumbers.phone);
}
```

4.6.5.46 try, catch and finally

These actions allow structured exception based error handling. This generally simplifies the handling of errors in deeply nested code.
Note: these commands are only supported by SWF players 7 or later.

When calling a function that could cause an exception error it is necessary to enclose the call in a **try{}** block of code. Any errors 'thrown' or generated by the function using a throw command can then be caught and analysed in the following **catch{}** block of code.

Note that multiple catch{} blocks can be used to test for different types of error. When multiple catch{} block are used, each catch block is expected to examine a different class or type of error.

The **finally{}** block of code is always executed irrespective of an error occurring or the use of return commands within the try or catch blocks of code. The finally{} block of code is typically used to clean up and delete any temporary objects that may have been created for the purposes of error checking.

The try, catch and finally blocks are typically arranged as follows:

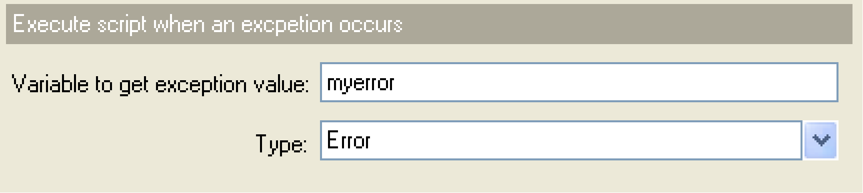
```
try
{
        // block of code that may generate exception error
}
catch(error:ErrorType1)
{
        // block of code to process errors associated with ErrorType1
        // the variable error contains details of the error.
}
catch(error:ErrorType2)
{
        // this code block is optional
        // block of code to process errors associated with ErrorType2
        // the variable error contains details of the error.
        // additional code blocks for any other error types should follow this block as required.
}
finally
{
        // this code block is optional
        // code in this block is always executed.
}
```

The following example demonstrates the use of exception based error processing:

```
function testthrow(p1)
{
    if (p1 != 0)
    {
        throw new Error("p1 not 0");
    }
}
onSelfEvent (load)
{
    try {
        testthrow(0);
    }

    catch (myerror:Error) {
        trace("Error: " add myerror);
    }
    finally {
        trace("All done");
    }
}
```

Note that changing the parameter of testthrow() from 0 to 1 will cause the error trace statement to appear in the debug panel. Note that in all cases the "All done" text appears in the debug panel.

Only the **catch{}** statement has an assist panel that allows user entry of parameters.



Execute script when an excpetion occurs

Variable to get exception value: myerror

Type: Error

4.6.5.47 throw

This command generates or 'throws' an error that can be caught by the catch code block. The calling section of code must be enclosed in a try code block.

This form of error handling allows a single block of code, the block associated with the catch statement to handle appropriate error processing. The advantage of this form of error processing is that errors generated by deeply nested function calls can be processed easily by the top level calling function without intermediate function calls needing to handle the error.

The type of the error variable should be Error or a class derived from the Error object.

Example:

```
function testthrow(p1)
{
    if (p1 != 0)
    {
        throw new Error("p1 not 0");
    }
}
```

4.6.6 Custom Classes

Custom classes can be created by defining a function that defines the class. This function is called the constructor function. The name given to the constructor function is the name of the class. It is often useful to assign the constructor as a _global variable so it can be accessed from anywhere in the movie.

The constructor function initialises instances of the class. Within the constructor function, use the 'this' keyword to refer to the instance being constructed. For example, a class for a Circle object that has a radius property:

```
    // define class Circle
    _global.Circle = function(radius) {
        this.radius = radius;
    }
```

To create an instance of a class use the 'new' operator. For example, to create an instance of the Circle class, and set the radius to 10:

```
    // create an instance of Circle
    var myCircle : Circle = new Circle(10);
```

The 'new' operator will create a new instance of class Circle named myCircle. The instance will be created using the _global.Circle function defined above.

Class methods can be defined by assigning them to a prototype property of the class.

For example, the following defines a getArea method for the Circle class

```
    // define method getArea
```

```
Circle.prototype.getArea = function() {
    return Math.PI * this.radius * this.radius;  // pi*r^2
}
```

Methods defined in this way can be called in a similar fashion to any built-in class. For example:

```
// create an instance of Circle
// and call getArea method
var myCircle : Circle = new Circle(10);
var area = myCircle.getArea();
trace("The area is: "+area);
```

A static or class method is a method that is related to the class, but not a specific instance. This type of method can be defined by directly assigning a function as a property of the class. In this case the **prototype** keyword must not be used.

Methods of this type can be called via the class name without having to use a specific instance of the class. For example:

```
// define class method calculateArea
Circle.calculateArea = function(radius) {
    return Math.PI * radius * radius;
}
// call class method calculateArea
var area = Circle.calculateArea(10);
trace("The area is: "+area);
```

Through subclassing it is possible to define a class so that it inherits all of the methods of another class. The class that inherits the methods is a "subclass". The class who's methods are inherited is the "superclass".

A subclass is created by:
1. Using the prototype keyword to reference the superclass.
2. Using the super keyword to access internal properties of the superclass.
3. Optionally defining new methods that help define the subclass.

For example, to create a subclass Cylinder that inherits from superclass Circle:

```
// define class Cylinder
_global.Cylinder = function (radius, height) {
    super (radius);       // item 2. radius of circle accessed via super key word.
    this.height = height;
}
// inherit from class Circle
Cylinder.prototype = new Circle;  // item 1. indicates that Cylinder inherits properties of class circle.
// define method getVolume. Item 3. new method that defines the volume of cylinder based on circle area and
Cylinder.prototype.getVolume = function() {
    return this.getArea() * this.height;
}
```

Once the subclass is constructed, the methods of both the subclass and the superclass are available.

```
// create an instance of Cylinder
// call subclass method getArea
// and call getVolume method
var myCylinder: Cylinder = new Cylinder(10,5);
var area = myCylinder.getArea();
trace("The area is: "+area);      // note this is area of circle, not external area of cylinder.
var volume = myCylinder.getVolume();
trace("The volume is: "+volume);
```

Full code example:

```
onFrame(1) {
    // define class Circle
    _global.Circle = function(radius) {
        this.radius = radius;
    }

    // define method getArea
    Circle.prototype.getArea = function() {
        return Math.PI * this.radius * this.radius;
    }

    // create an instance of Circle
    // and call getArea method
    var myCircle : Circle = new Circle(10);
    var area = myCircle.getArea();
    trace("The area is: "+area);

    // define class method calculateArea
    Circle.calculateArea = function(radius) {
        return Math.PI * radius * radius;
    }
    // call class method calculateArea
    var area = Circle.calculateArea(10);
    trace("The area is: "+area);

    // define class Cylinder
    _global.Cylinder = function (radius, height) {
        super (radius);
        this.height = height;
    }
    // inherit from class Circle
    Cylinder.prototype = new Circle;
    // define method getVolume
    Cylinder.prototype.getVolume = function() {
        return this.getArea() * this.height;
    }

    // create an instance of Cylinder
    // call subclass method getArea
    // and call getVolume method
    var myCylinder: Cylinder = new Cylinder(10,5);
    var area = myCylinder.getArea();
    trace("The area is: "+area);
    var volume = myCylinder.getVolume();
    trace("The volume is: "+volume);
}
```

4.6.7 Scripting Tutorials

The following tutorials can be loaded from the Samples area (File | Samples | Tutorials | filename.swi).

.swi File	Tutorial Name	Purpose
helloworld.swi	Hello World	Demonstrate the use of the trace command and Debug Panel
flowcontrol.swi	Variables and Flow Control	Demonstrate the use of variables, if and while statements
button.swi	Button	Demonstrates the on(release) Button Event
button1.swi	Button with rollOver	Demonstrate the on(rollOver), on(rollOut) Button Events. Demonstrate the _xscale and _yscale properties
calculator.swi	Calculator	Introduces the Text Object with Input and Dynamic text properties. Introduces the Math Object
properties.swi	Object Properties	Introduce the Object properties: _X, _Y, _xscale, _yscale, _width, _height, _alpha, _rotation, _name and _visible
dragging.swi	Mouse Dragging	Introduces startDragUnlocked(), startDragLocked() and stopDrag() methods. Introduces onSelfEvent(press) and onSelfEvent(release) Events to start and stop dragging

collide.swi	Collision Detection	Introduces isNearTarget() and isNearThis() methods
game.swi	Simple Gaming	Introduces User defined functions, use of onSelfEvent (load) Event Handling function to initialise variables and the use of the onSelfEvent (enterFrame) Event Handling function to animate Objects
game1.swi	Physics Properties	Shows how various gaming functions can be simplified by using Physics Properties.
moviecontrol.swi	Movie Control	Demonstrates how to control timelines playing in Movie Clips and how Movie Clips can control the main movie timeline using various methods of the gotoAndPlay() action.
welcome.swi	Input and Dynamic Text	Demonstrates how users can add text to the movie to be used at a later time in the same movie. Introduces methods for user input through Input Text fields and storing that information as a variable to be used in dynamic content
maxarray.swi	Arrays and Random	Demonstrates how to use Arrays to display a random message. Introduces methods for Arrays and Math.randomInt()
txtmessage.swi	Data Transfer - text	Demonstrates how data can be transferred to and from .htm, .asp, .php or text files that are resident on a server. Introduces the methods loadVariables() and shows how you can load an external .txt file into your movie at runtime.

It is recommended that Assist mode be selected in the Script Panel for the entry of the following tutorials. The tutorials assume that you are familiar with the SWiSH Max drawing tools.

4.6.7.1 Hello World

Description

This is a step-by-step tutorial for creating a Movie displaying the words "Hello World" in the Debug Panel.

Aim

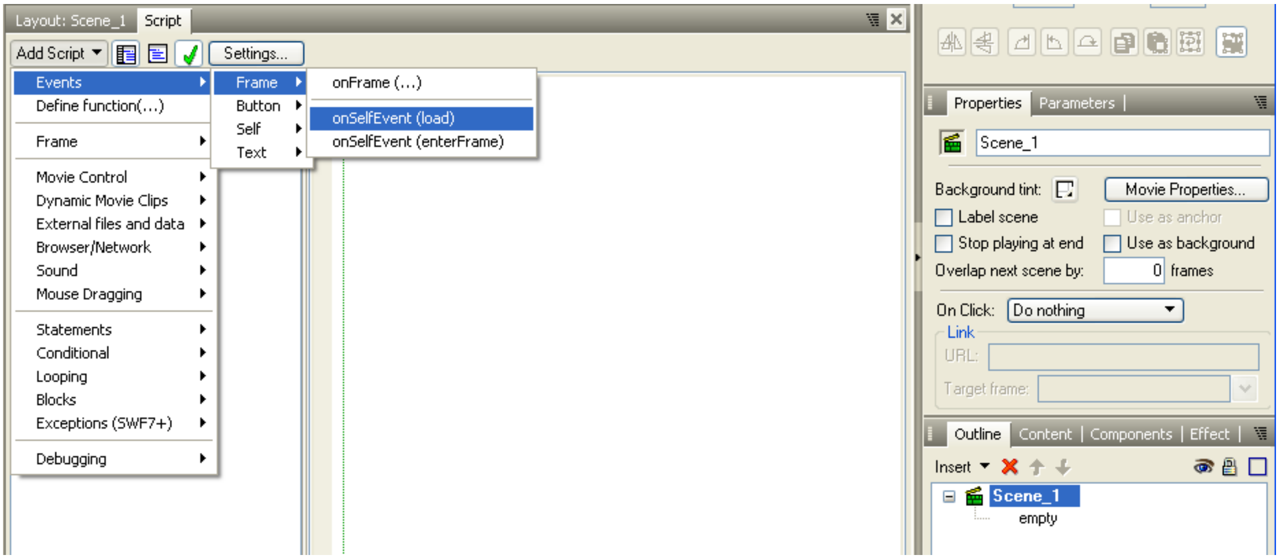
This Movie introduces the method of entering scripts, explains use of the onSelfEvent (load) Event and demonstrates the use of the trace command and Debug Panel.

.swi file

"helloworld.swi"

1. Start with a new project (File | New) and save the file as "myhelloworld.swi"

2. Select the 'Script' Panel, then enter an onSelfEvent (load) Event for Scene 1, as shown below.



Scene_1

Background tint:

□ Label scene
□ Use as anchor
□ Stop playing at end
□ Use as background

Overlap next scene by: 0 frames

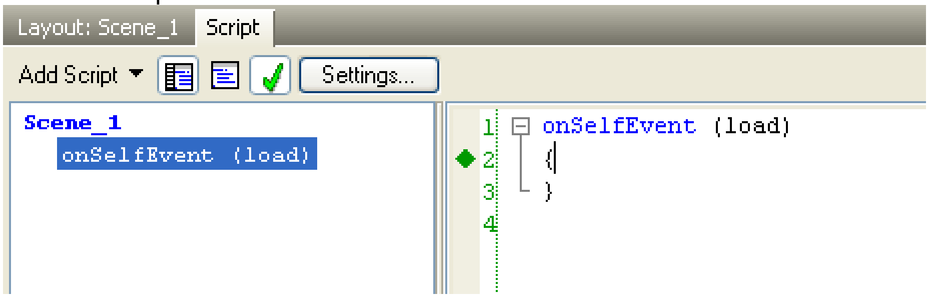
On Click: Do nothing

Link

URL:

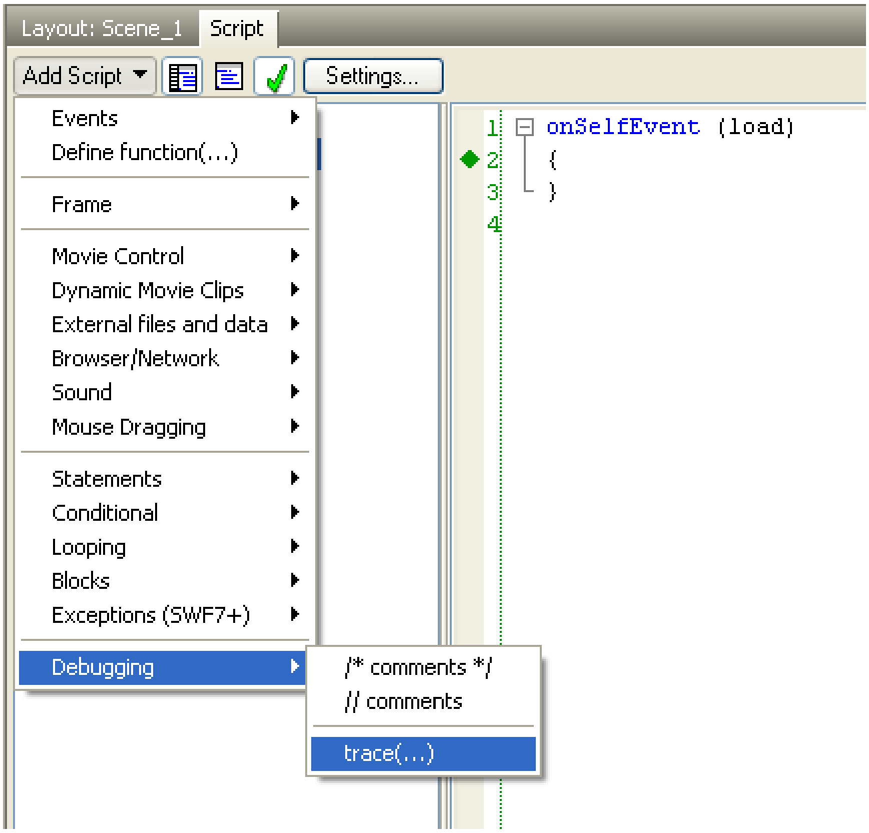
Target frame:

The 'Script' Panel should now look like this:



```
onSelfEvent (load)
{
}
```

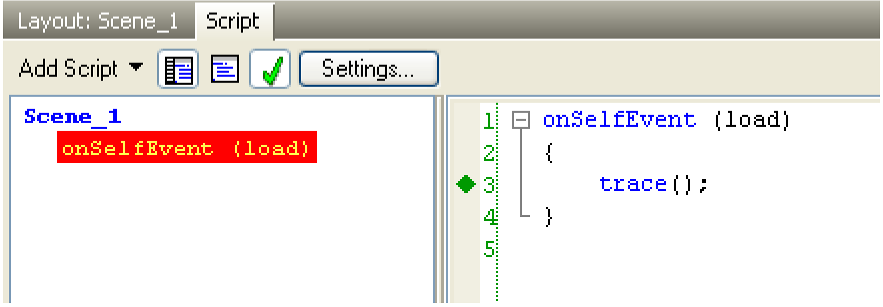
3. Add a trace command to the onSelfEvent (load) Event by selecting the onSelfEvent (load) function, as shown above. Press the 'Add Script' button then select Debugging | trace(...) from the menu as shown below:



```
onSelfEvent (load)
{
}
```

Note: It is possible to append a Scripting command by right-clicking on the command you want the new command to appear after, then selecting the Add Script Menu item. Sections of script can also be duplicated and moved using the Cut and Paste Menu options

The 'Script' Panel should now look like this:



```
onSelfEvent (load)
{
    trace();
}
```

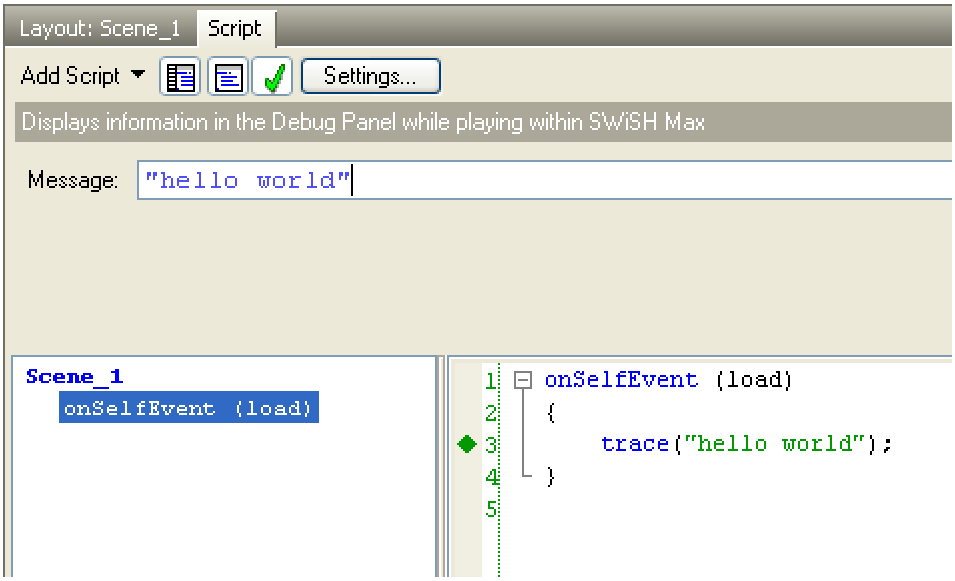
Note: The function is shown in red. This is because the trace statement is currently incomplete

4. Complete the trace statement by adding the string "Hello world" (including the quotation marks) so that the trace statement looks like this:

   trace("hello world");

   Alternatively you can enable the assist pane using the assist pane button, use the left mouse button to click on the word **trace** then enter the text into the Message Area of the assist pane. The assist pane can be used to enter parameters if you are unsure of the exact syntax.

   The Panels should now look like this if you have turned on the assist pane:



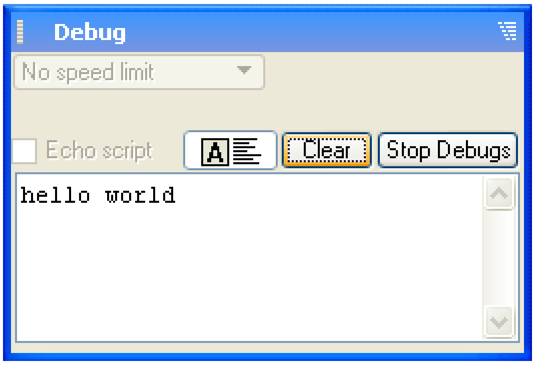
Displays information in the Debug Panel while playing within SWiSH Max

Message: "hello world"

```
onSelfEvent (load)
{
    trace("hello world");
}
```

Note: If the syntax is correct, the function will no longer be shown in red in the script outline pane.
Note: If you are confident in the use of command syntax, you can enter commands directly by typing into the Script Editor section of the Script panel.

5. Check your script by pressing the 'Play' button to run your Movie. The 'Debug' Panel if not already shown should appear as a floating panel and show the following:



```
hello world
```

6. Save your Movie

Analysis
When the 'Play' button is pressed, the Movie loads. This causes the script in the onSelfEvent (load) Event Handling Function to be executed.
The trace statement containing the string "Hello World" is then executed, causing the string to be displayed in the Debug Panel.

4.6.7.2 Variables and Flow Control

Description
This step-by-step tutorial develops the previous tutorial by introducing variable calculations and flow control.
After the Movie displays the words "Hello World" in the Debug Panel, it will calculate the results of y = x * x in the Debug Panel, where x has the values of 0 to 10.
When the result y is greater than 50, it will add the additional comment "wow that is big".

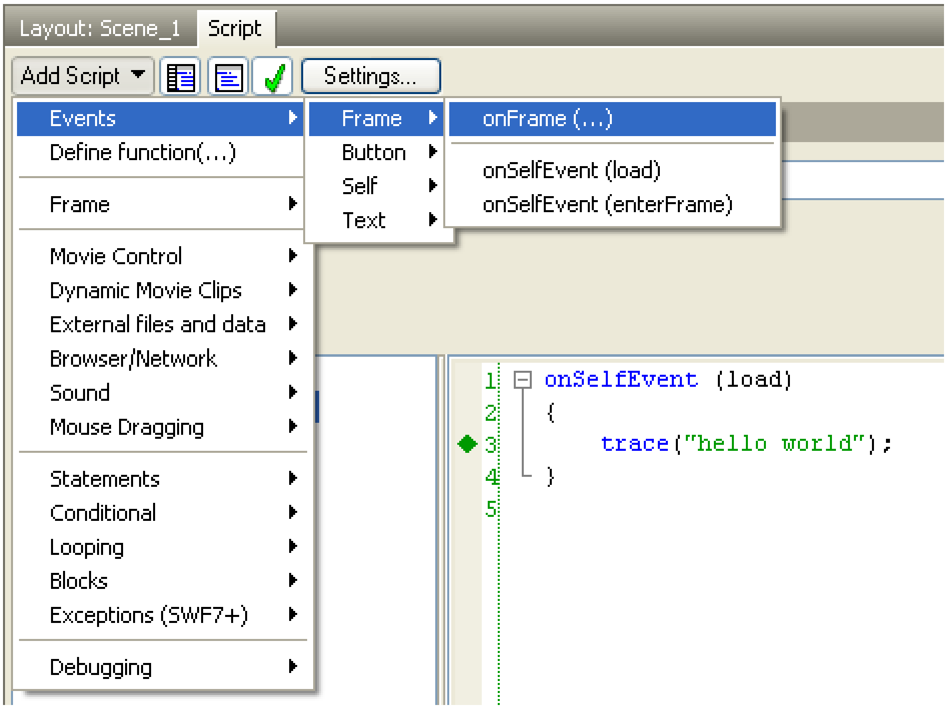
Aim

This Movie introduces the onFrame() Event, demonstrates the use of the trace command and Debug Panel and demonstrates the use of variables, as well as the if and while conditional statements.

.swi file
"flowcontrol.swi"

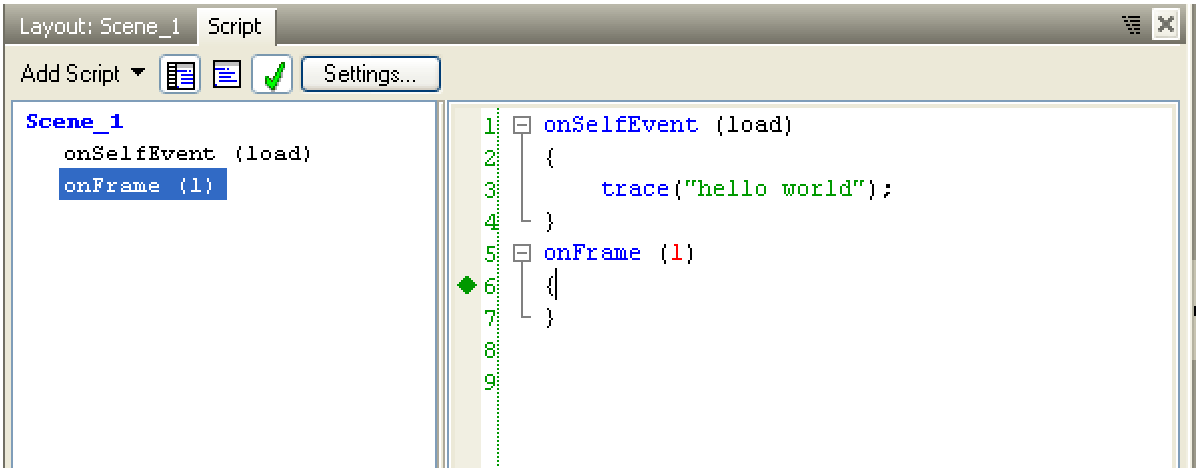
1. Load the .swi file "helloworld.swi". Ensure that the 'Play' button is not pressed and save the file as "myflowcontrol.swi".

2. Enter an onFrame Event to Scene_1. This Event will occur when the Movie reaches the specified Frame.



```
onSelfEvent (load)
{
    trace("hello world");
}
```

As Frame 1 is the currently selected Frame in the Timeline, the onFrame Event will be for Frame 1 of Scene_1. After inserting the Event, the 'Script' Panel should look like this:
(**Note**: the assist pane has been turned off to improve readability)



```
onSelfEvent (load)
{
    trace("hello world");
}
onFrame (1)
{
}
```

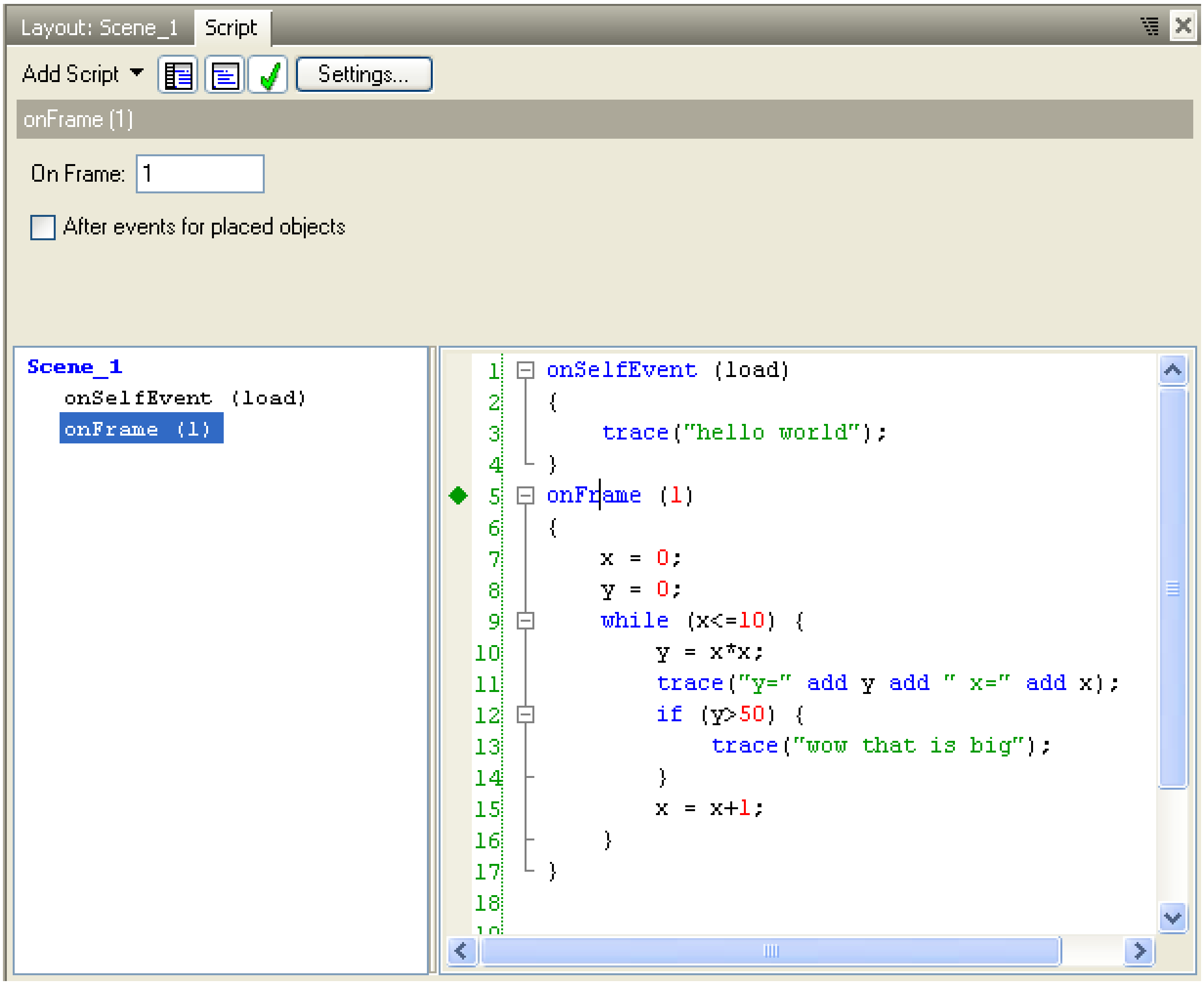
Note:

- The Timeline now indicates via an 's' that a Scripting Event will occur in Frame 1 of Scene 1. If needed, this onFrame Event can be moved to a different Frame by altering the number in the brackets for the onFrame event. Alternatively, use the left mouse button to select the line containing the onFrame() statement, enable the assist pane and enter the number into the On Frame field. The Event can also be moved to a different Frame by dragging the 's' in the Timeline
- The onSelfEvent (load) event was entered in the previous tutorial (Hello World).

2. Turn on the assist pane and add the following statements to the onFrame (1) Event. Note that script can also be added by typing directly into the script editor. The Menu commands used to enter the script are shown under the **Comments** heading

Statement	**Comments.**
`x = 0;`	Enter using Add Script | Statements | name = expr; Type "x" into the Name field (this field defines a property or variable). Then add the value "0" (zero - with no quotes) into the expression field below the Operator. In this case the Target: field can be left blank. This option is used to define the name of the Object that contains the property or variable. Leaving this value blank is the same as entering the value this
`y = 0;`	Enter using Add Script | Statements | name = expr; Another way to enter this statement is to copy and paste the x=0 statement, then change to parameters to make this statement y = 0;
`while (x <= 10) {`	Enter using Add Script | Looping | while (...) { Enter x <= 10 into the while field of the assist pane
`  y = x * x;`	Enter using Add Script | Statements | name = expr; or copy and modify previous statement
`  trace("y = " add y add " x = " add x);`	Enter using Add Script | Debugging | trace(...) Note that the add command is used to join multiple strings
`  if (y > 50) {`	Enter using Add Script | Conditional | if (...) {
`    trace("wow that is big");`	Enter using Add Script | Debugging | trace(...) This command is only executed if the value of y is greater than 50
`  }`	Entered as part of the if statement. Shows the end of the if statement
`  x = x + 1;`	Click on the ending bracket for the IF statement, and enter this script using Add Script | Statements | name = expr; If this statement is missing, x will never increase and loop will continue indefinitely. Note that x++ or x += 1 statements could have also been used
`}`	Entered as part of the while statement. Shows the end of the while loop

Once you have entered the above script commands, your 'Script' Panel should look like this:

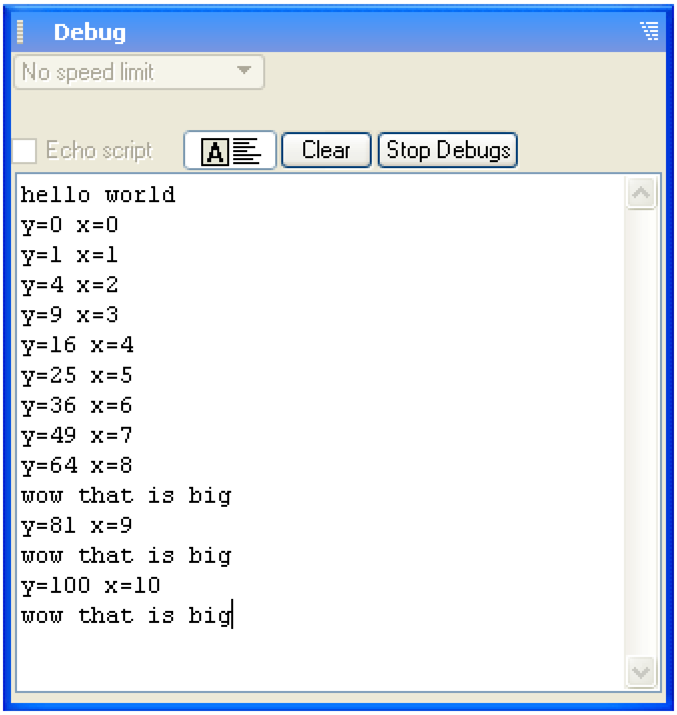


On Frame: 1

□ After events for placed objects

```
onSelfEvent (load)
{
    trace("hello world");
}
onFrame (1)
{
    x = 0;
    y = 0;
    while (x<=10) {
        y = x*x;
        trace("y=" add y add " x=" add x);
        if (y>50) {
            trace("wow that is big");
        }
        x = x+1;
    }
}
```

3. To check your script, select the Debug tab among the Panels displayed on the right hand side and press the 'Play' button to run your Movie. The 'Debug' Panel should show the following:



```
hello world
y=0 x=0
y=1 x=1
y=4 x=2
y=9 x=3
y=16 x=4
y=25 x=5
y=36 x=6
y=49 x=7
y=64 x=8
wow that is big
y=81 x=9
wow that is big
y=100 x=10
wow that is big
```

Analysis

When the 'Play' button is pressed, the Movie loads. This causes the script in the onSelfEvent (load) Event Handling function to be executed.
The trace statement containing the string "Hello World" is then executed causing the string to be displayed in the Debug Panel.

Once the Movie has been loaded it starts playing. On frame 1 of Scene_1 the onFrame(1) Event Handling function containing the following script is executed:

```
x = 0;
y = 0;
while (x <= 10) {
            y = x * x;
            trace("y = " add y add " x = " add x);
            if (y > 50) {
                        trace("wow that is big");
            }
            x = x + 1;
}
```

x and y are both Variables. All variables should be defined a value before use. For this reason, the statement x=0 should be included as the subsequent statement x = x + 1 assumes that x already has a value. As the variable y is only assigned values, the y=0 statement is not strictly necessary, but it is always good practice to set up initial values.

The while loop will continue to execute the contained statements until the value of x exceeds 10. As the last statement of the while loop is x = x + 1, the loop will execute with the values of x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and terminate when x = 11.

The trace statement trace("y = " add y add " x = " add x); is used to display the values for both y and x. The add operator is used to join multiple strings into a single string. Note that the variables x and y are converted to strings for display in the Debug Panel.

On each iteration of the while loop, the value of y is compared to 50 by the if (y > 50) statement. If the variable y is greater than 50 then the statement contained by the if statement, trace("wow that is big"); is executed. This causes the line "wow that is big" statement to be displayed each time y is bigger than 50.

4.6.7.3 Button

Description
This is a step-by-step tutorial for creating a button that executes script when pressed.

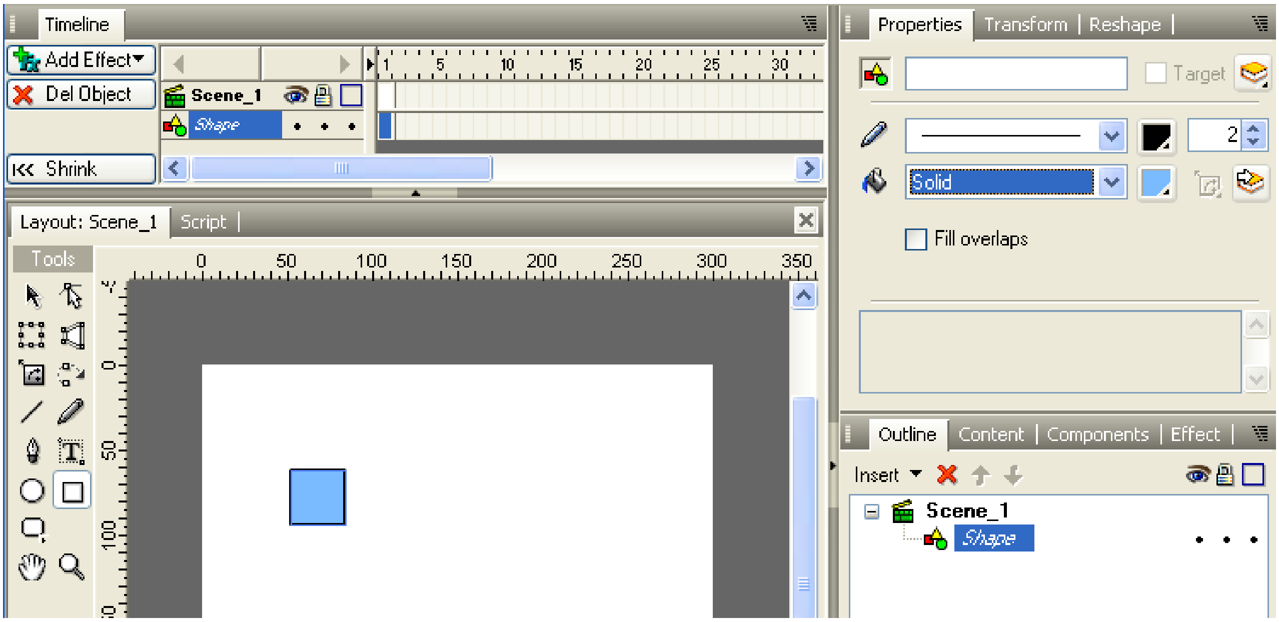
Aim
This tutorial introduces the procedure for making a simple button and demonstrates the on(press) Button Event.

.swi file
"button.swi"

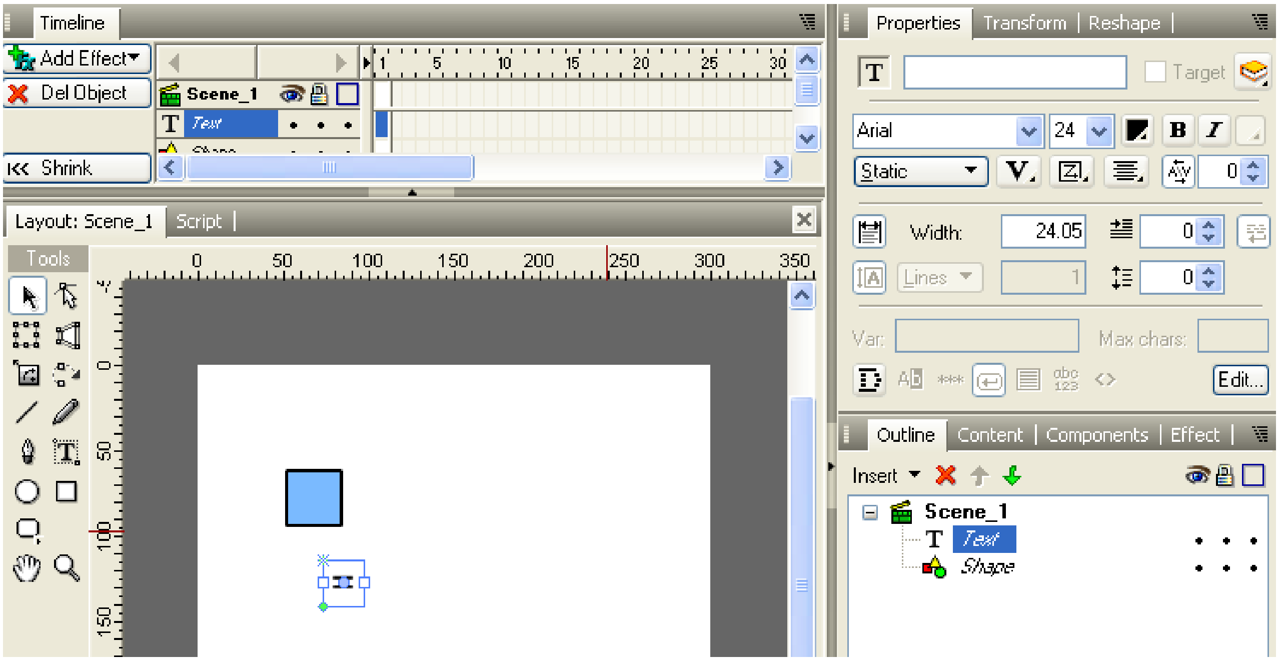
1. Start with a new project (File | New) and save the file as "mybutton.swi"

2. Create a button.

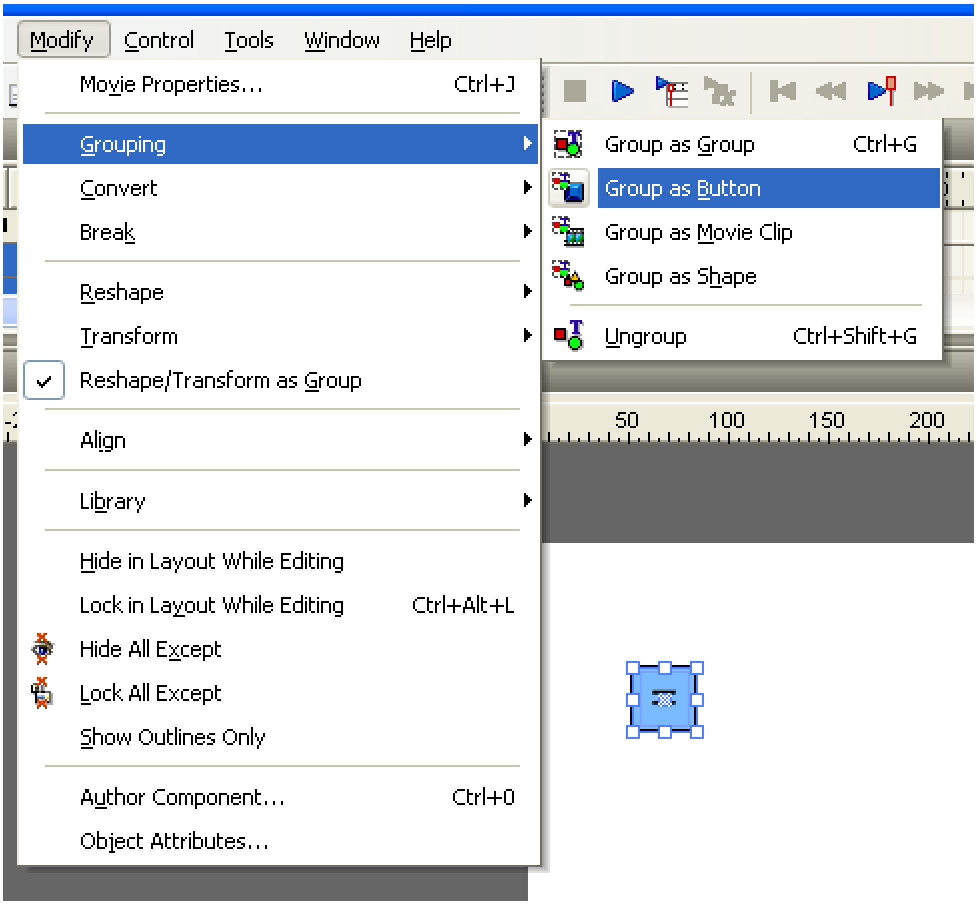
   Select the Layout Panel and use the Rectangle tool to create a small rectangle or square.



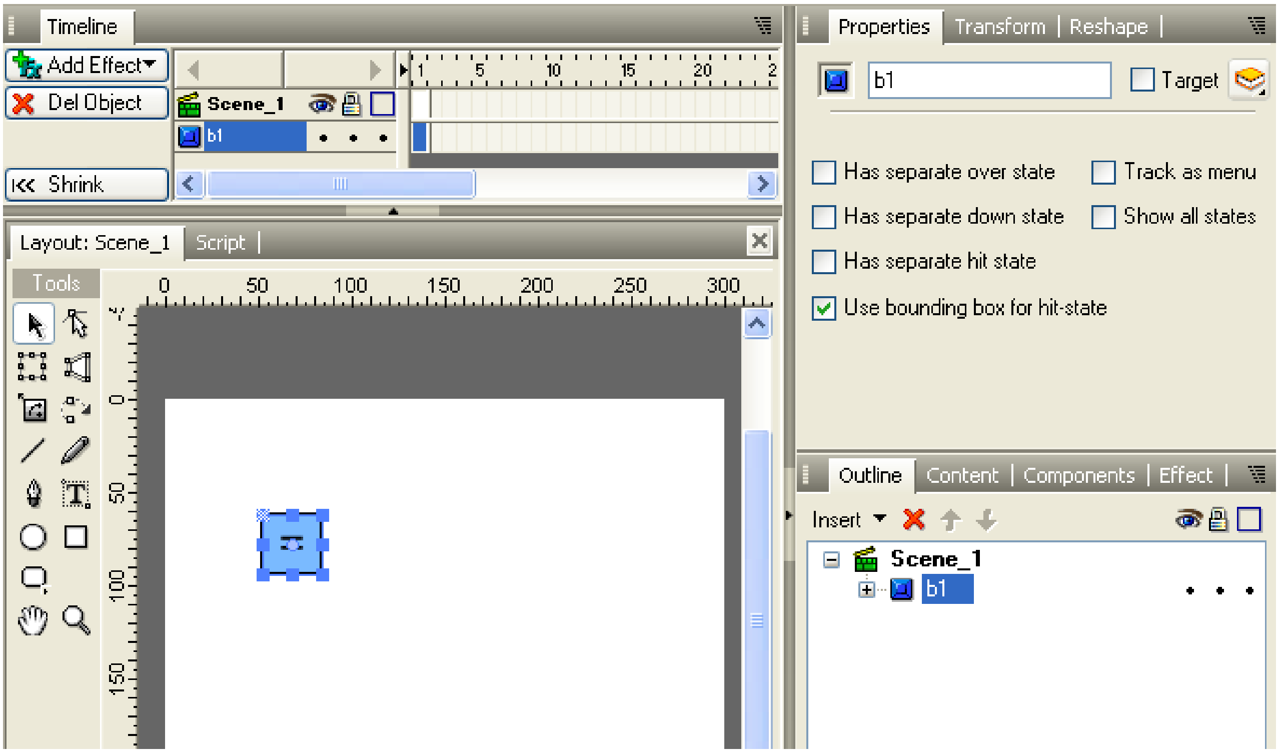
   Use the Text Tool to create the static text "=".



Move the static text over the rectangle, then select both the text and the shape. Use the Modify menu and select Grouping then Group as Button.



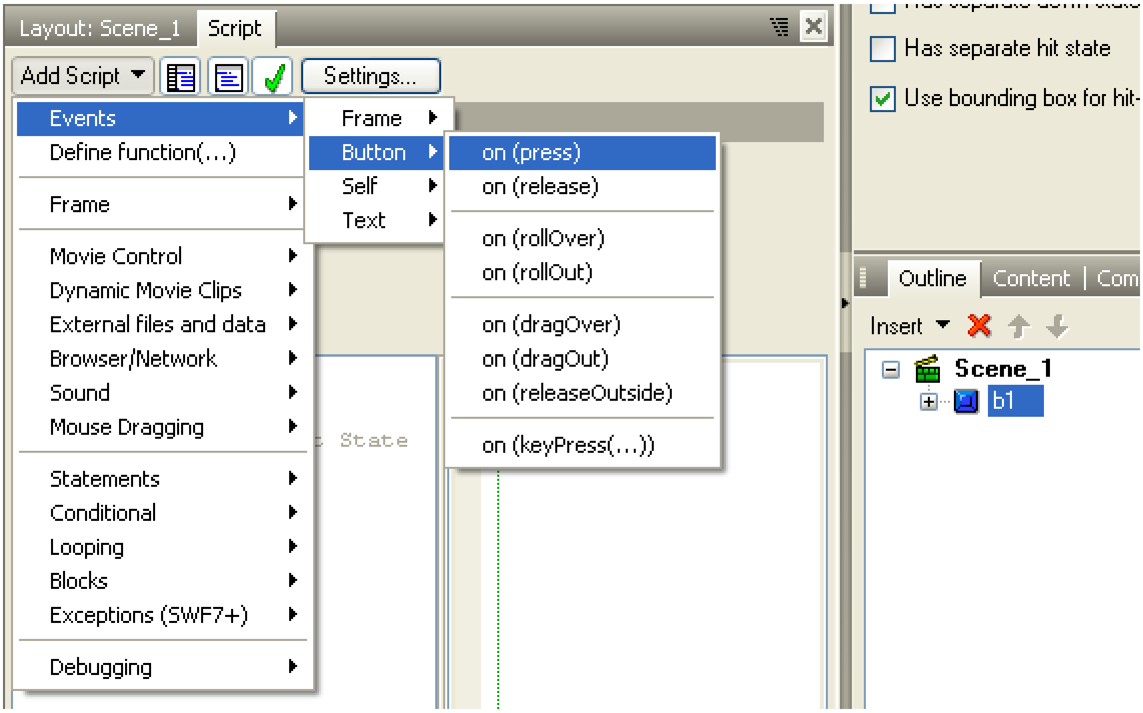
Name the button b1 and select the checkboxes as shown below



b1

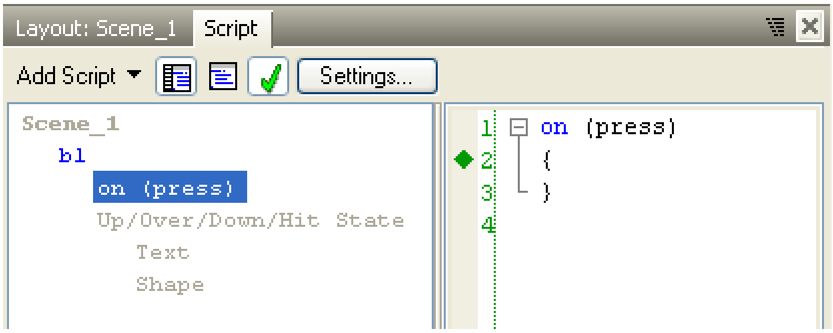
☐ Has separate over state ☐ Track as menu
☐ Has separate down state ☐ Show all states
☐ Has separate hit state
☑ Use bounding box for hit-state

3. Add some script to indicate when the button is pressed. Ensure the 'b1' button is selected in the 'Outline' Panel. Select the 'Script' Panel and use the 'Add Script' button to add an on(press) Button Event for the button. This is entered via the Menu items Events | Button | on(press).



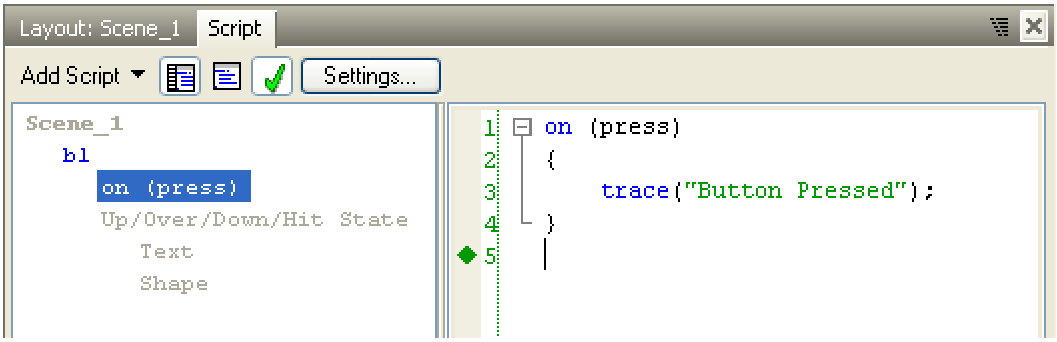
☐ Has separate hit state
☑ Use bounding box for hit-

The resulting script should be displayed:



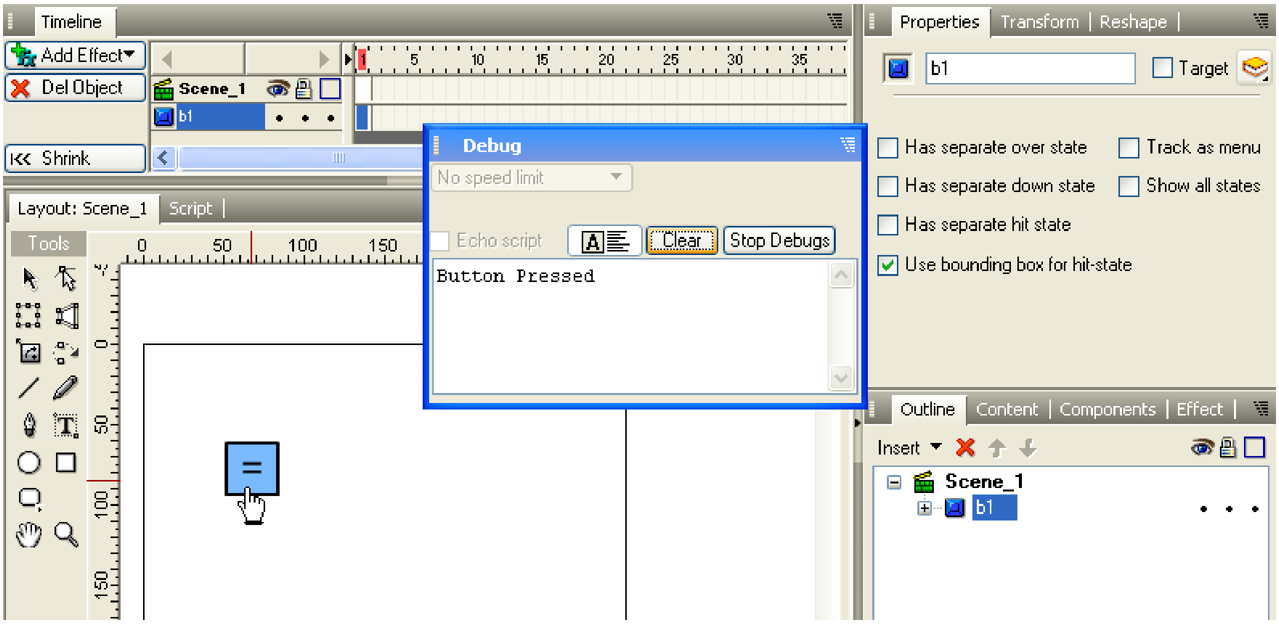
```
on (press)
{
}
```

Add the trace statement **trace("Button Released");** using the Menu commands [Add Script | Debugging | trace(...)](). The resulting script should look like:



```
on (press)
{
    trace("Button Pressed");
}
```

4. Test the button by selecting the Debug tab among the Panels displayed on the right hand side of the 'Layout' Panel. Press the 'Play' button to run your Movie. Each time the left mouse button is pressed while over the button, the statement "Button Pressed" should appear in the 'Debug' Panel



```
Button Pressed
```

b1 □ Target

□ Has separate over state □ Track as menu
□ Has separate down state □ Show all states
□ Has separate hit state
☑ Use bounding box for hit-state

Analysis
Two Objects, a shape and a static Text Object, were grouped and made into a button using the Group as button command.

Once the button was created, an on(press) Button Event was added. The statements contained within this Event routine execute each time the left mouse button is pressed while the mouse is positioned over the button.

A trace statement was added to the on(press) Button Event to demonstrate the button Action in the 'Debug' Panel.

4.6.7.4 Button with rollOver

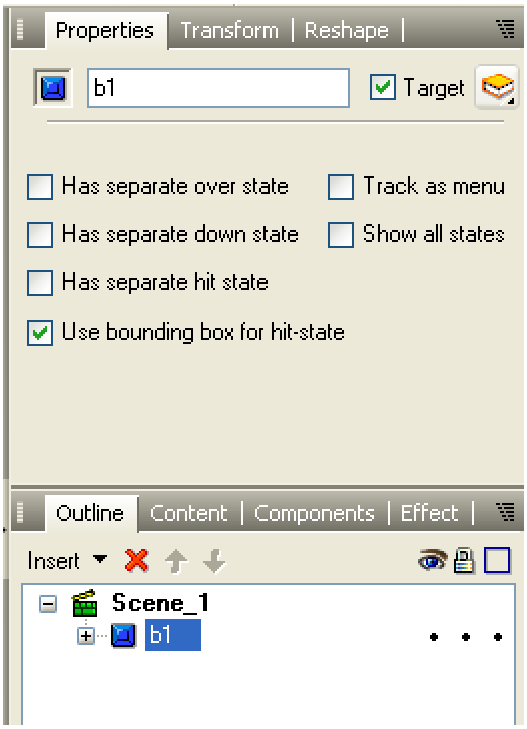
Description
This is a step-by-step tutorial for creating an animated button that changes size when the mouse is positioned over it.

Aim
This tutorial demonstrates the on(rollOver), on(rollOut) Button Events as well as the _xscale and _yscale properties.

.swi file
"button1.swi"

1. Continue from previous button tutorial, or load the "button.swi" file and save the file as "mybutton1.swi".

2. Check the 'Target' checkbox in the Button Panel. This makes the button a Scripting Object, which allows us to access and manipulate the properties associated with it



b1 ☑ Target

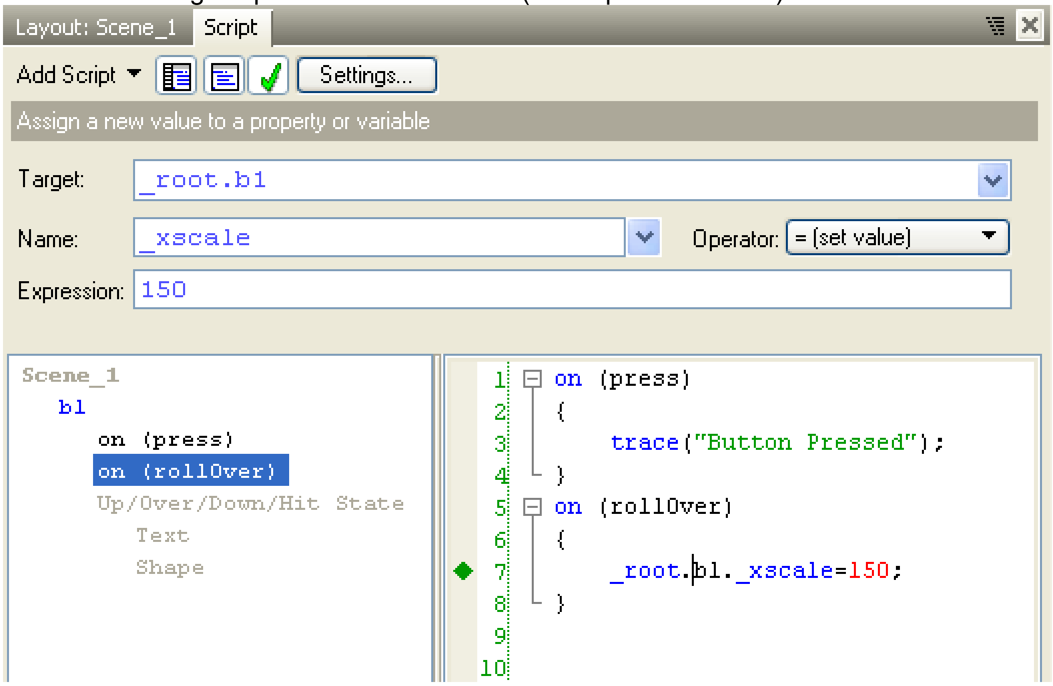
□ Has separate over state
□ Track as menu
□ Has separate down state
□ Show all states
□ Has separate hit state
☑ Use bounding box for hit-state

3. Select the 'b1' button in the 'Outline' Panel, if not already selected. Select the 'Script' Panel and add an on(rollOver) Button Event for the button by selecting the Menu items Add Script | Events | Button |

on(rollOver)

4. Within the on (rollOver) function add the script **_root.b1._xscale = 150;**
This is added via the menu items Add Script | Statements | name = expr;
If using the assist pane, in the 'Target' field, enter **_root.b1,**
In the 'Name' field, select _xscale using the drop-down menu.
Leave the 'Operator' field as =
In the bottom section enter the number **150**.

The resulting script should look like this (assist pane is shown):



Assign a new value to a property or variable

Target: _root.b1

Name: _xscale

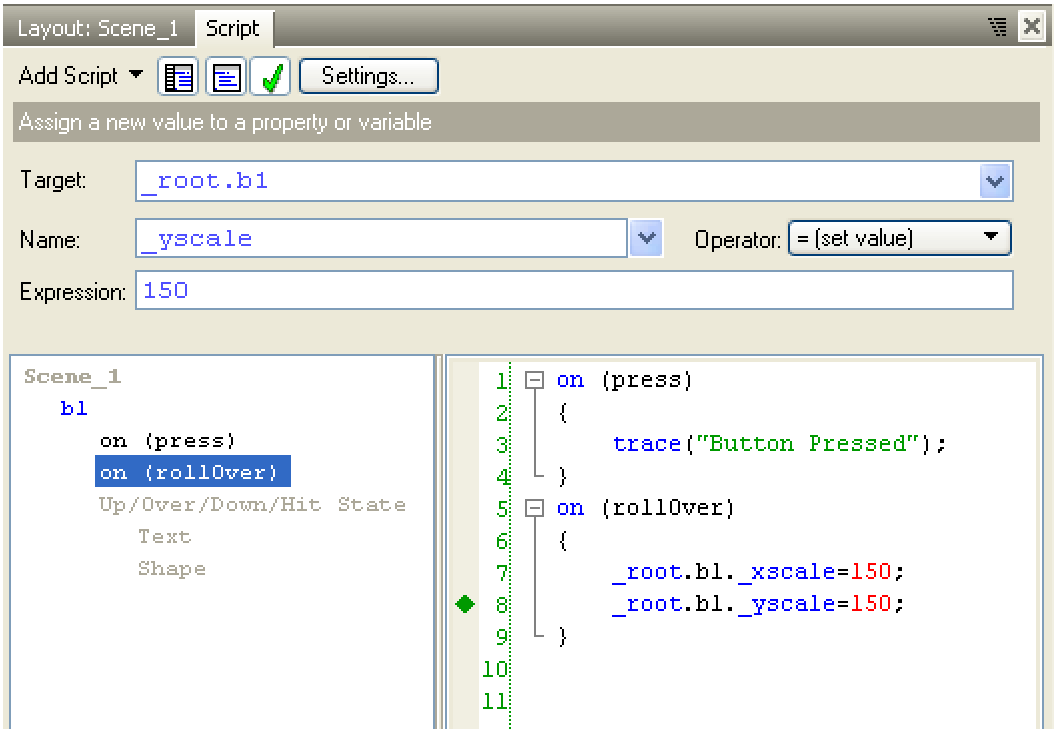
Operator: = (set value)

Expression: 150

```
on (press)
{
    trace("Button Pressed");
}
on (rollOver)
{
    _root.b1._xscale=150;
}
```

Use a similar procedure (or use right click, Copy and Paste then modify parameters) to enter the statement:
_root.b1._yscale = 150;

The resulting script should look like this:



Assign a new value to a property or variable

Target: _root.b1

Name: _yscale

Operator: = (set value)

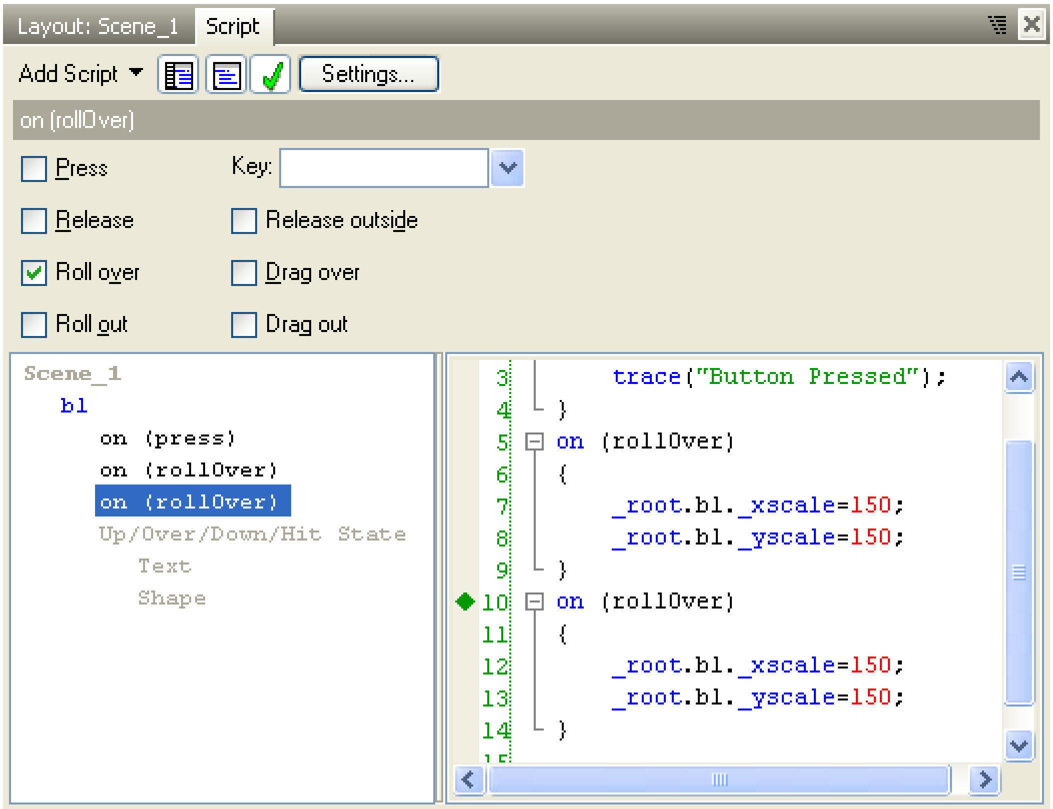
Expression: 150

```
Scene_1
   b1
      on (press)
      on (rollOver)
      Up/Over/Down/Hit State
         Text
         Shape
```

```
on (press)
{
    trace("Button Pressed");
}
on (rollOver)
{
    _root.b1._xscale=150;
    _root.b1._yscale=150;
}
```

5. Create an on(rollOut) Button Event to return the button to its original size when the mouse moves away from the button. This Event can be created by repeating **Step 4** using the on(rollOut) Event function. A quicker way to create the new Event is to copy the existing one and modify it appropriately, as shown below:

   Use the right mouse button to right-click on the **on (rollOver) {** function definition. Use the Copy and Paste options to duplicate the existing function. The resulting script should look like this:



on (rollOver)

☐ Press Key:

☐ Release ☐ Release outside

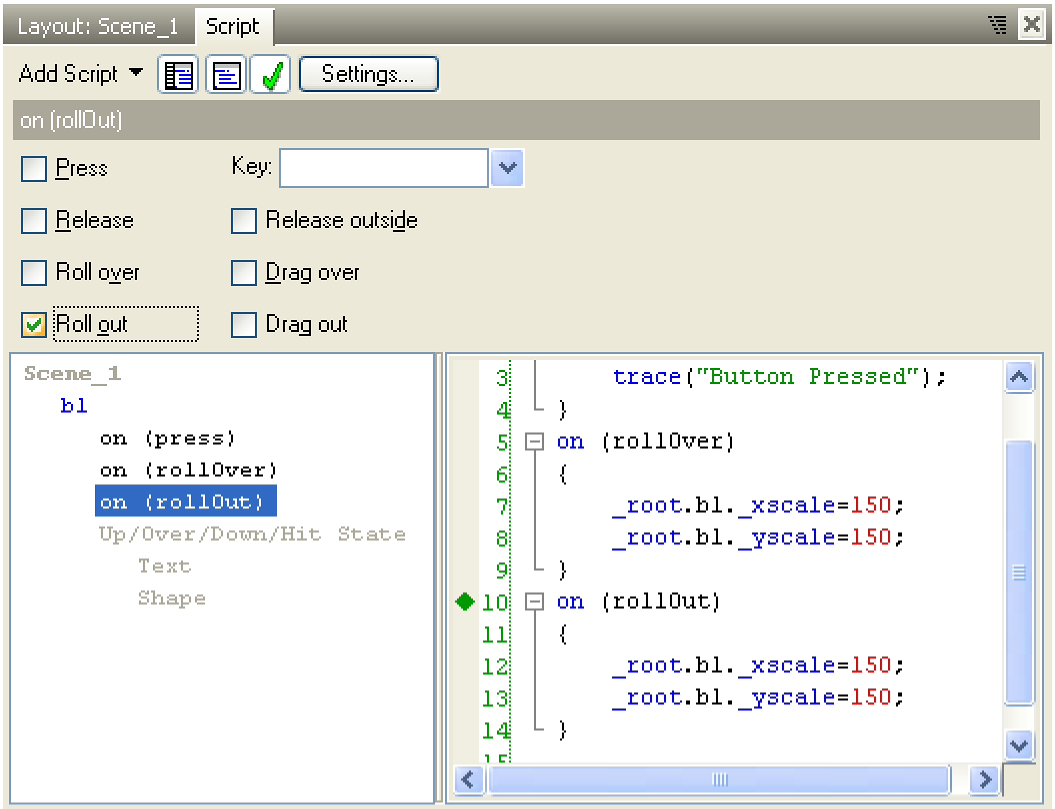
☑ Roll over ☐ Drag over

☐ Roll out ☐ Drag out

```
Scene_1
  bl
     on (press)
     on (rollOver)
     on (rollOver)
     Up/Over/Down/Hit State
        Text
        Shape
```

```
        trace("Button Pressed");
    }
on (rollOver)
    {
        _root.bl._xscale=150;
        _root.bl._yscale=150;
    }
on (rollOver)
    {
        _root.bl._xscale=150;
        _root.bl._yscale=150;
    }
```

Uncheck the 'Roll over' checkbox and check the 'Roll out' checkbox.
The resulting script should look like this:



☐ Press Key:

☐ Release ☐ Release outside

☐ Roll over ☐ Drag over

☑ Roll out ☐ Drag out

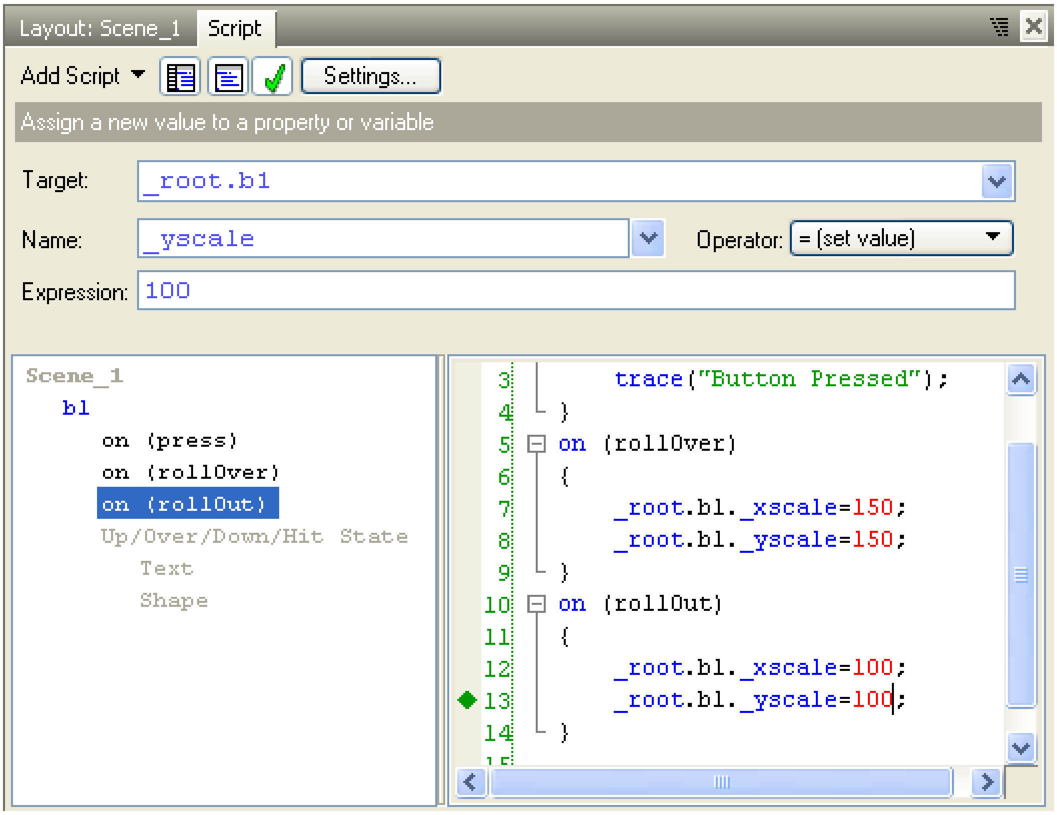
```
Scene_1
  bl
     on (press)
     on (rollOver)
     on (rollOut)
     Up/Over/Down/Hit State
        Text
        Shape
```

```
        trace("Button Pressed");
}
on (rollOver)
{
        _root.bl._xscale=150;
        _root.bl._yscale=150;
}
on (rollOut)
{
        _root.bl._xscale=150;
        _root.bl._yscale=150;
}
```

Now modify the statements contained in the on (rollOut) Event function so that they are:

```
_root.b1._xscale = 100;
_root.b1._yscale = 100;
```

The resulting script should look like this:



Target: _root.b1

Name: _yscale

Operator: = (set value)

Expression: 100

```
Scene_1
   b1
      on (press)
      on (rollOver)
      on (rollOut)
      Up/Over/Down/Hit State
         Text
         Shape
```

```
        trace("Button Pressed");
}
on (rollOver)
{
        _root.b1._xscale=150;
        _root.b1._yscale=150;
}
on (rollOut)
{
        _root.b1._xscale=100;
        _root.b1._yscale=100;
}
```

6. Test the button by select the Debug tab among the Panels displayed on the right hand side of the 'Layout' Panel. Press the 'Play' button to run your Movie. Moving the mouse over the button, should cause the button to become larger. Once the mouse moves away from the button, the button should return to its normal size. Pressing the left mouse button while the mouse is positioned over the button will cause the words "Button Pressed" to be displayed in the 'Debug' Panel (which was covered in the ' Button' tutorial)

Analysis

Additional Event Handling routines were added to the button to handle the on(rollOver) and on(rollOut) Button Events. When the on(rollOver) Event occurs, the button is scaled to 150% of its original size. When the on(rollOut) Event occurs, the button is scaled to 100% of its original size.

The scaling is achieved by changing the _xscale and _yscale properties. As the button is an Object that occurs in the main Movie, its properties can be referenced via _root.b1.<property>.

By design, Button Events are designed to work on the containing Object. Consequently, if the commands in the Button Event routines were entered as this._xscale = ... or simply _xscale = ..., the Effect would be to scale the main Movie, not the button. In contrast, the onSelfEvent() routine works on the current Object.

4.6.7.5 Movie Control

Description:

Demonstrates how to control timelines playing in Movie Clips and how Movie Clips can control the main movie timeline.

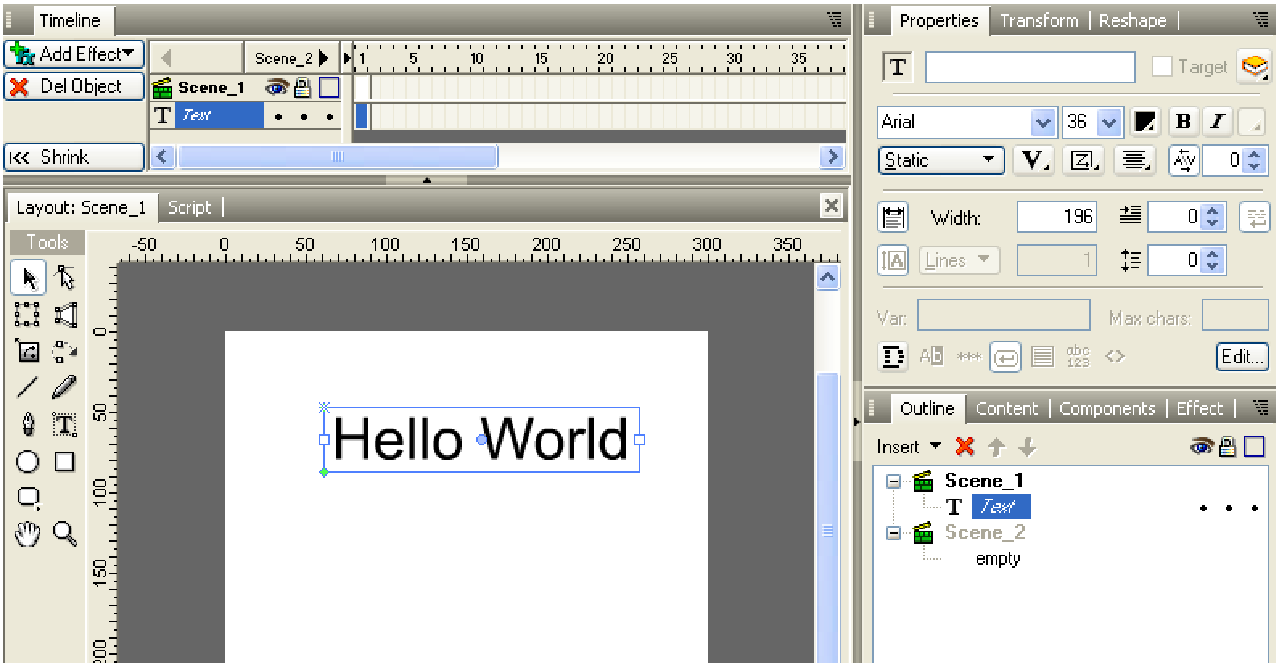
Aim

This tutorial introduces various uses of the gotoAndPlay() actions.

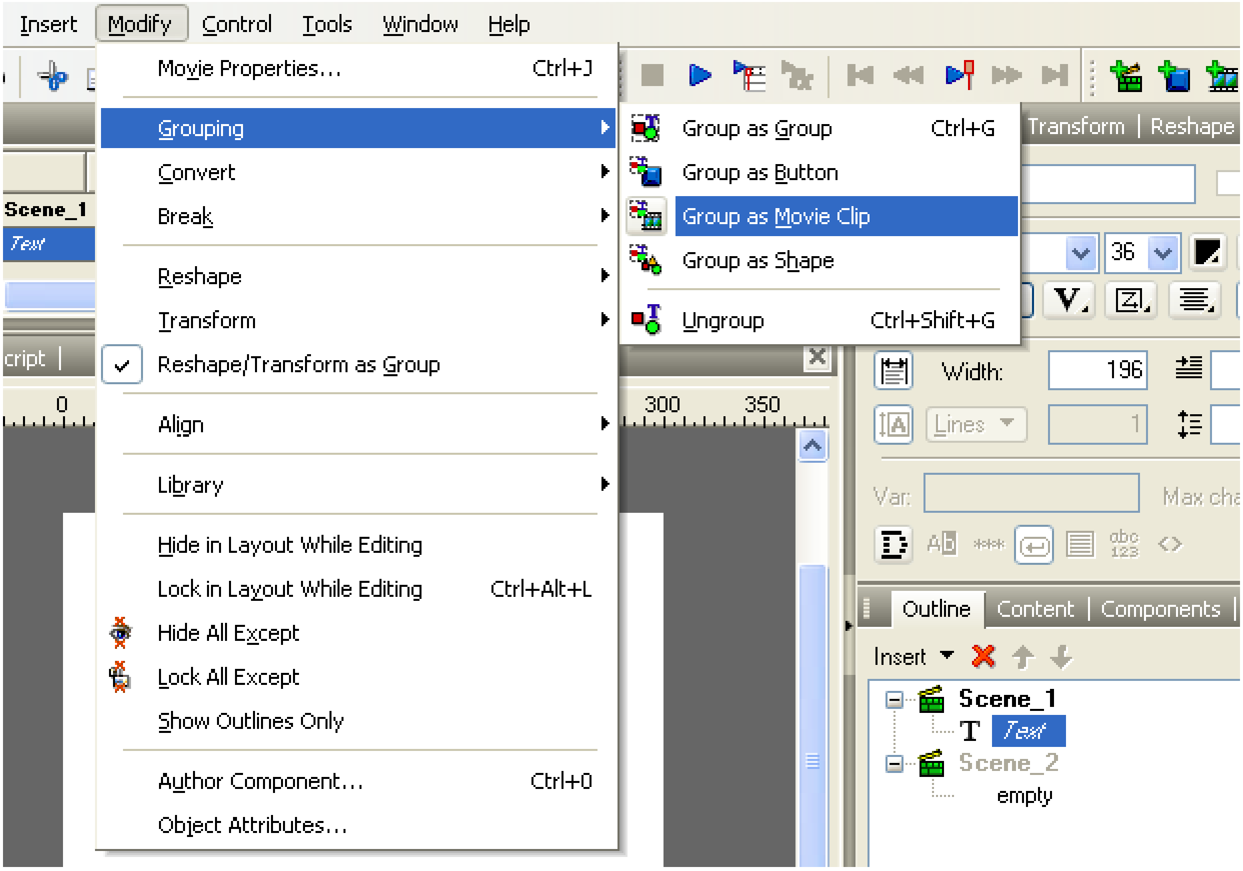
1. Open the file "helloworld.swi" ... save the file as "moviecontrol.swi".

2. On the 'Outline' panel, use the Insert menu and insert a new Scene into the movie - it should be titled Scene_2 by default.

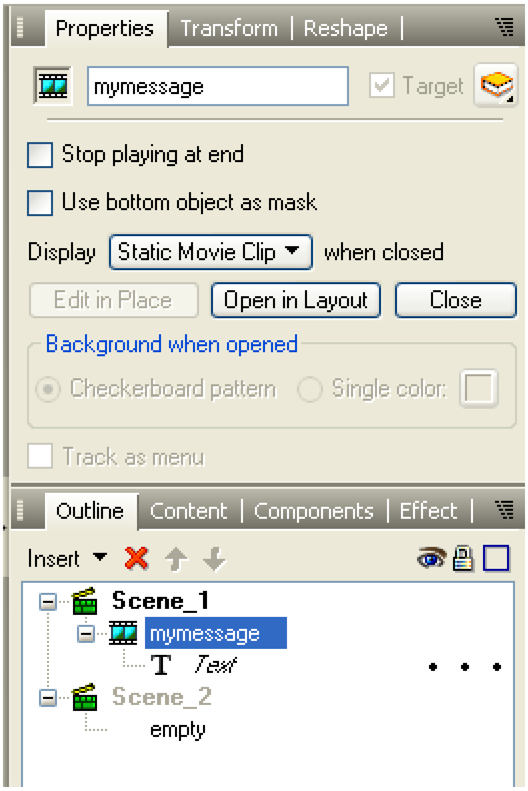
3. Click on "Scene_1" in the 'Outline' panel and insert a Text object into the movie by using the text tool. Change the default text on the text panel to read "Hello World" and make sure that it is a Static text object. Set the text justification to Center ≡ by using the Justification options drop-down menu ≡ on the text panel, and make sure the text is using the Vector character options V as shown below:



4. Make sure the text object is highlighted in the 'Outline' panel, then use the Modify menu and select **Grouping | Group as Movie Clip** (or you can also Right-Click on the text object in the Outline panel and follow the same procedure.



5. Name the Movie Clip "mymessage" on the Properties panel. Make sure to select (highlight) the Movie Clip in the 'Outline' panel and click on the (+) icon to open it - this will allow you to edit the timeline inside the Movie Clip.



Properties | Transform | Reshape

mymessage ☑ Target

□ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip when closed

Edit in Place | Open in Layout | Close

Background when opened

● Checkerboard pattern ○ Single color:

□ Track as menu

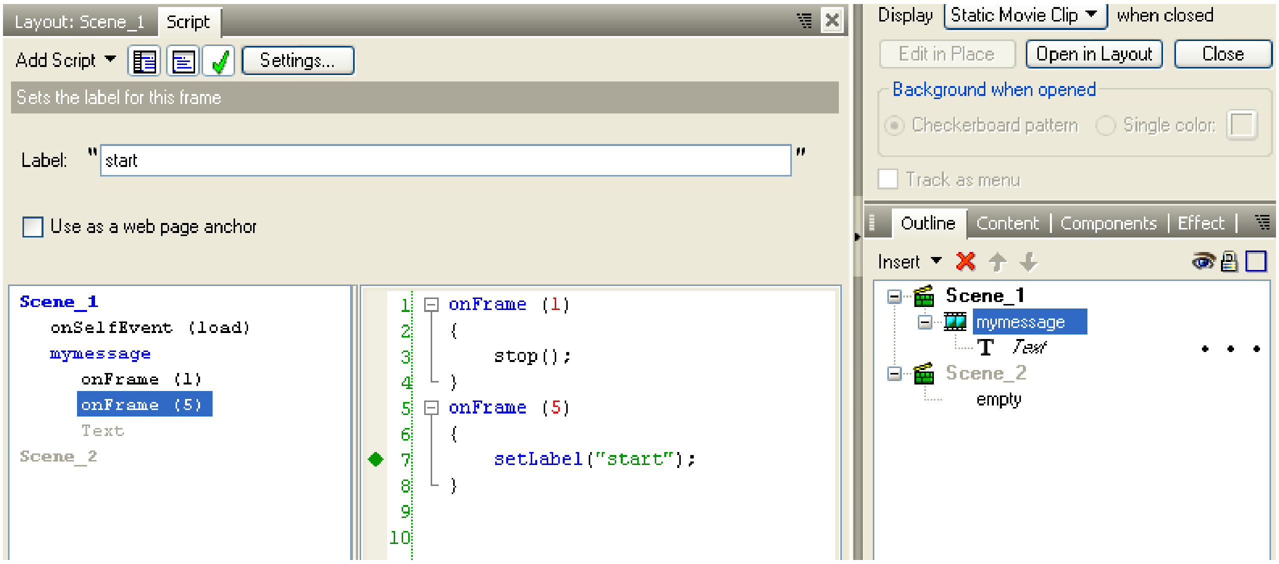
Outline | Content | Components | Effect

Insert

```
Scene_1
  mymessage
    Text
Scene_2
  empty
```

5. Open the 'Script' panel. Press the 'Add Script' button and select **Events | Frame | onFrame(...)** from the menu - leave the frame number set to "1". Next, press the 'Add Script' button again and select **Movie Control | stop()** from the menu.

6. Press the 'Add Script' button again and add another **onFrame()** event and change the frame number to "5". Use the 'Add Script' button again and select **Frame | setLabel()** from the menu and set the label name as "start". The script panel should appear as show below:



Layout: Scene_1 | Script

Add Script | Settings...

Sets the label for this frame

Label: "start"

□ Use as a web page anchor

```
Scene_1
  onSelfEvent (load)
  mymessage
    onFrame (1)
    onFrame (5)
    Text
Scene_2
```

```
onFrame (1)
{
    stop();
}
onFrame (5)
{
    setLabel("start");
}
```

Display Static Movie Clip when closed

Edit in Place | Open in Layout | Close

Background when opened

● Checkerboard pattern ○ Single color:

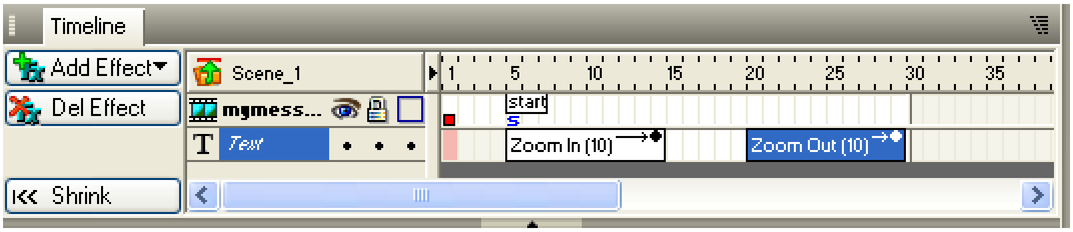
□ Track as menu

Outline | Content | Components | Effect

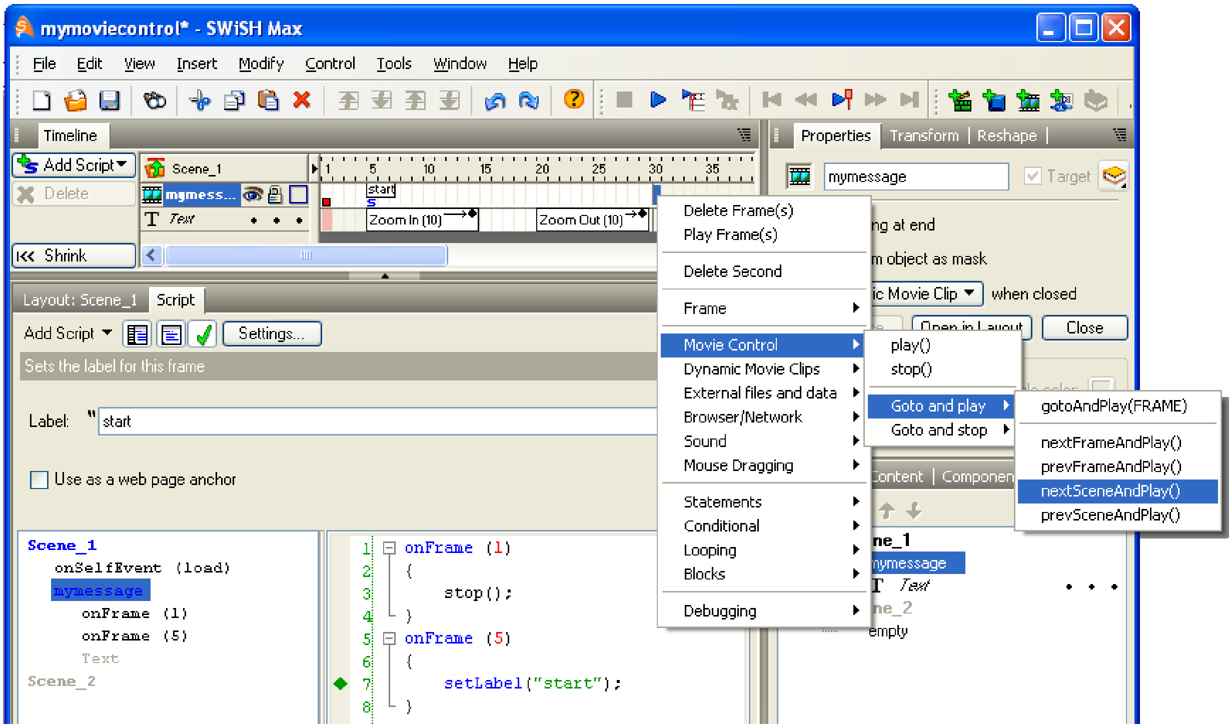
Insert

```
Scene_1
  mymessage
    Text
Scene_2
  empty
```

7. In the 'Timeline' panel, Right-Click at Frame 5 on the row for the "Hello World" text object and select **Zoom | Zoom In** from the menu. Next, Right-Click at Frame 20 on the same row and select **Zoom | Zoom Out** from the menu. The 'Timeline' panel should be shown as below:



8. Right-Click on Frame 30 for the "mymessage" Movie Clip row (directly under the ruler) and select **Movie Control | gotoAndPlay | nextSceneAndPlay()** from the script menu.



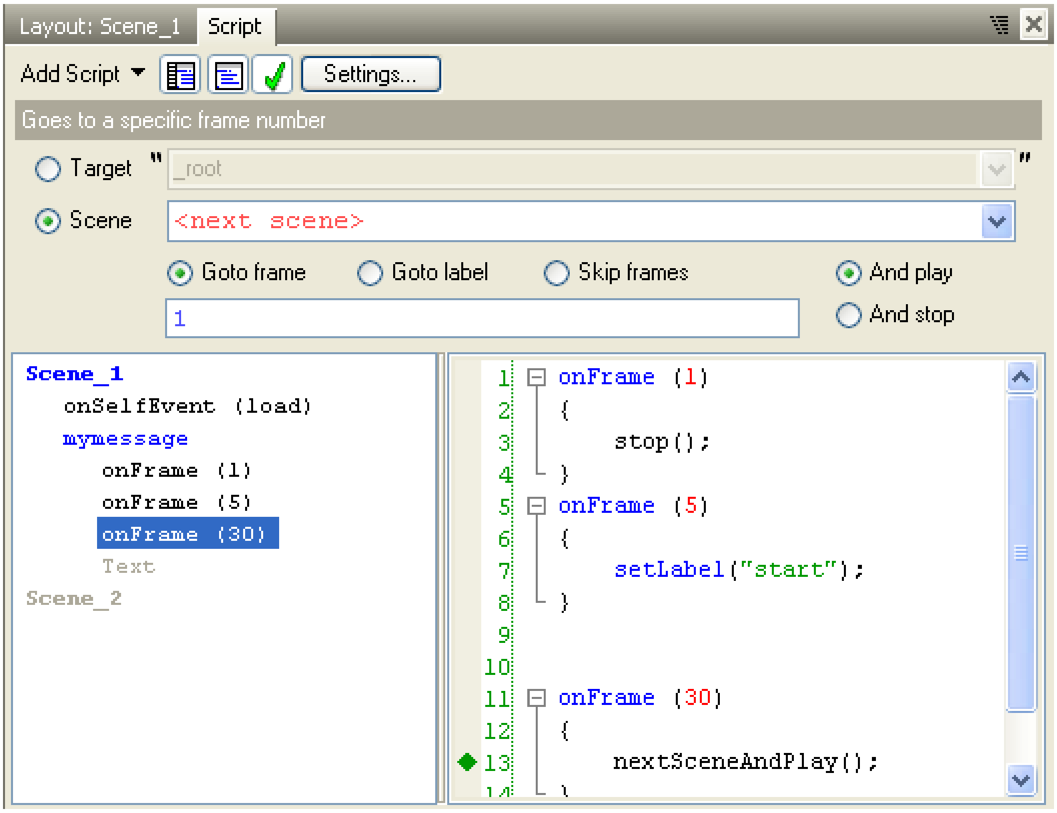
Label: " start

☐ Use as a web page anchor

```
Scene_1
    onSelfEvent (load)
    mymessage
        onFrame (1)
        onFrame (5)
        Text
Scene_2
```

```
onFrame (1)
{
    stop();
}
onFrame (5)
{
    setLabel("start");
}
```

9. The 'Script' panel should be show as below:



Goes to a specific frame number

○ Target "_root"

● Scene <next scene>

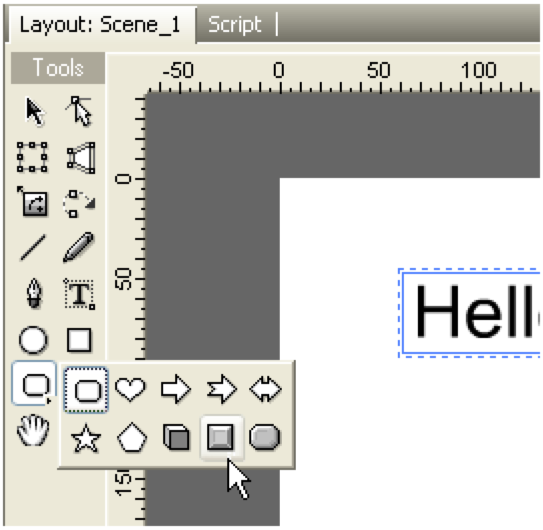
● Goto frame ○ Goto label ○ Skip frames ● And play ○ And stop

1

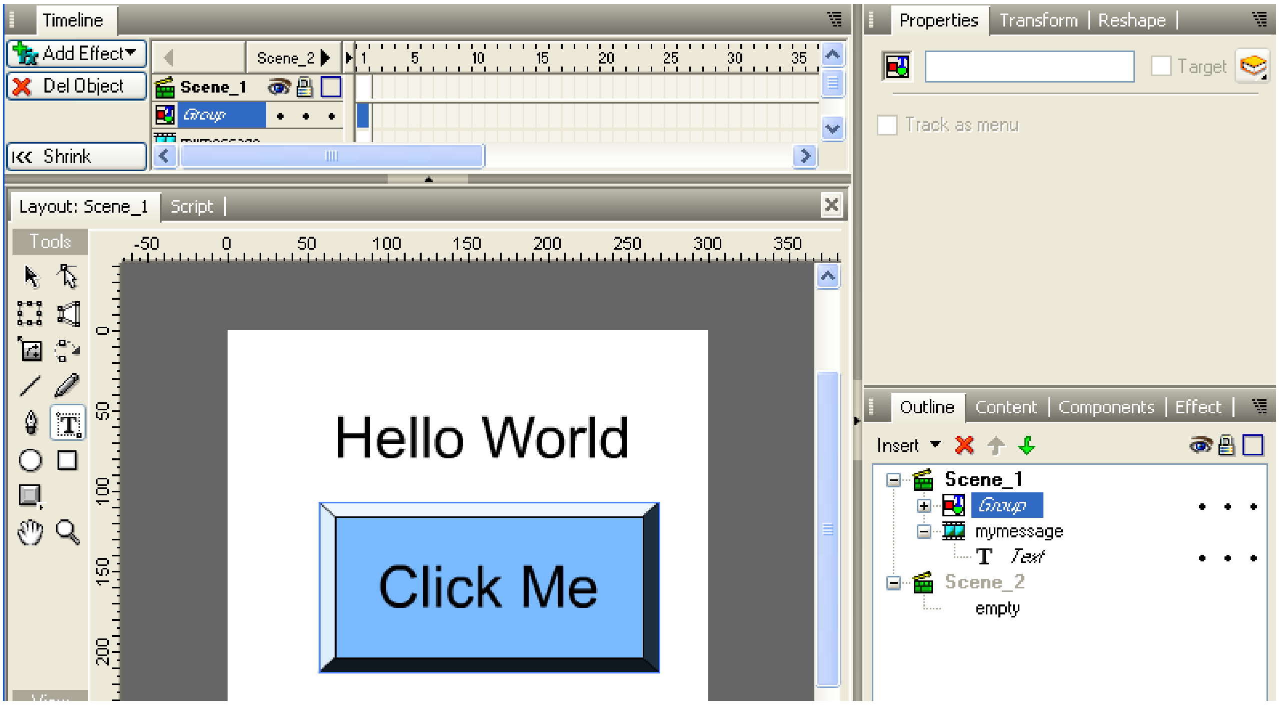
```
Scene_1
    onSelfEvent (load)
    mymessage
        onFrame (1)
        onFrame (5)
        onFrame (30)
        Text
Scene_2
```

```
onFrame (1)
{
    stop();
}
onFrame (5)
{
    setLabel("start");
}


onFrame (30)
{
    nextSceneAndPlay();
}
```

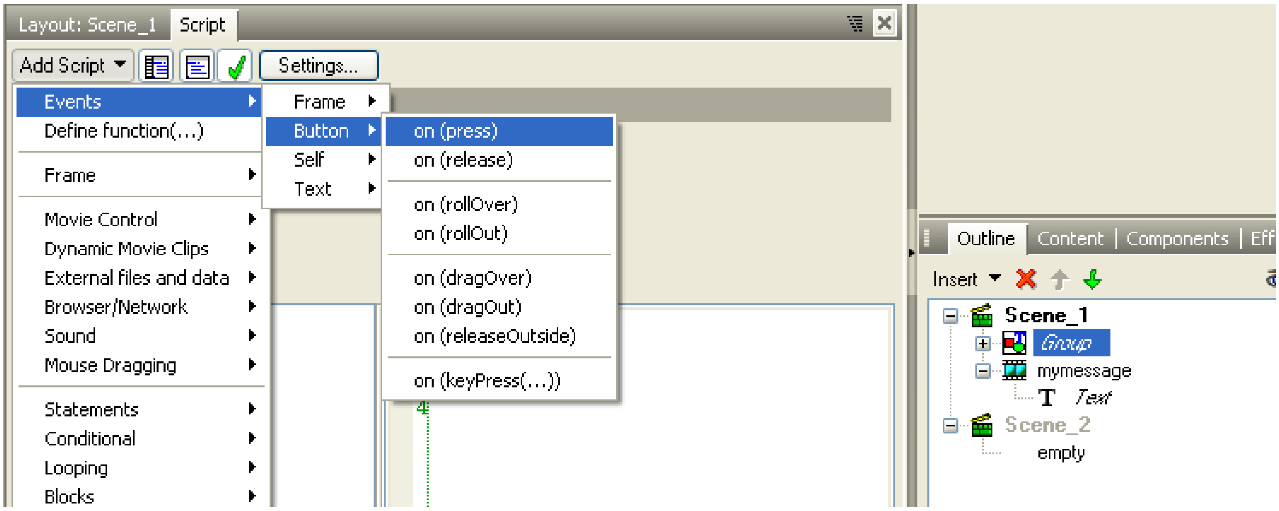
10. Click on "Scene_1" in the 'Outline' panel and click on the AutoShape tool. Draw a Beveled Button shape on the Layout panel below your text object.



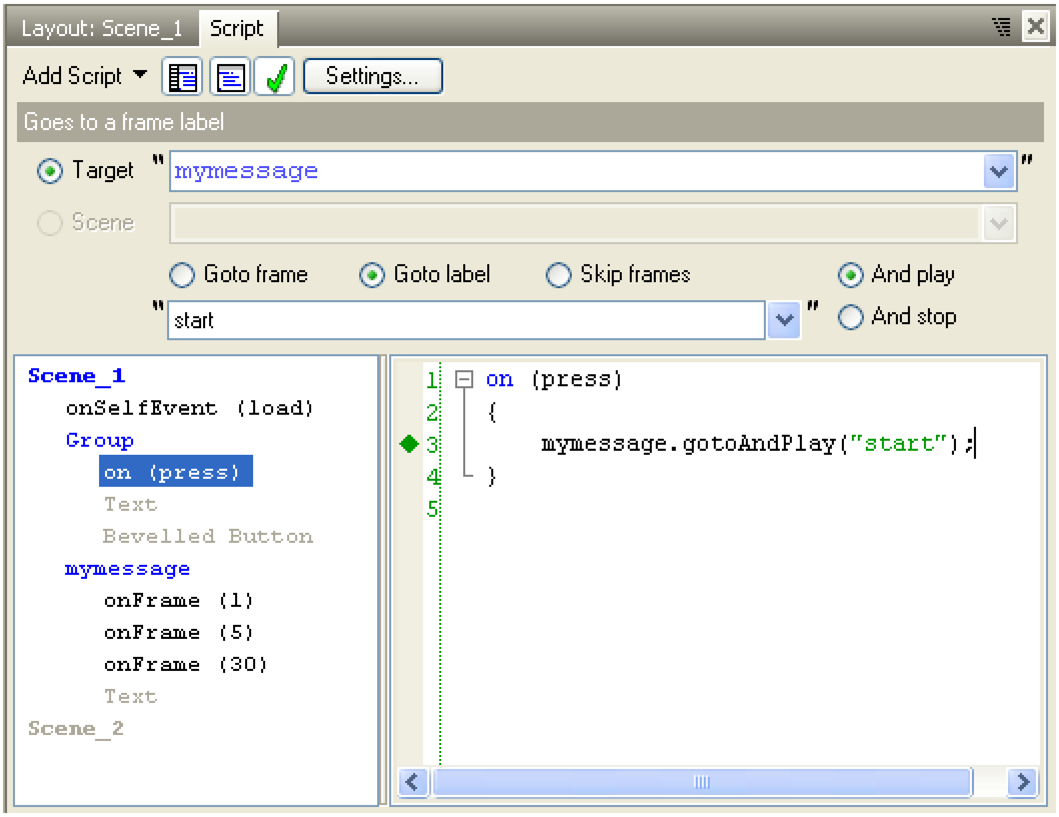
11. Insert a Static text object using the text "Click Me" and position it over top of the beveled button shape. Select both the text object and the beveled button shape (hold down the CTRL key while clicking on each object in the 'Outline' panel) then use the Modify menu and select **Grouping | Group as Group** from the menu.



12. With the Group selected in the 'Outline' panel, open the 'Script' panel and press the 'Add Script' button. Select **Events | Button | on(press)** from the menu.



13. Press the 'Add Script' button again and select Movie Control | gotoAndPlay | gotoAndPlay(LABEL) from the menu. Use the 'Target' drop-down menu and select the "mymessage" Movie Clip from the list then select the "start" label. The resulting script should be shown as below:



● Target "mymessage"

○ Scene

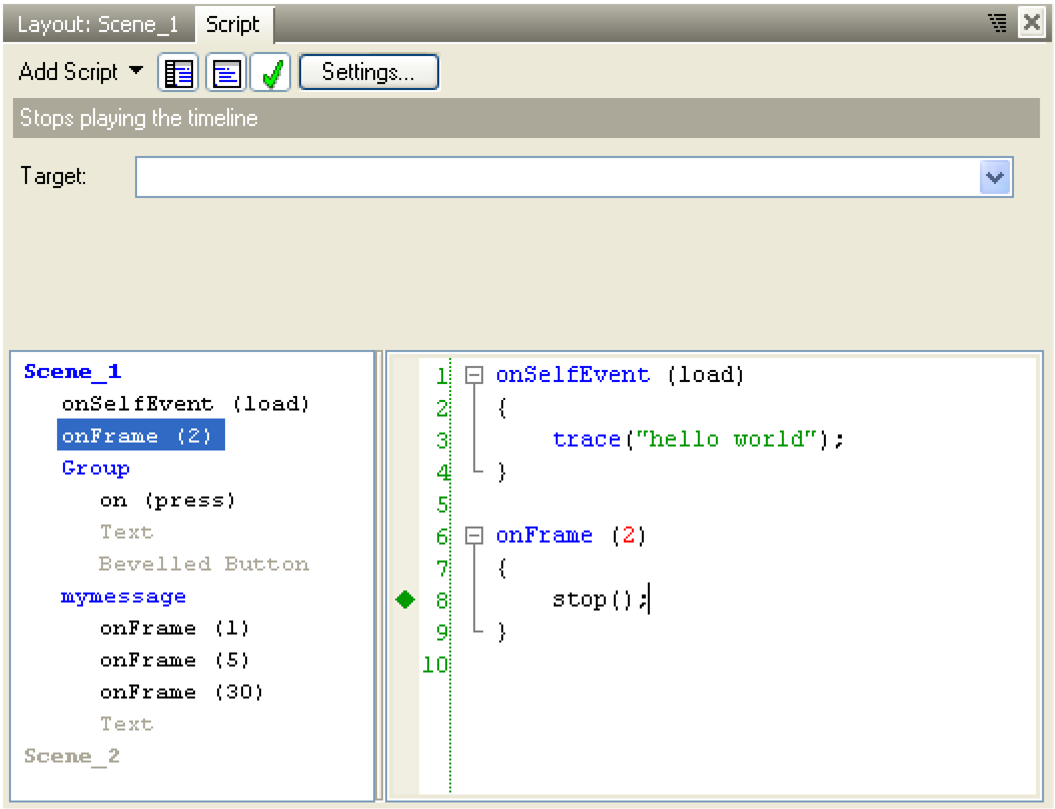
○ Goto frame ● Goto label ○ Skip frames ● And play

"start" ○ And stop

```
Scene_1
    onSelfEvent (load)
    Group
        on (press)
        Text
        Bevelled Button
    mymessage
        onFrame (1)
        onFrame (5)
        onFrame (30)
        Text
Scene_2
```

```
on (press)
{
    mymessage.gotoAndPlay("start");
}
```

14. Select "Scene_1" on the 'Outline' panel and add a stop() action on Frame 2 (using the methods described above).



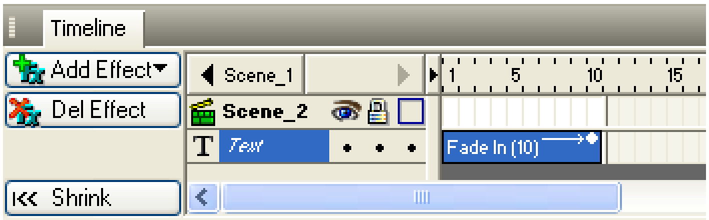
Stops playing the timeline

```
Scene_1
    onSelfEvent (load)
    onFrame (2)
    Group
        on (press)
        Text
        Bevelled Button
    mymessage
        onFrame (1)
        onFrame (5)
        onFrame (30)
        Text
Scene_2
```

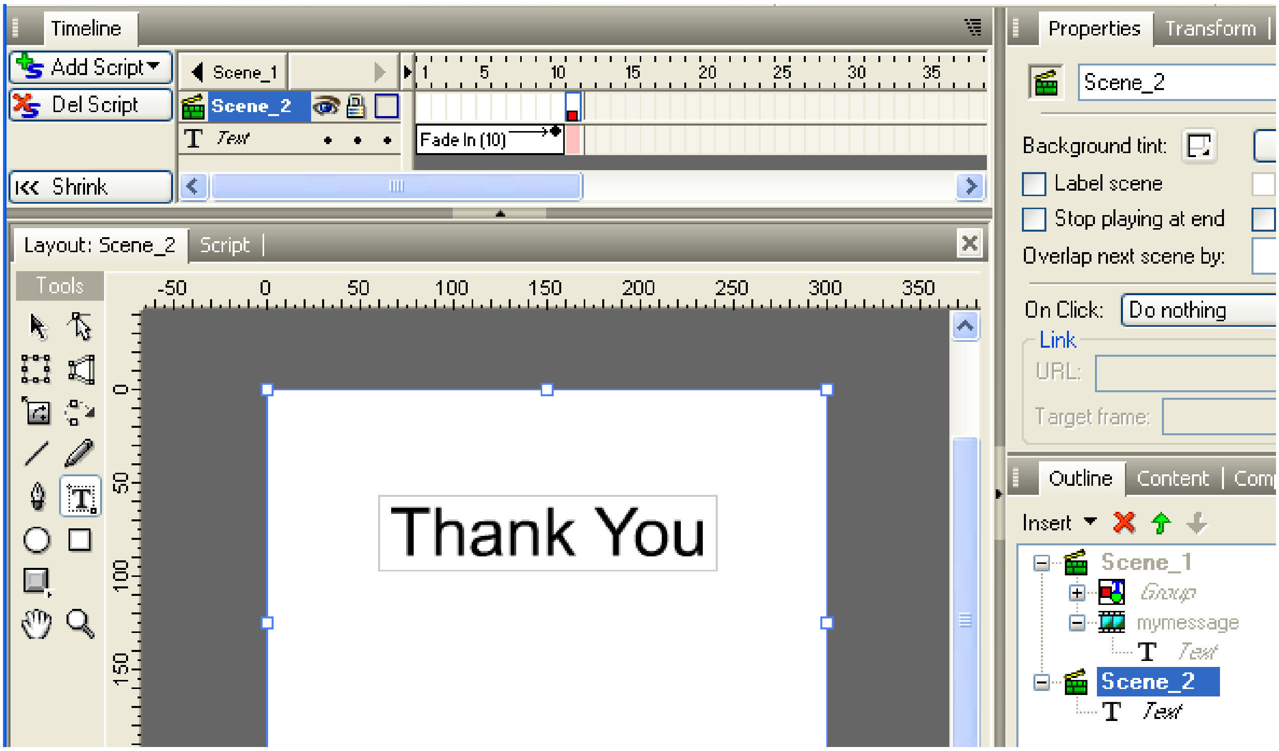
```
onSelfEvent (load)
{
    trace("hello world");
}

onFrame (2)
{
    stop();
}
```

15. Select "Scene_2" on the 'Outline' panel and insert a Static Text object. Change the text to read "Thank You".

16. In the 'Timeline' panel, Right-Click at Frame 1 on the row for the "Thank You" text object and select **Fade | Fade In** from the menu. The 'Timeline' panel should be shown as below:



17. Right-Click at Frame 11 on the row for "Scene_2" in the 'Timeline' panel and select **Movie Control | stop()** from the menu.



18. Make sure you are in the 'Layout' panel then press the ▶ 'Play Movie' button from the Control Toolbar. When the movie starts playing, press the "Click Me" button. You should see the text "Hello World" zoom in/out then the text "Thank You" will fade in.

19. Press the 'Stop' movie button and save the .swi file.

Analysis
When the "Click Me" button is pressed, it sends a command to the Movie Clip "mymessage" telling it to go to the label "start". At the label the text animation begins, and when the animation stops (at Frame 30) there is an action inside the Movie Clip telling the main timeline to go to the next Scene and play. By using the gotoAndPlay() action - you can set different Targets to send commands from the main timeline to a Movie Clip and from the Movie Clip back to the main timeline.

4.6.7.6 Calculator

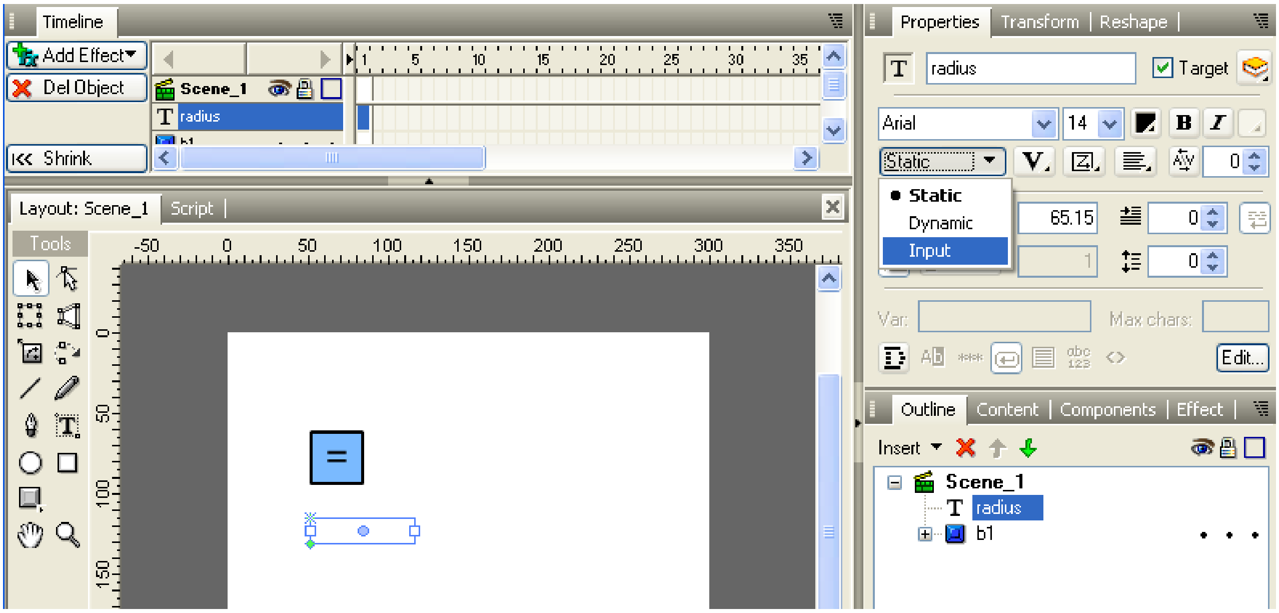
Description
This is a step-by-step tutorial for creating a simple calculator that calculates the area of a circle, given the radius of the circle (PI * r * r).

Aim
This tutorial introduces the Text Object with Input and Dynamic text properties and the Math Object. It also uses the on(press) Button Event developed in the previous tutorial.

.swi file
"calculator.swi"

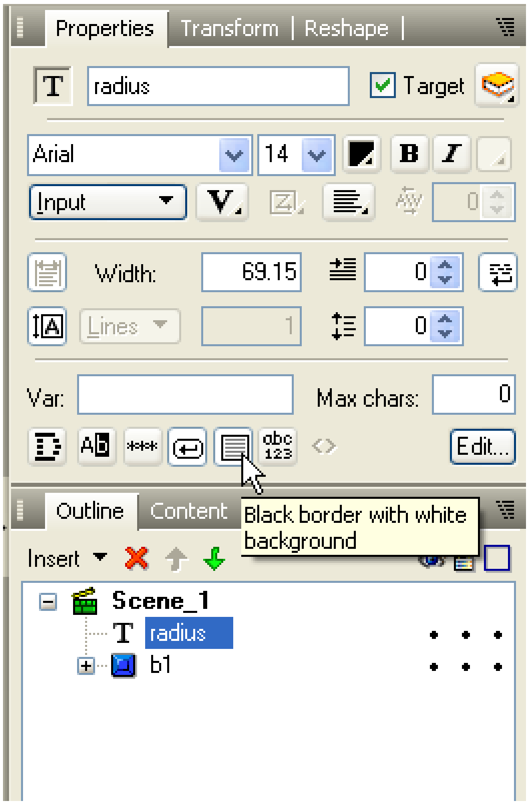
1. Continue from the Button with rollOver tutorial or load the file "button1.swi" and save as "mycalculator.swi"

2. Insert a Text Object using the Text Tool or the Menu items Insert | Text. Name the new Text Object radius, tick the 'Target' checkbox and set to the text type to Input in the Text Panel

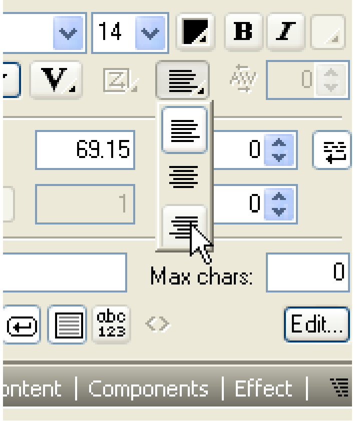


3. Select the "Advanced Options" for this Text Object and set the option for 'Black Border with White Background' ▤ as shown below

Note: The handles around the Text Object change once the type is changed to an Input Text Object



4. Select right justification, as shown below:


14

69.15

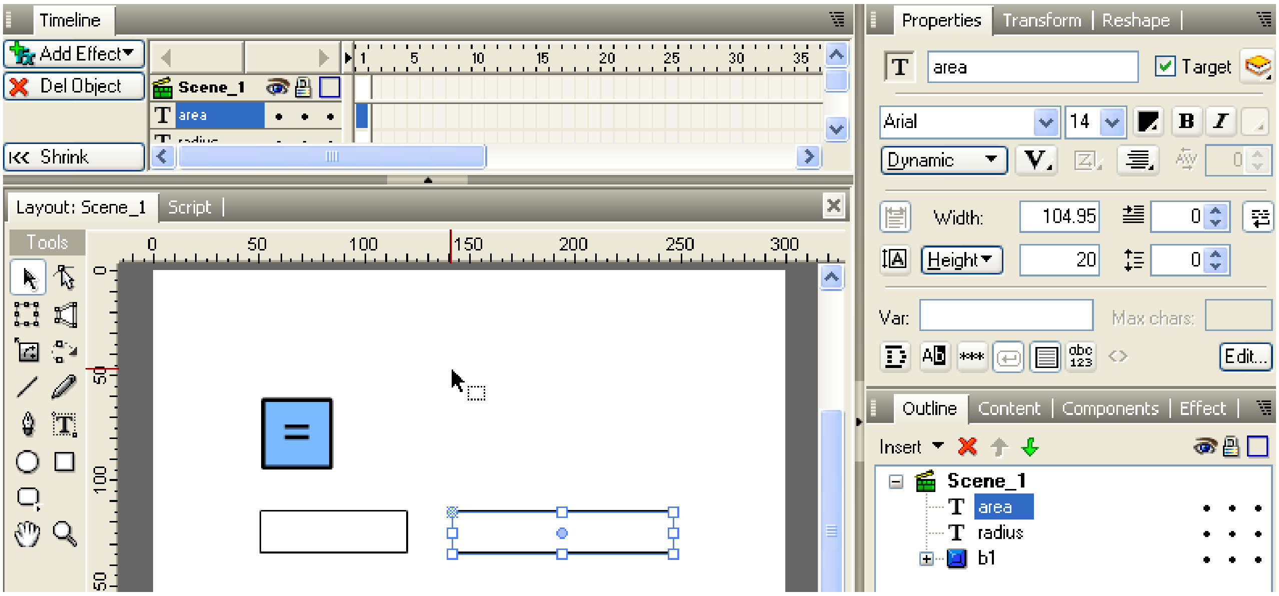
0

1

0

Max chars: 0

5. Create a Dynamic Text Object to display the result of the calculation by using the Text Tool or Insert | Text to insert a Text Object. Name the new Text Object "area", tick the 'Target' checkbox and set to the text type to Dynamic in the Text Panel. Clear the text so that an empty text field is displayed and select right justification. Use the Selection Tool to stretch the text field using the cross-shaped handles, as shown below. The advanced settings will be the same as those selected for the previous Text Object and do not need to be altered at this stage


area ☑ Target

Arial 14

Dynamic

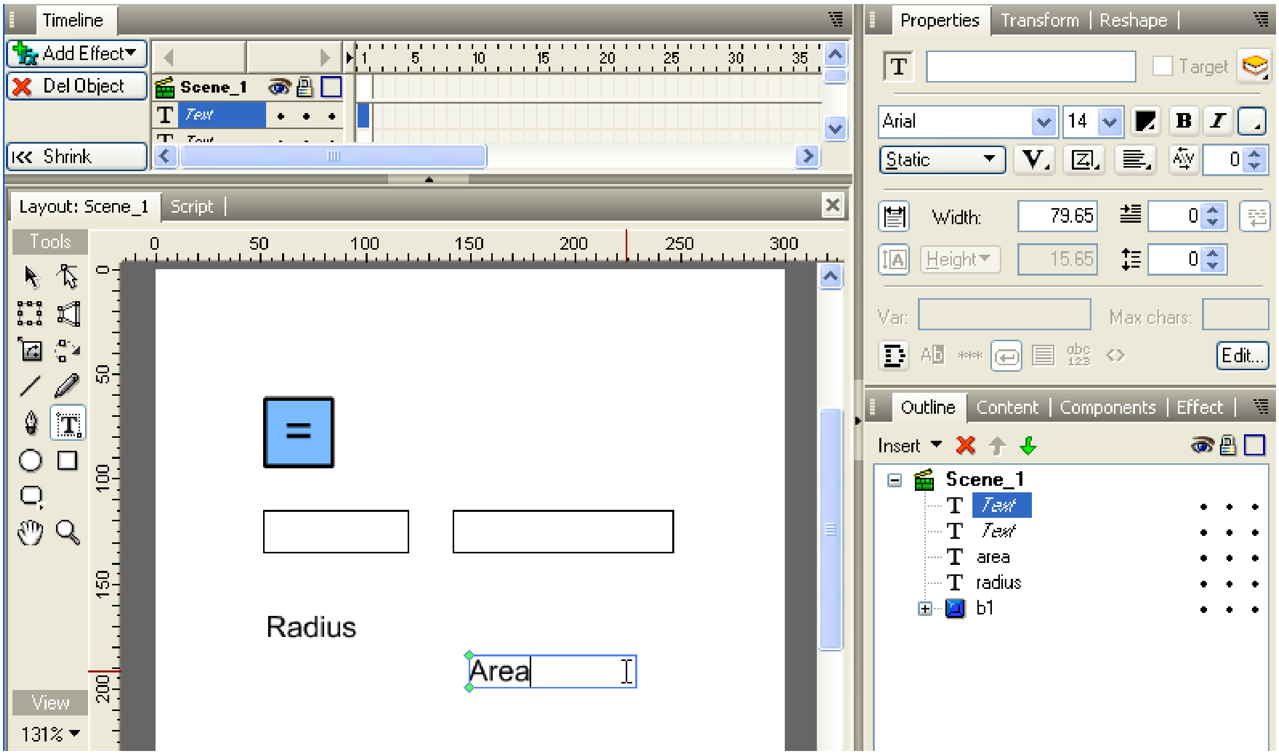
Width: 104.95 0

Height 20 0

Var:

Max chars:

6. Use the Text Tool or Insert | Text to insert some Static text ("area" and "radius") to be used as headings



Arial 14 Static

Width: 79.65

Height 15.65

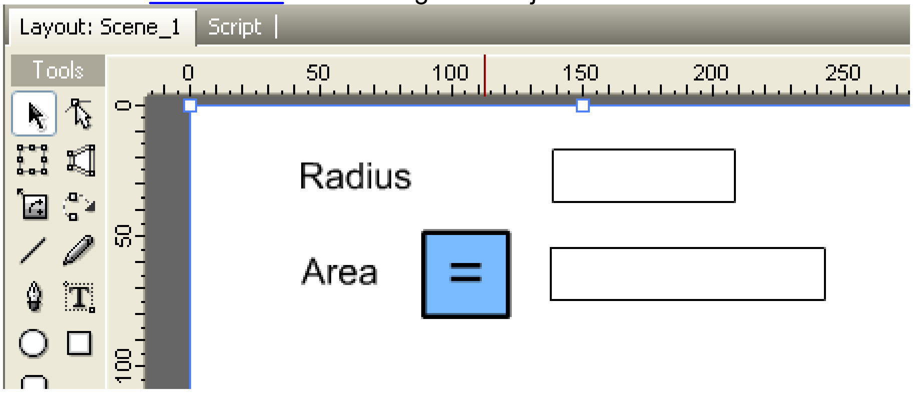
Var:

Max chars:

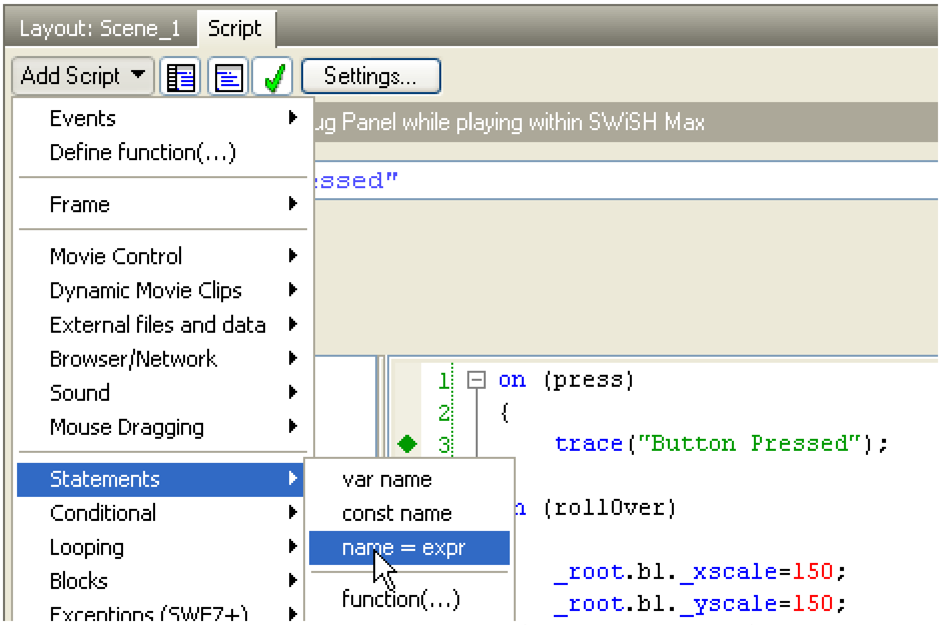
Note:

- If the text has a line border around it, use the 'Advanced...' button to turn the border off
- Although the new Text Objects appear in the 'Outline' Panel with the same names as the previously inserted Text Objects, a conflict will not occur at the Scripting level as the new Text Objects are not Scripting Objects (they are unnamed and do not have the 'Target' checkbox set)

7. Use the Select Tool to re-arrange the Objects as shown below:



8. Modify the on(press) Button Event function to calculate the area of the circle when the button is pressed. Select the button then select the 'Script' Panel. Position the cursor at the end of the trace statement then use the Add Script button and select Statements | name = expr;

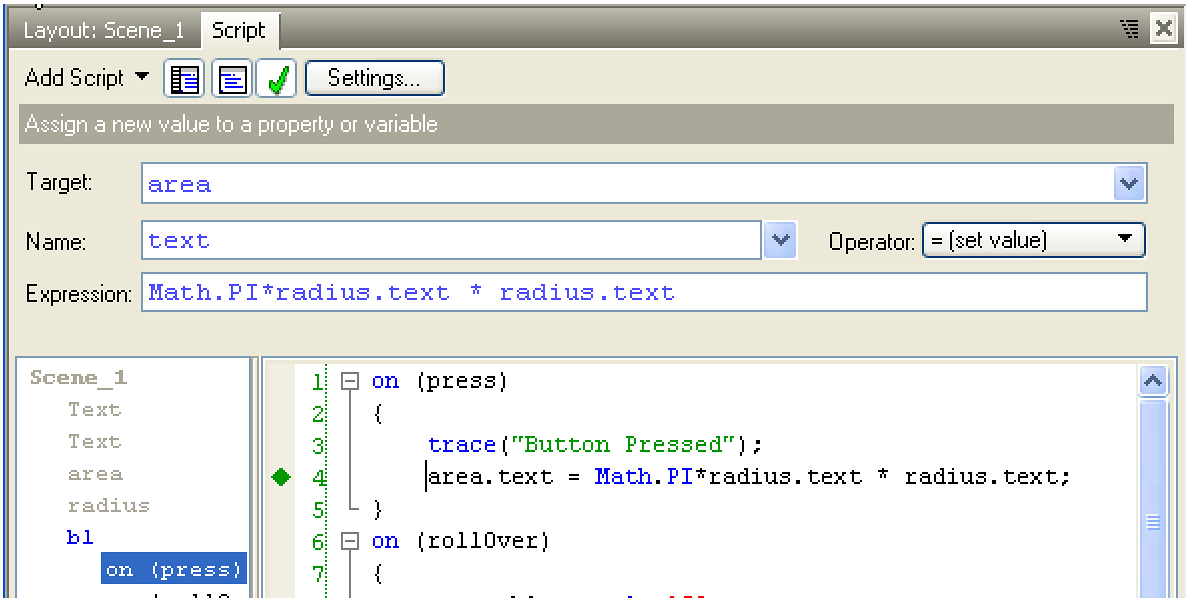


```
:ssed"
```

```
on (press)
{
    trace("Button Pressed");

n (rollOver)

    _root.bl._xscale=150;
    _root.bl._yscale=150;
```

For the 'Target:' parameter, select area (the Dynamic Text Object added in step 5) from the pull-down list.
For the 'Name:' parameter, select Text | text from the pull-down list.
Leave the operator as "=".
Enter the formula "Math.PI * radius.text * radius.text" in the formula area

Note: "Math.PI" is case sensitive. "PI" could also have been used



Target: area

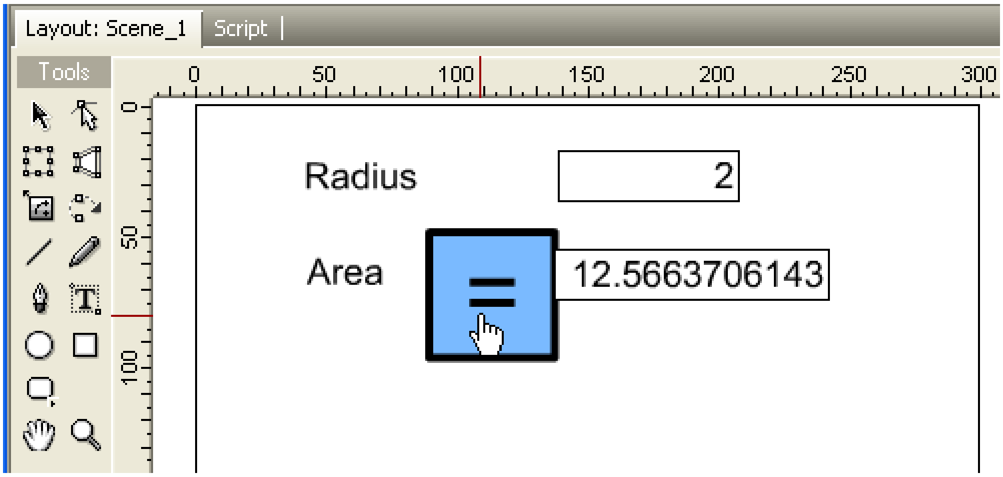
Name: text

Operator: = (set value)

Expression: Math.PI*radius.text * radius.text

```
on (press)
{
    trace("Button Pressed");
    area.text = Math.PI*radius.text * radius.text;
}
on (rollOver)
{
```

9. Test the calculator by reselecting the 'Layout' Panel and press the 'Play' button (fonts may appear to change because system fonts are used). Enter the number 2 into the 'radius' text field and click the '=' button. The following should be displayed in the 'Layout' Panel.



Analysis
The input Text Object radius is a Scripting Object. It is used to allow the entry of the input parameter, radius, the radius of a circle.
Within the script, the input value can be accessed via the name "radius.text". "radius" is the name of the Object, "text" is the property that contains the entered value.

When the button is pressed, the script contained within the button on(press) Event Handler is executed. This results in the calculation Math.PI * radius.text * radius.text being placed into "area.text", the text property of a Dynamic Text Object named "area". This results in the display of the calculated value.

Note: The area of a circle is given by the formula "PI * r * r"

4.6.7.7 Object Properties

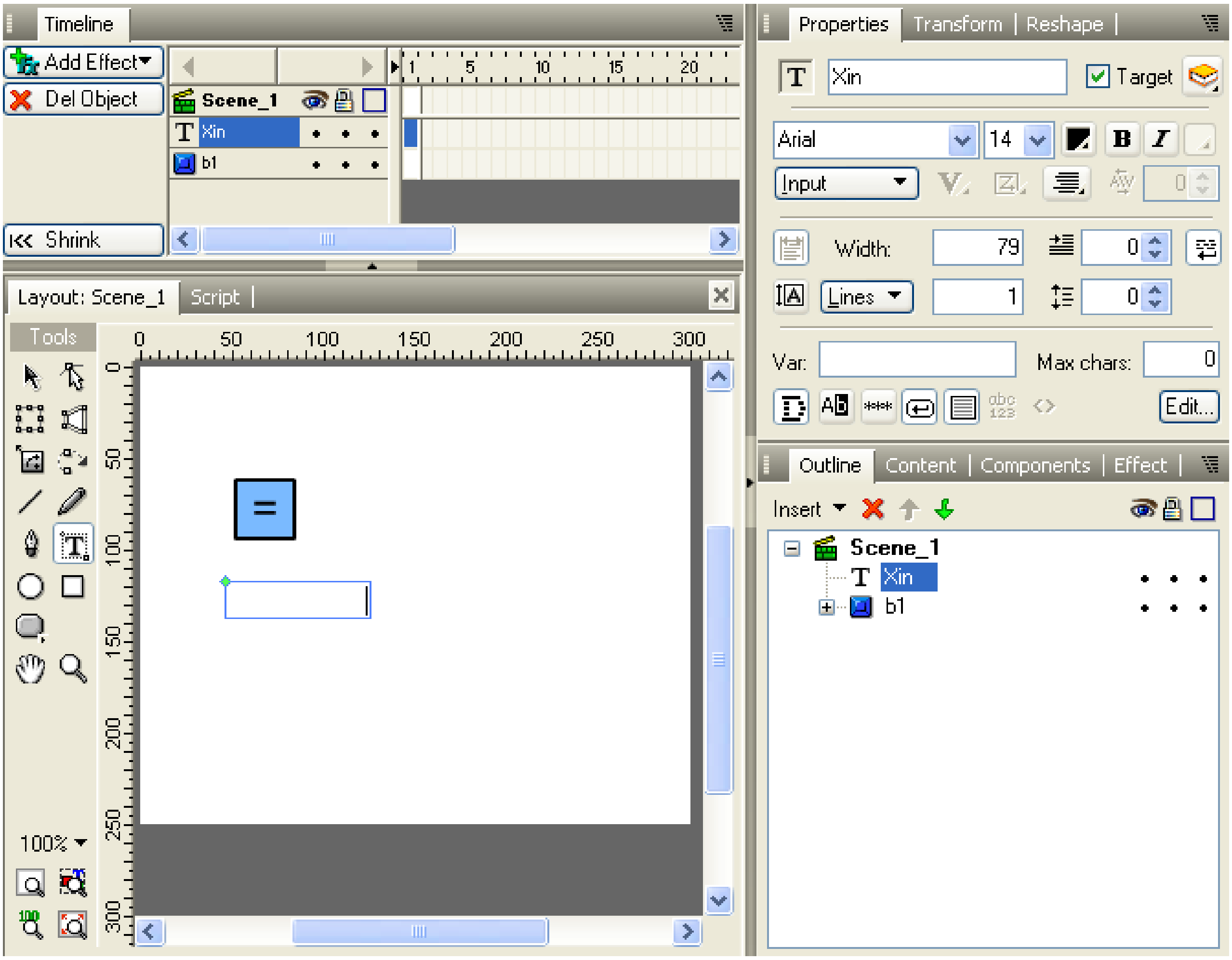
Description
This is a step-by-step tutorial demonstrating how to manipulate the properties of Text Objects.

Aim
This tutorial uses multiple Text Objects with Input and Dynamic text properties and introduces the Object properties:
_x, _y
_xscale, _yscale
_width, _height
_alpha
_rotation
_name
_visible.

It also uses the on(press) Button Event developed in the previous tutorial.

.swi file
"properties.swi"

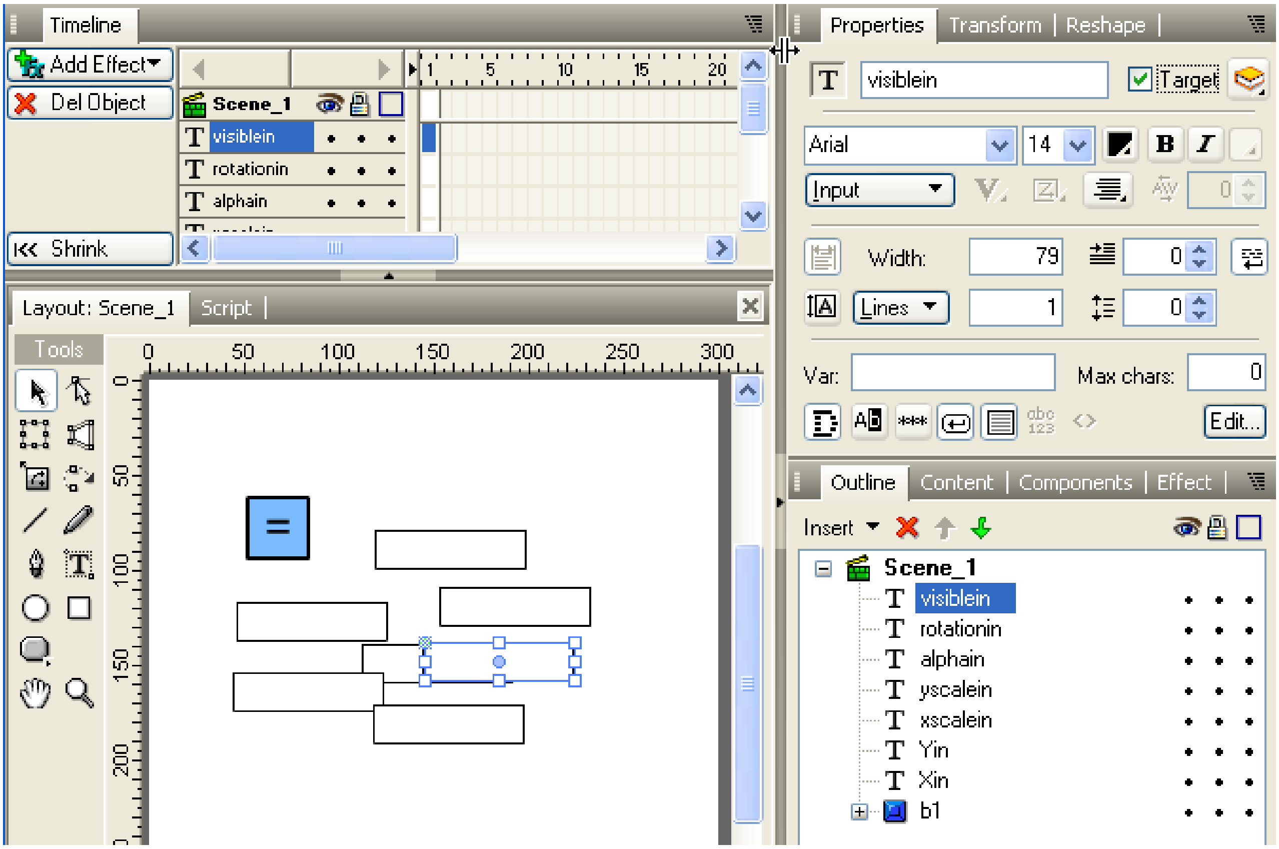
1. Load file "button1.swi". Create an Input Text Object called Xin. (See Calculator tutorial for more detail). Save the file as "myproperties.swi".



Xin ☑ Target
Arial 14
Input
Width: 79 0
Lines 1 0
Var: Max chars: 0

Select the right-justified text option, Black border with white background and dynamic text options.

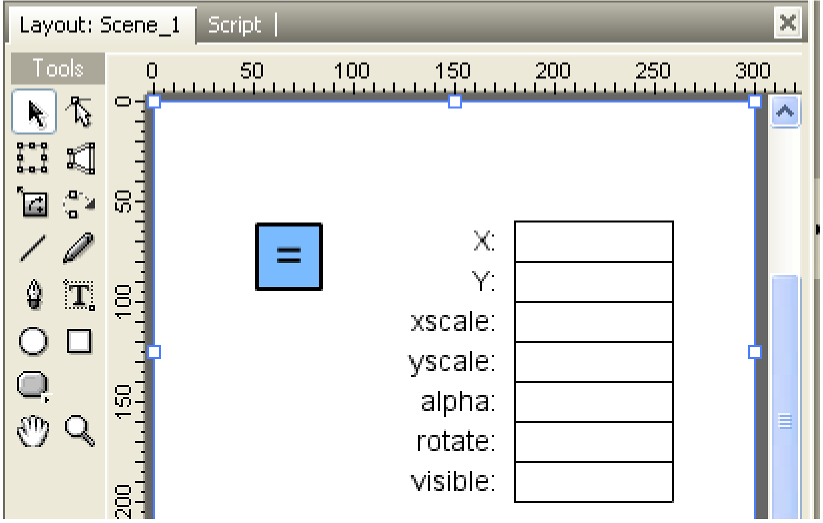
2. Create similar Text Objects called Yin, xscalein, yscalein, alphain, rotationin, visiblein. You can create the new Objects using the Text Tool, or by copying and pasting the Xin Text Object and renaming it



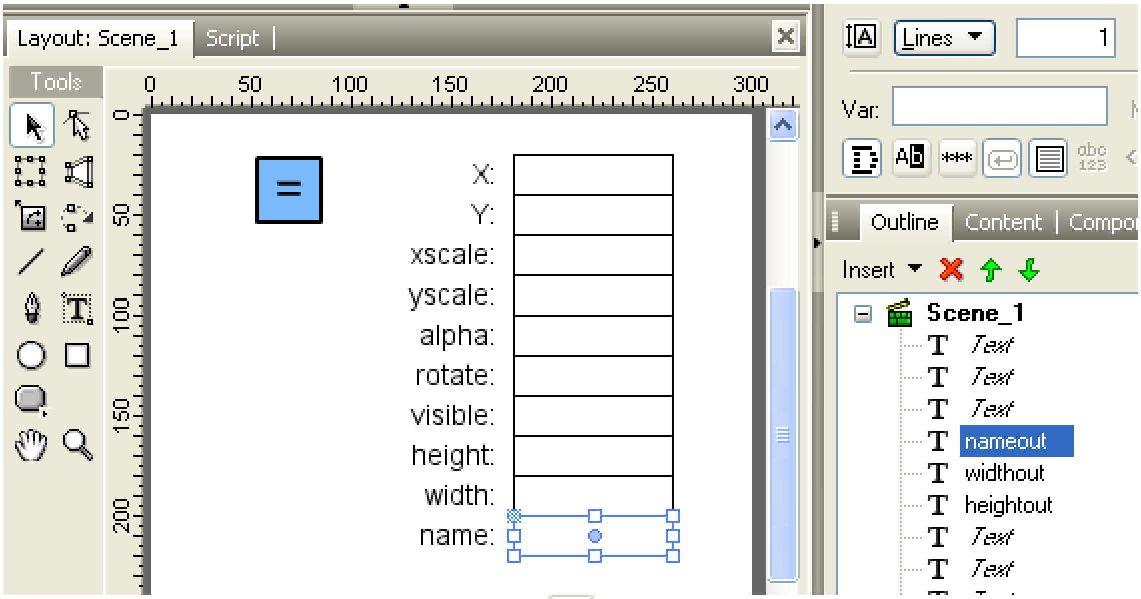
visiblein ☑ Target
Arial 14 B I
Input
Width: 79 0
Lines 1 0
Var: Max chars: 0

Scene_1
- visiblein
- rotationin
- alphain
- yscalein
- xscalein
- Yin
- Xin
- b1

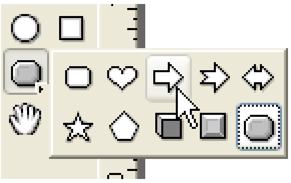
3. Add static text titles for the input boxes and re-arrange as shown below. You may find this easier if you enable Snap To Grid



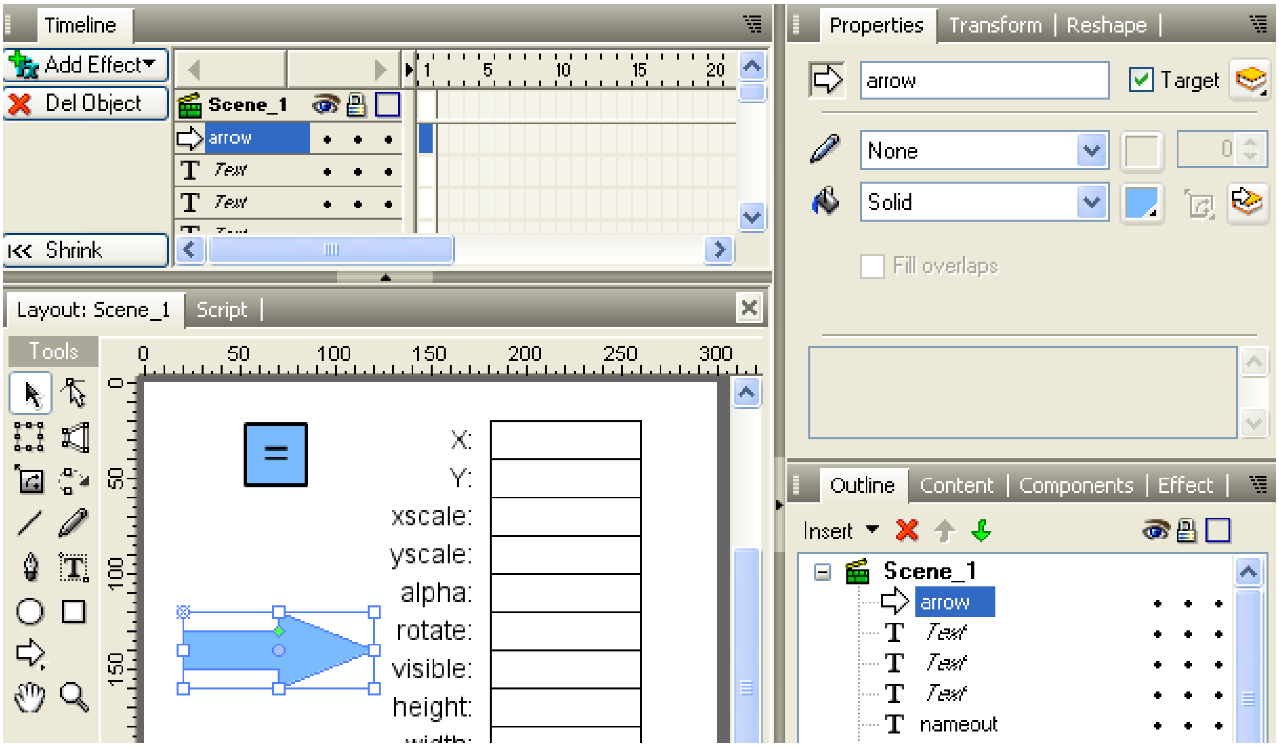
4. Add Dynamic Text Objects heightout, widthout and nameout. Add corresponding static text titles for the new fields.



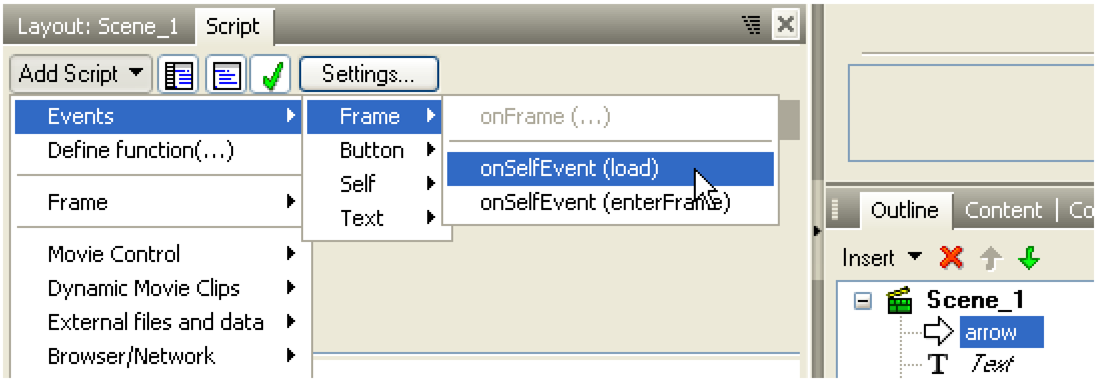
5. Use the Autoshape Tool to draw an arrow ⇨ as shown below. You may have to position the mouse over the ▢ Autoshape icon in the Toolbar and hold the left mouse button to see the selection Menu that allows selection of the arrow shape ⇨. Name the object "arrow" and tick the 'Target' checkbox.



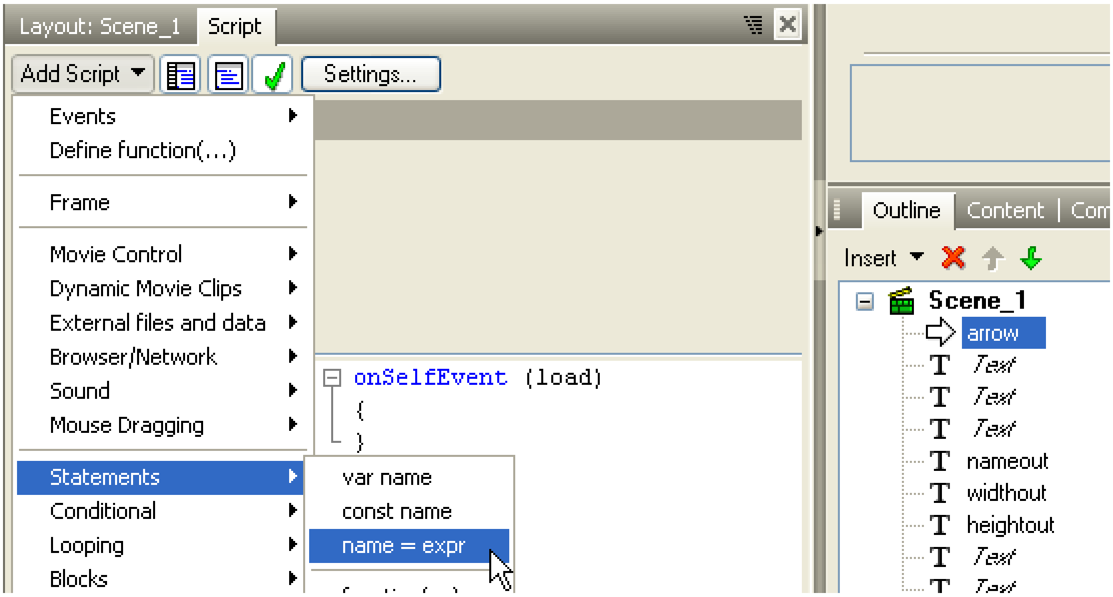
Note: After naming the Object, you will have to move away from the 'Name:' field to allow the 'Target' checkbox to appear. The Object MUST be named and have its 'Target' checkbox ticked for it to become a Scripting Object



6. Select the arrow shape in the 'Outline' Panel and select the Script tab to add an onSelfEvent (load) Event Handling function for the arrow. This function will be used to initialise the Input and Dynamic text values

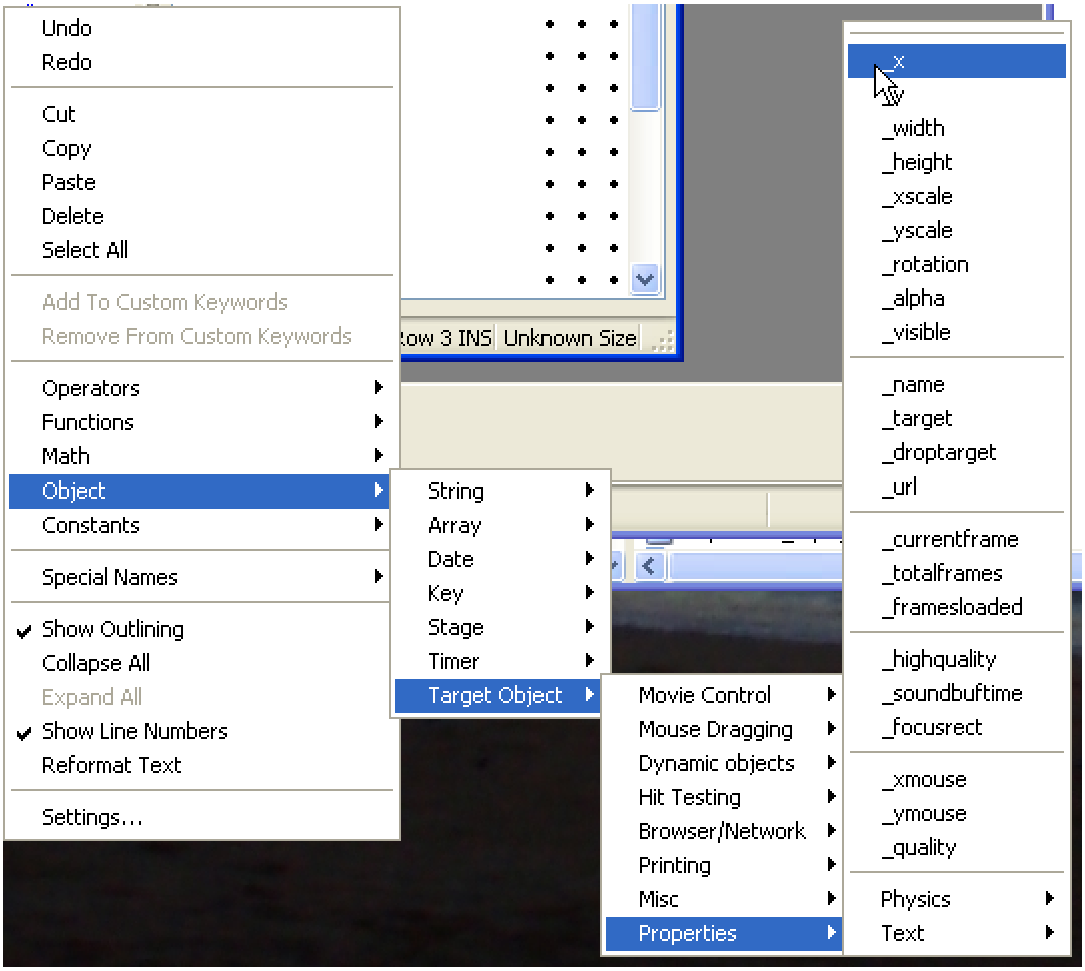


7. Add a **name=expr;** type statement to the **onSelfEvent (load)** Event function.

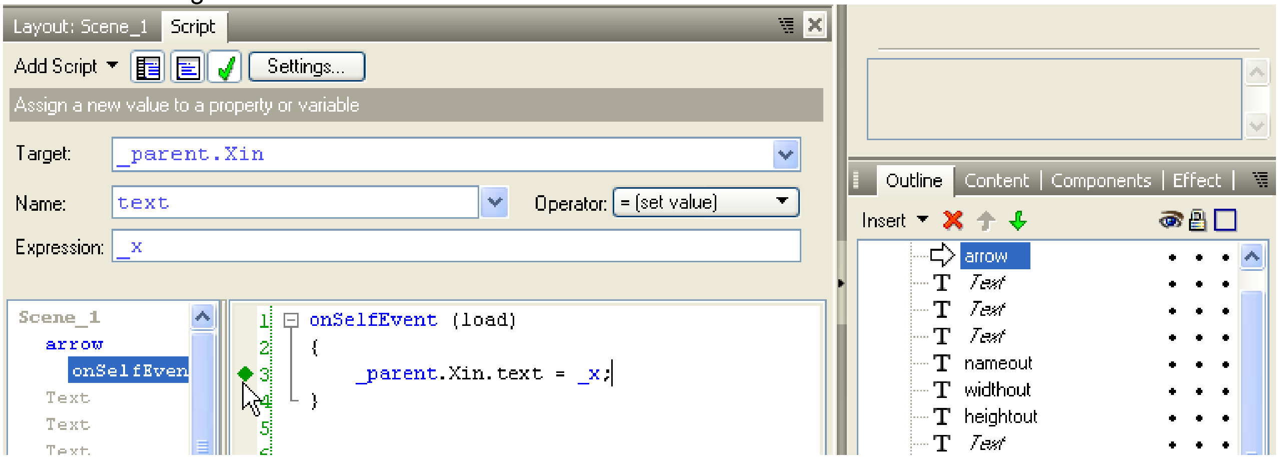


```
onSelfEvent (load)
{
}
```

For the 'Target:' parameter, select _parent.Xin from the pull-down list.
For the 'Name:' parameter, select Text | text from the pull-down list.
Leave the operator as =.
Enter _x into the Expression field.
Alternatively position the cursor on the right hand side of the equals sign in the script editor and use the right mouse button. Select the _x property from the **Object | Target Object | Properties | _x** context menu.



The resulting function should look like this:



Assign a new value to a property or variable

Target: _parent.Xin

Name: text   Operator: = (set value)

Expression: _x

```
onSelfEvent (load)
{
    _parent.Xin.text = _x;
}
```

8. In a similar way, expand the onLoad function so that it contains the following statements:

```
_parent.Xin.text = _x;
_parent.Yin.text = _y;
_parent.xscalein.text = _xscale;
_parent.yscalein.text = _yscale;
_parent.alphain.text = _alpha;
_parent.rotationin.text = _rotation;
_parent.visiblein.text = _visible;
_parent.heightout.text = _height;
_parent.widthout.text = _width;
_parent.nameout.text = _name;
```

Note:
You may find it easier to enter one of the statements and then use Copy and Paste to expand that to multiple statements. After entering the multiple statements, edit each individual statement to contain the required data.
You could also copy and paste the code above into the Script editor panel.

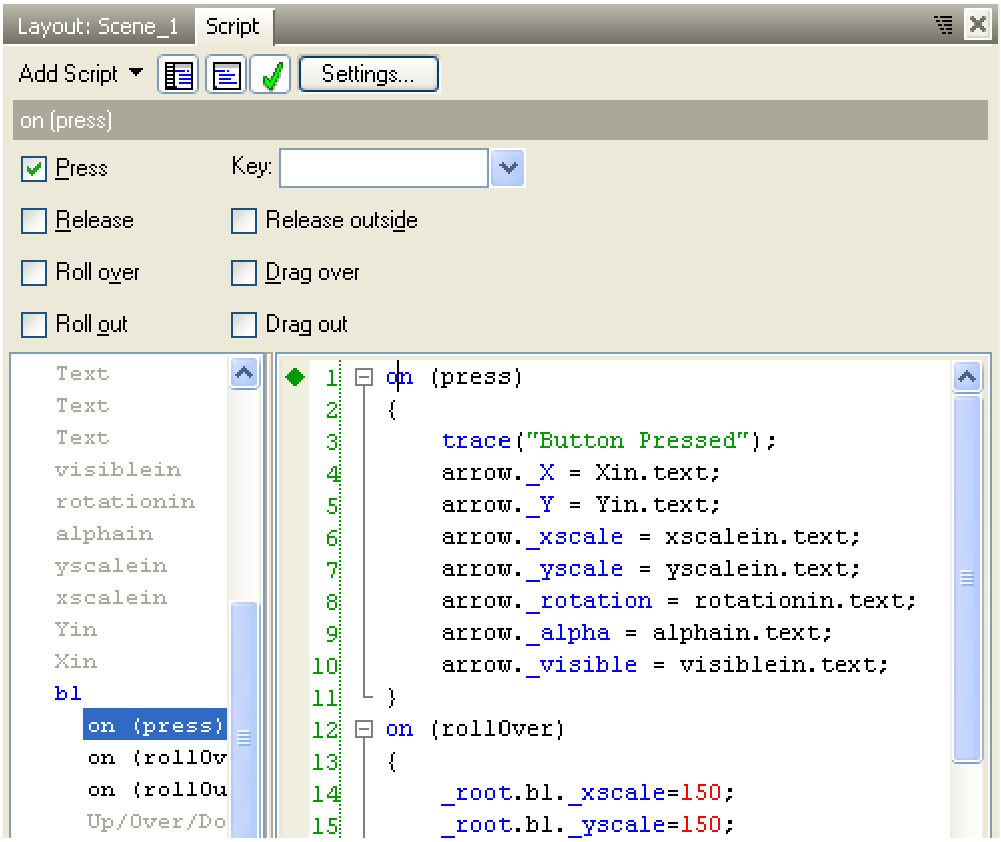
When the function is executed, the arrow properties will be assigned to the text property of the Input and Dynamic Text Objects

10. Add the following statements to the on (press) Event Handling function of the 'b1' button:

```
arrow._X = Xin.text;
arrow._Y = Yin.text;
arrow._xscale = xscalein.text;
arrow._yscale = yscalein.text;
arrow._rotation = rotationin.text;
arrow._alpha = alphain.text;
arrow._visible = visiblein.text;
```

Do this by selecting the 'b1' button in the 'Outline' Panel.

The on (press) function should appear as shown:



☑ Press Key:

□ Release □ Release outside

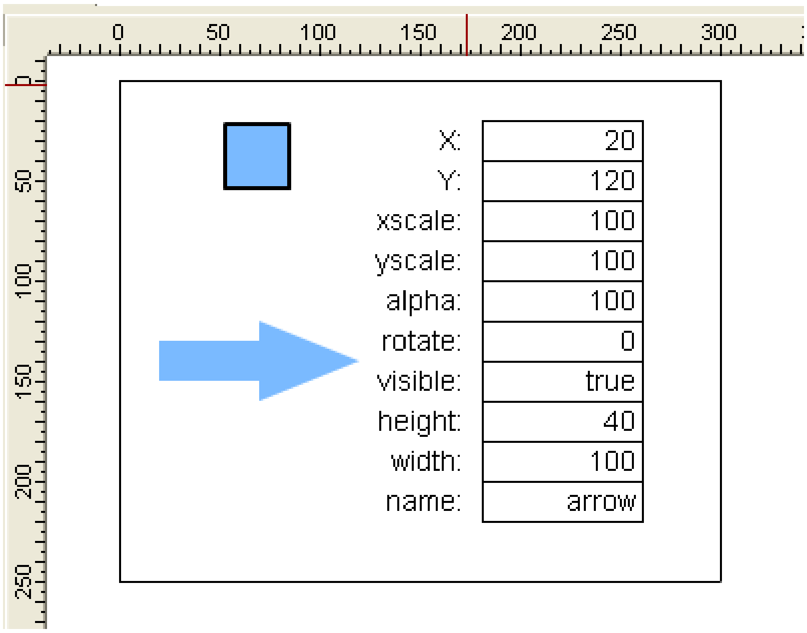
□ Roll over □ Drag over

□ Roll out □ Drag out

```
on (press)
{
    trace("Button Pressed");
    arrow._X = Xin.text;
    arrow._Y = Yin.text;
    arrow._xscale = xscalein.text;
    arrow._yscale = yscalein.text;
    arrow._rotation = rotationin.text;
    arrow._alpha = alphain.text;
    arrow._visible = visiblein.text;
}
on (rollOver)
{
    _root.b1._xscale=150;
    _root.b1._yscale=150;
```

When this function is executed, the arrow properties will be updated with new values obtained from the text property of the various input Text Objects

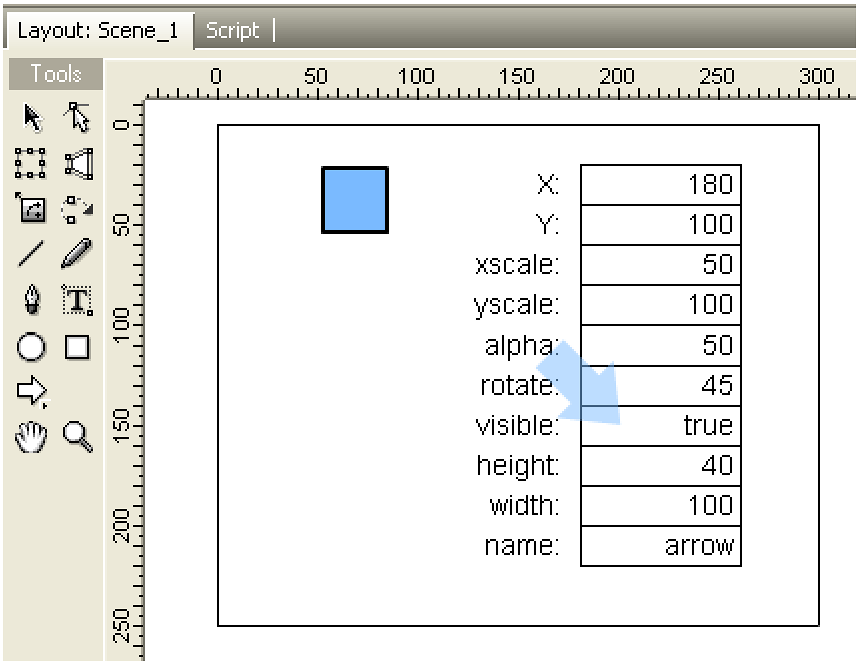
11. Select the 'Layout' tab and press the 'Play' button. The initial properties of the arrow Object should be displayed.



X:	20
Y:	120
xscale:	100
yscale:	100
alpha:	100
rotate:	0
visible:	true
height:	40
width:	100
name:	arrow

Note: The values you achieve for X, Y, height and width may be different depending on where and how large you drew the arrow

It is possible to alter the X, Y, xscale, yscale, rotation and visible settings. If you modify those settings while the movie is playing then press the "=" button, the arrow with the revised settings will be displayed.



X:	180
Y:	100
xscale:	50
yscale:	100
alpha:	50
rotate:	45
visible:	true
height:	40
width:	100
name:	arrow

Note: If you modify the "visible" property to "0" (zero) it will hide the Arrow.

Analysis
This example demonstrates the various properties that are associated with a Scripting Object.

The arrow onSelfEvent (load) Event Handling function is used to load the Input and Dynamic Text Objects with the initial properties of the arrow Object. The arrow properties _X, _Y, _xscale, _yscale, _alpha, _rotation and _visible may be altered by modifying the appropriate Input Text Object and then clicking on the button. The button on (press) Event Handling function is used to copy the revised property settings from the Input Text Objects to the arrow Object.

Note: Altering the _xscale and _yscale properties will cause the _height and _width properties to change. The tutorial example will not display this change as the _height and _width properties are only read when the arrow is initially loaded

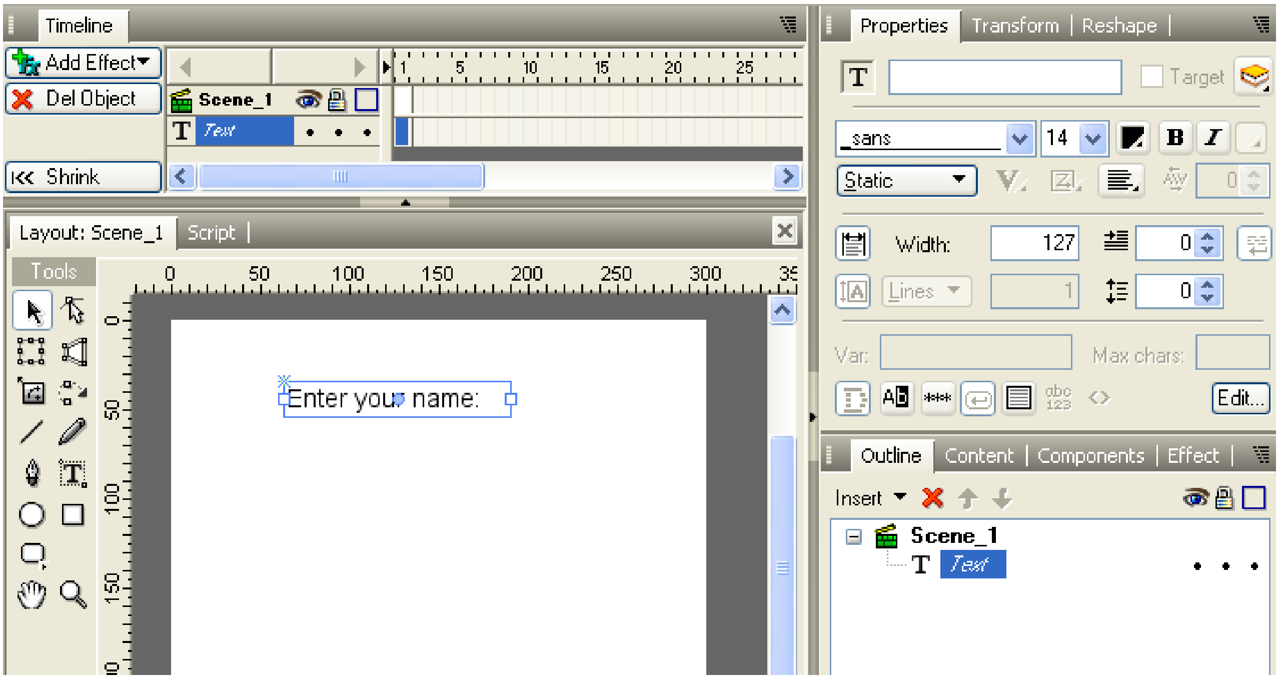
4.6.7.8 Input and Dynamic Text

Description
Demonstrates how users can add text to the movie to be used at a later time in the same movie.

Aim
Introduces methods for user input through Input Text fields and storing that information as a variable to be used in dynamic content.

1. Create a new file and save the file as "mywelcome.swi"

2. Insert a Static text object and type in the text "Enter your name:".
Note: As multiple fonts and text sizes are supported in the same static text object, it is necessary to select the text you want to change before altering the font or text size.

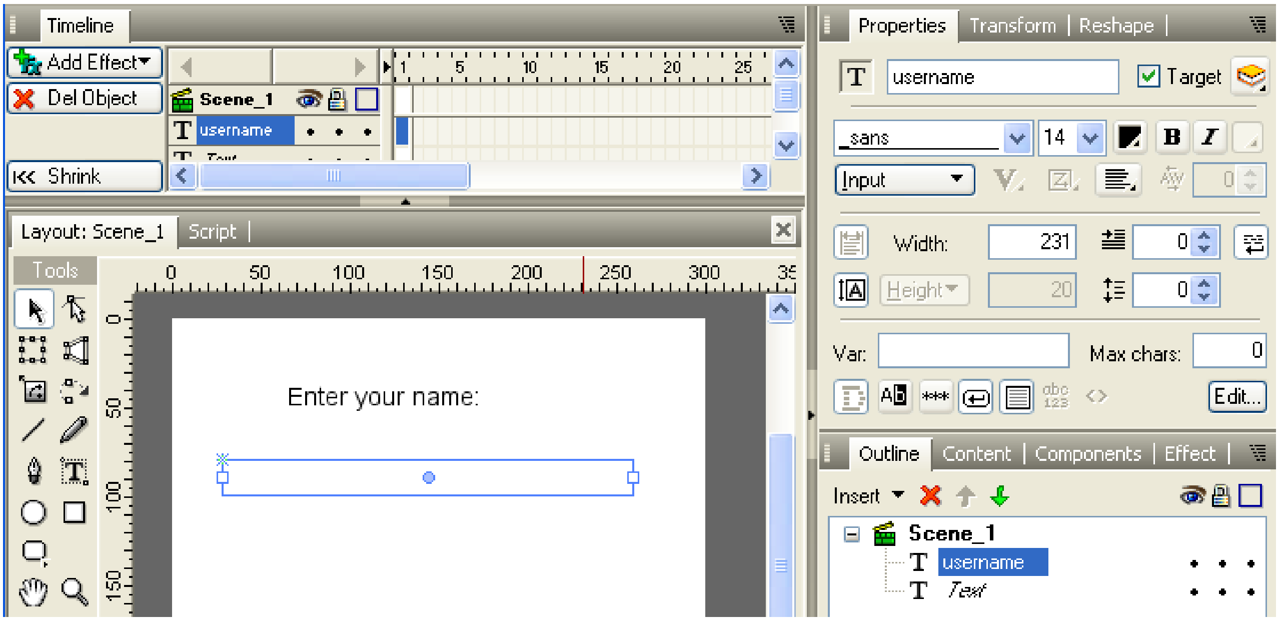


Enter your name:

_sans 14 Static

Width: 127

3. Insert an Input Text object. Name the text field "username" and tick the 'Target' option on the text panel. Press the button for a black border and white background ▤



username

_sans 14

Input

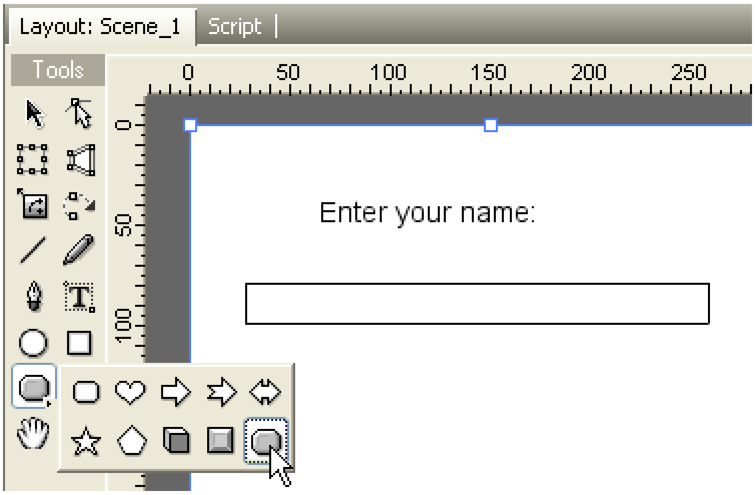
Width: 231

Height 20

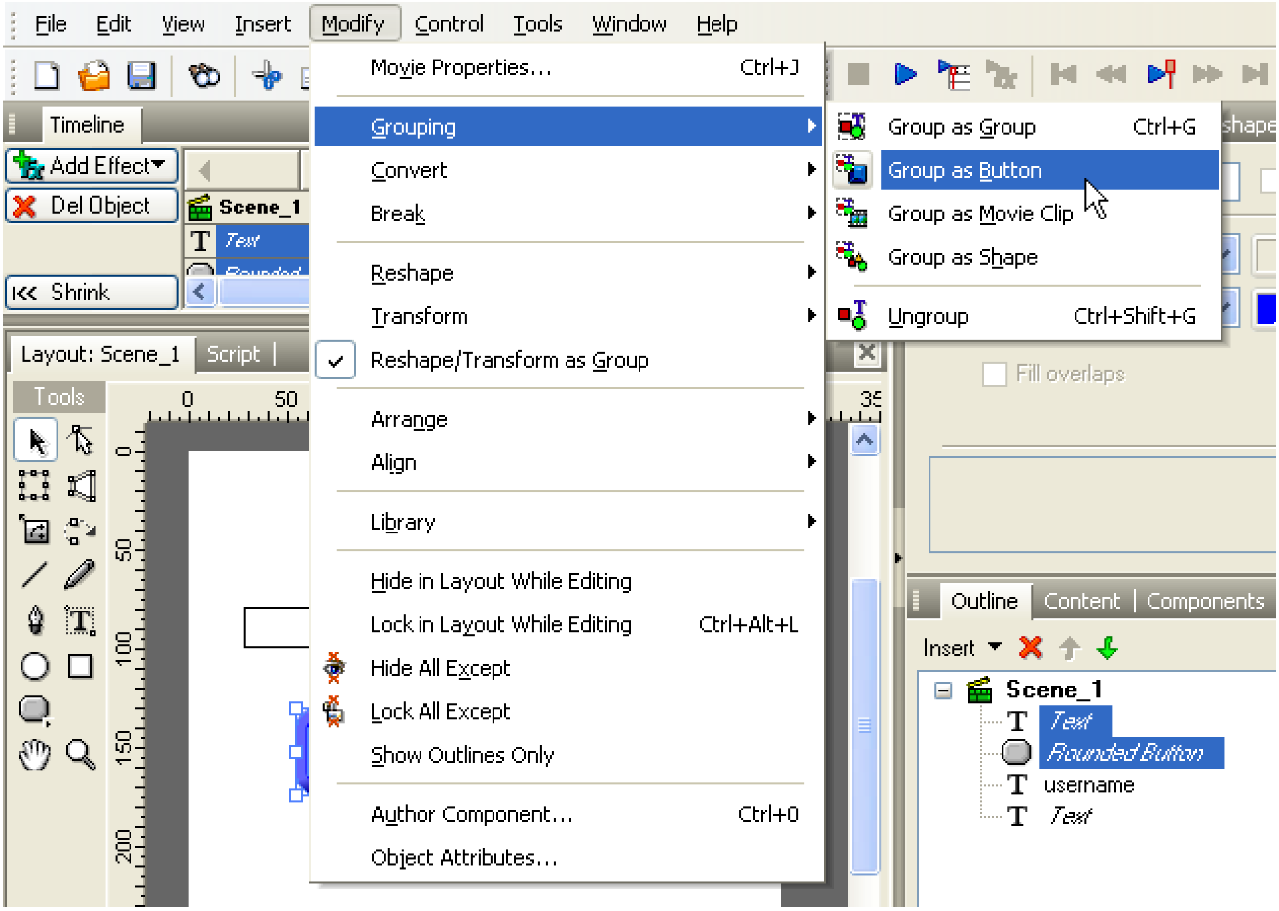
Var:

Max chars: 0

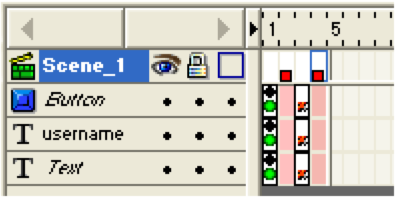
4. Use the AutoShape tool to draw a 'Rounded Button' below the Input text field. Change its color to Blue on the Shape panel. Use the green handles to adjust the gradient position and bevel size. Set the border line to None (this generally looks better than thin border lines with curved sections).



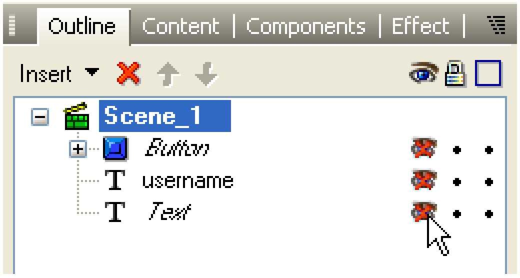
5. Insert a Static text object with the text "Continue" and place it over top of the Rounded Button. Change the text to Bold and White Hold down the CTRL key and click on both the Static text object and the Rounded Button shape to highlight them. Use the Modify menu and select **Grouping | Group as Button**.



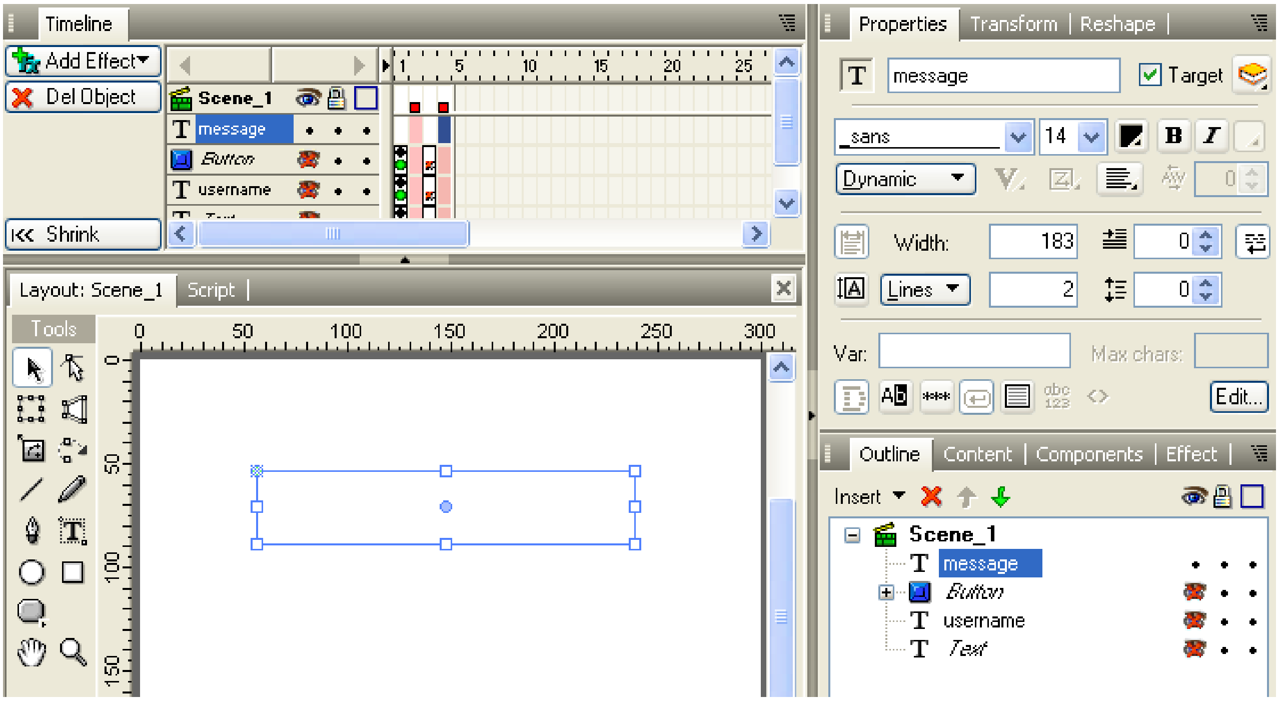
6. In the 'Timeline' panel, at Frame 1 add a Place effect for all the objects. Do this by right clicking on the timeline in frame one for the associated object and then select Place.
Add a 'Stop' - **Movie Control | stop()** - action on Frame 2 of the Scene_1 row in the 'Timeline'. At Frame 3 for each of the objects add a Remove effect then add another 'Stop' action at Frame 4 of the Scene_1 row.
The Timeline should be shown as below:



7. Click on the Eye icon next to each object in the 'Outline' panel to hide them from the stage.



8. Insert a Dynamic text object and name it "message" on the text panel - remember to tick the 'Target' option as well. Open the 'Dimensions' text options and untick the Auto-Height option. Set it to use 'Lines' rather than 'Height' and enter '2' as the number of lines it should use.



message ☑ Target

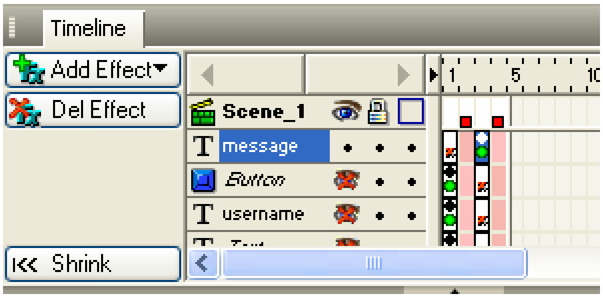
_sans 14

Dynamic

Width: 183 0

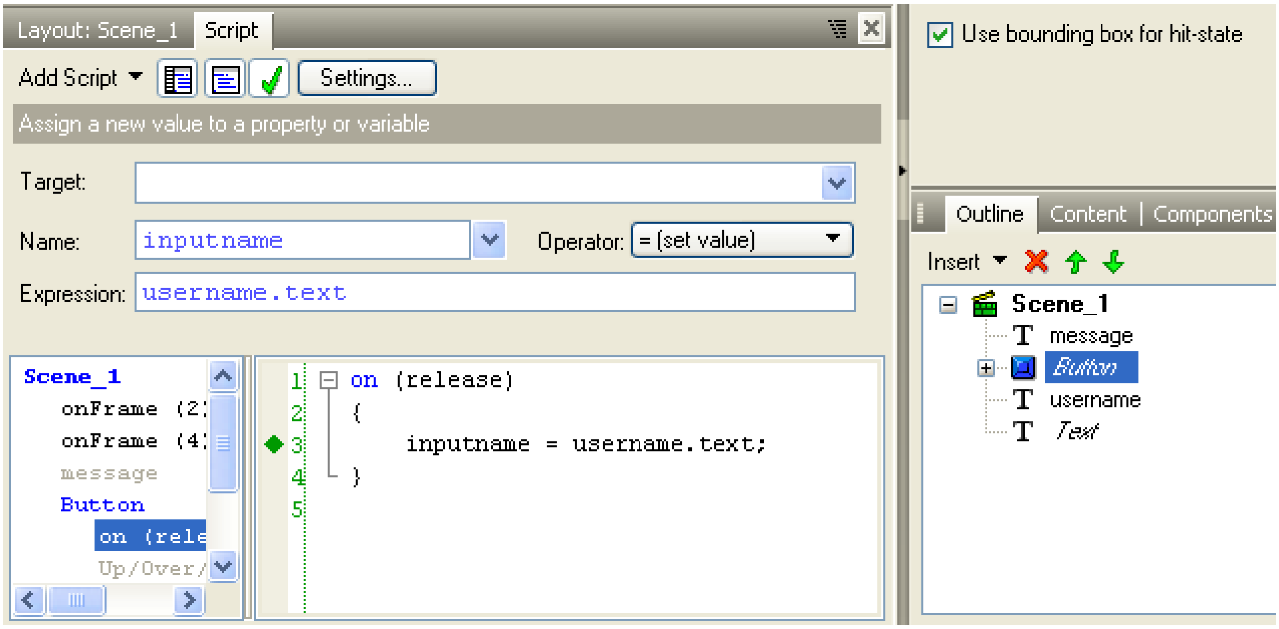
Lines 2 0

9. In the 'Timeline' panel on the "message" row, insert a Remove effect at Frame 1 and a Place effect at Frame 3. The 'Timeline' panel should be shown as below:



10. Select the Button object in the 'Outline' panel then open the 'Script' panel. Press the 'Add Script' button and select **Events | Button | on(release)** from the menu.

11. Press the 'Add Script' button again and select **Statements | name = expr;** from the menu. Leave the Target field empty and enter "inputname" into the Name field. Leave the Operator set at "=" then enter "username.text" in the value field. Here we are creating a new Variable named "inputname" that will store the text typed into the Input text field by the user. The 'Script' panel for the Button object should be shown as below:



☑ Use bounding box for hit-state

Assign a new value to a property or variable

Target:

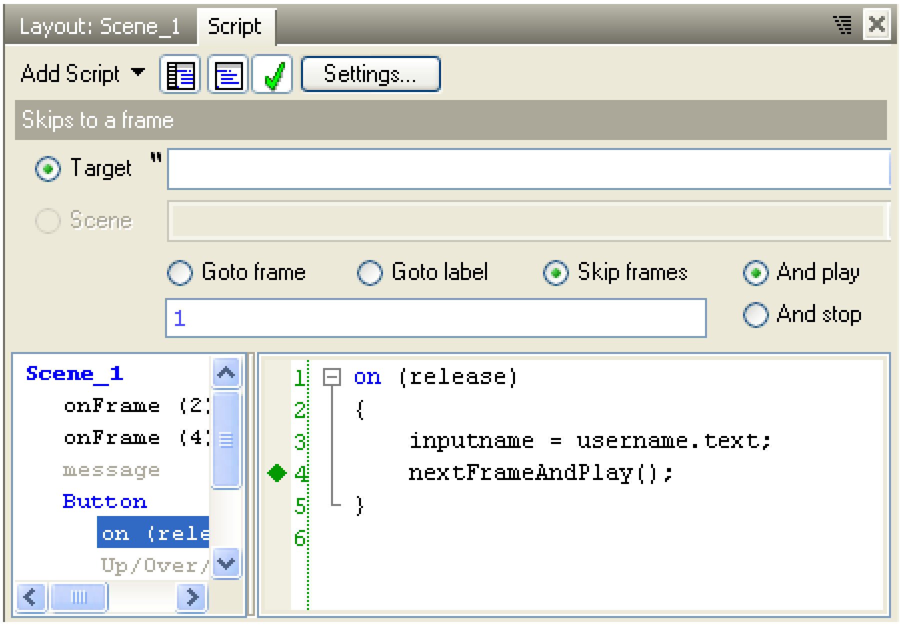
Name: inputname

Operator: = (set value)

Expression: username.text

```
on (release)
{
    inputname = username.text;
}
```

12. Press the 'Add Script' button again and select **Movie Control | gotoAndPlay | nextFrameAndPlay()** from the menu.



Skips to a frame

● Target

○ Scene

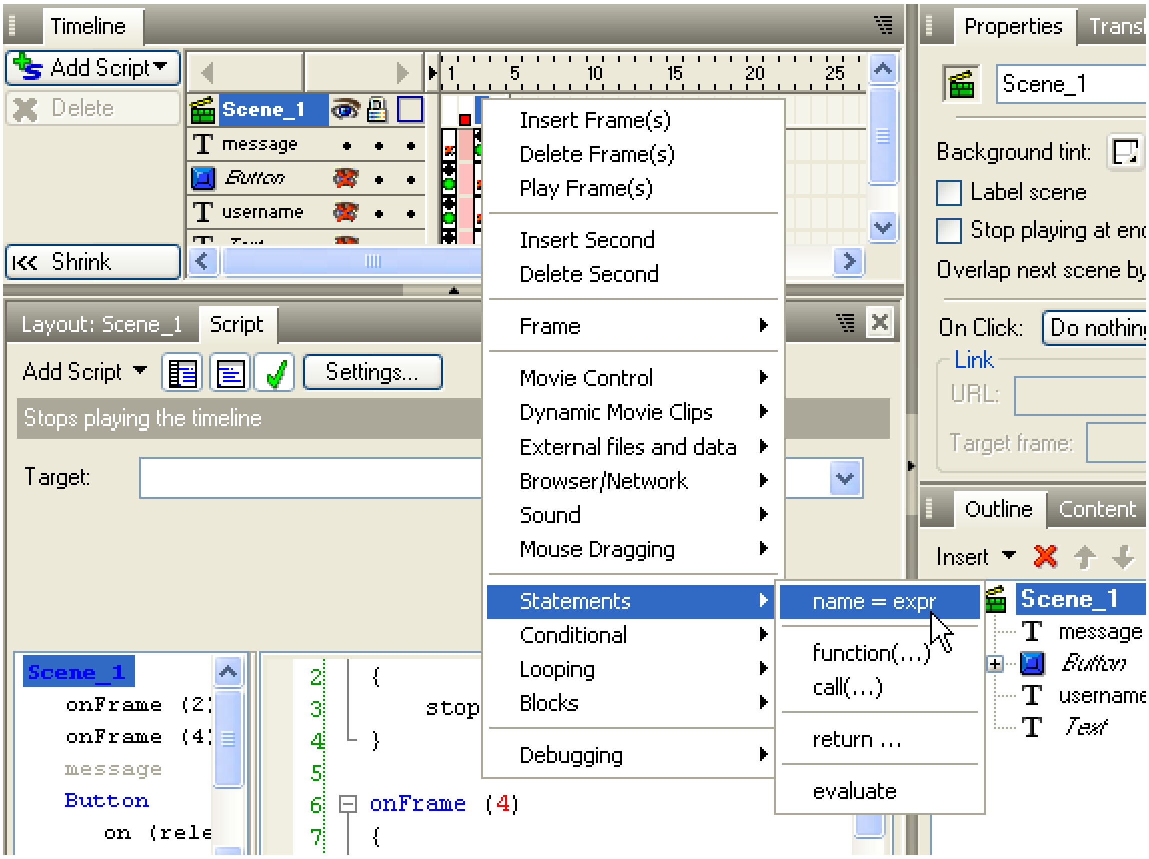
○ Goto frame ○ Goto label ● Skip frames ● And play

1 ○ And stop

```
on (release)
{
    inputname = username.text;
    nextFrameAndPlay();
}
```

Note: It is important that the variable "inputname" be defined before the gotoAndPlay() action is executed.

13. In the 'Timeline' panel, Right-Click at Frame 3 on the row for Scene_1 and select **Statements | name = expr;** from the list.

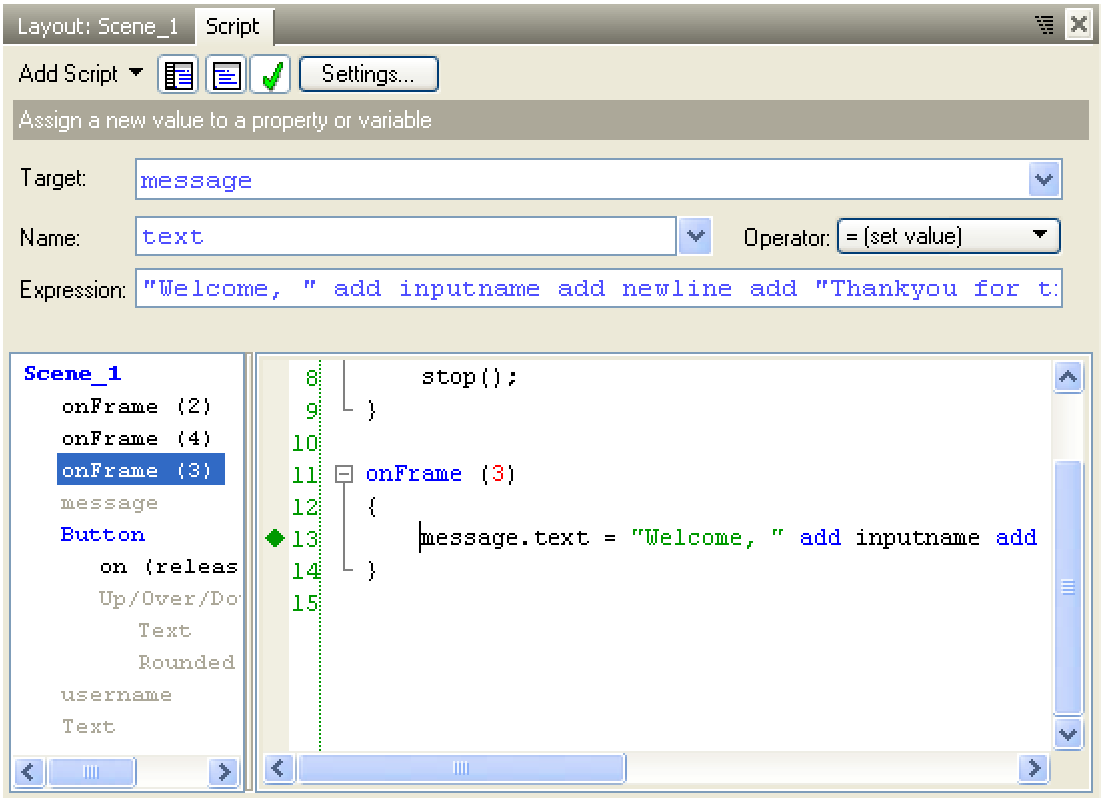


```
Scene_1
    onFrame (2)
    onFrame (4)
    message
    Button
        on (rele
```

```
{
    stop
}
onFrame (4)
{
```

14. Use the drop-down menu next to the Target field and select the Dynamic text object 'message' from the list. In the Name field use the drop-down menu and select **Text | text.** Leave the Operator set to "=" and enter the text below into the Expression field.

"Welcome, " add inputname add newline add "Thankyou for trying SWiSH Max!"



Target: message

Name: text

Operator: = (set value)

Expression: "Welcome, " add inputname add newline add "Thankyou for t:

```
        stop();
    }

    onFrame (3)
    {
        message.text = "Welcome, " add inputname add
    }

```

Note: The text displayed inside quotes is the static message that will be displayed. Using the "add" is the same as adding one string at the end of another (which is why you enter spaces manually inside quotes). We have added the variable "inputname" which will be entered by the user. The "newline" command will enter a carriage return into the text field (starting a new line).

15. Go back to the 'Layout' panel and press the 'Play' button to preview the movie. Type your name into the input text field then press the Continue button. The welcome message should be displayed with the name you entered.

Note: If you only see one line of text displayed in the Message, you will either need to increase the width of the Dynamic text object or make sure that you have the Auto-Height option turned off and have it set to use 2 Lines of text.

16. Press the 'Stop' button and save your movie.

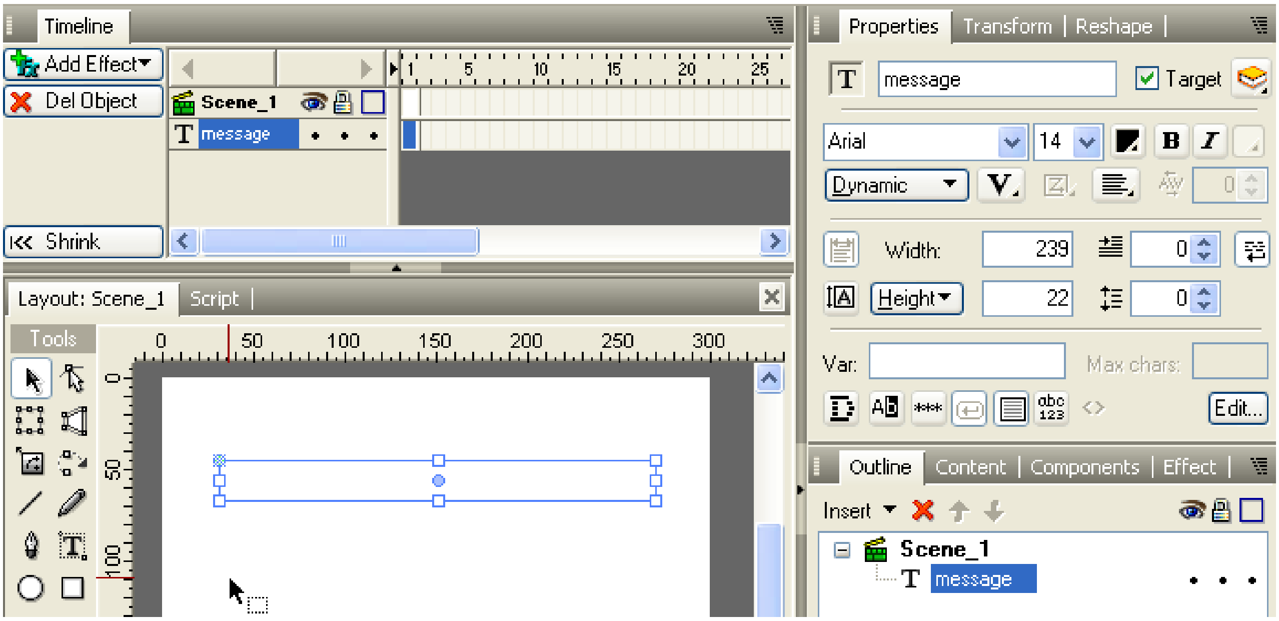
Analysis
When the 'Play' button is pressed, the movie begins with the Input text field in place. After the 'Continue' button is pressed, the variable "inputname" is created and contains whatever was typed into the Input text field. This variable is then displayed in the Dynamic text field "message".

4.6.7.9 Arrays and Random

Description
Demonstrates how to use Arrays to display a random message.

Aim
Introduces methods for Arrays and Math.randomInt()

1. Create a new file and save it as "mymaxarray.swi". Insert a Dynamic text object and name it "message" on the 'Text' panel. Be sure to tick the 'Target' option so that it can be used in our script.



Properties: T message ☑ Target

Arial 14 B I

Dynamic

Width: 239 0

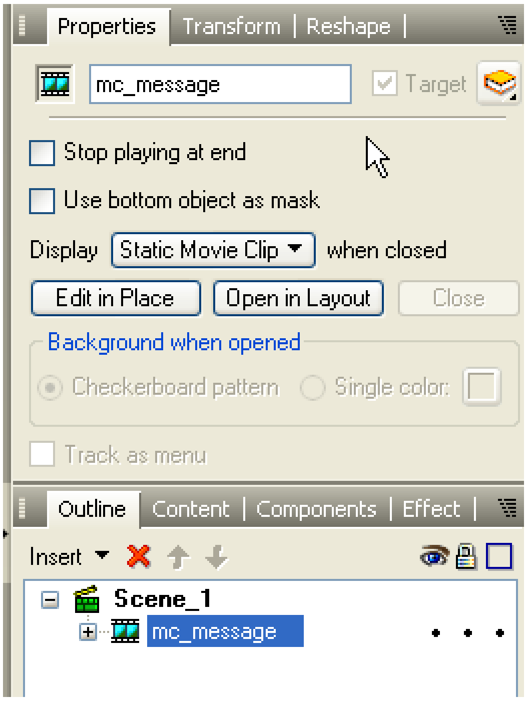
Height 22 0

Var:

Max chars:

Outline: Scene_1 › T message

2. Right-Click on the text object in the 'Outline' panel and select **Grouping | Group as Movie Clip** from the menu. Name the Movie Clip "mc_message" on the 'Properties' panel.



Properties: mc_message ☑ Target

□ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip when closed

Background when opened: ● Checkerboard pattern ○ Single color:

□ Track as menu

Outline: Scene_1 › mc_message

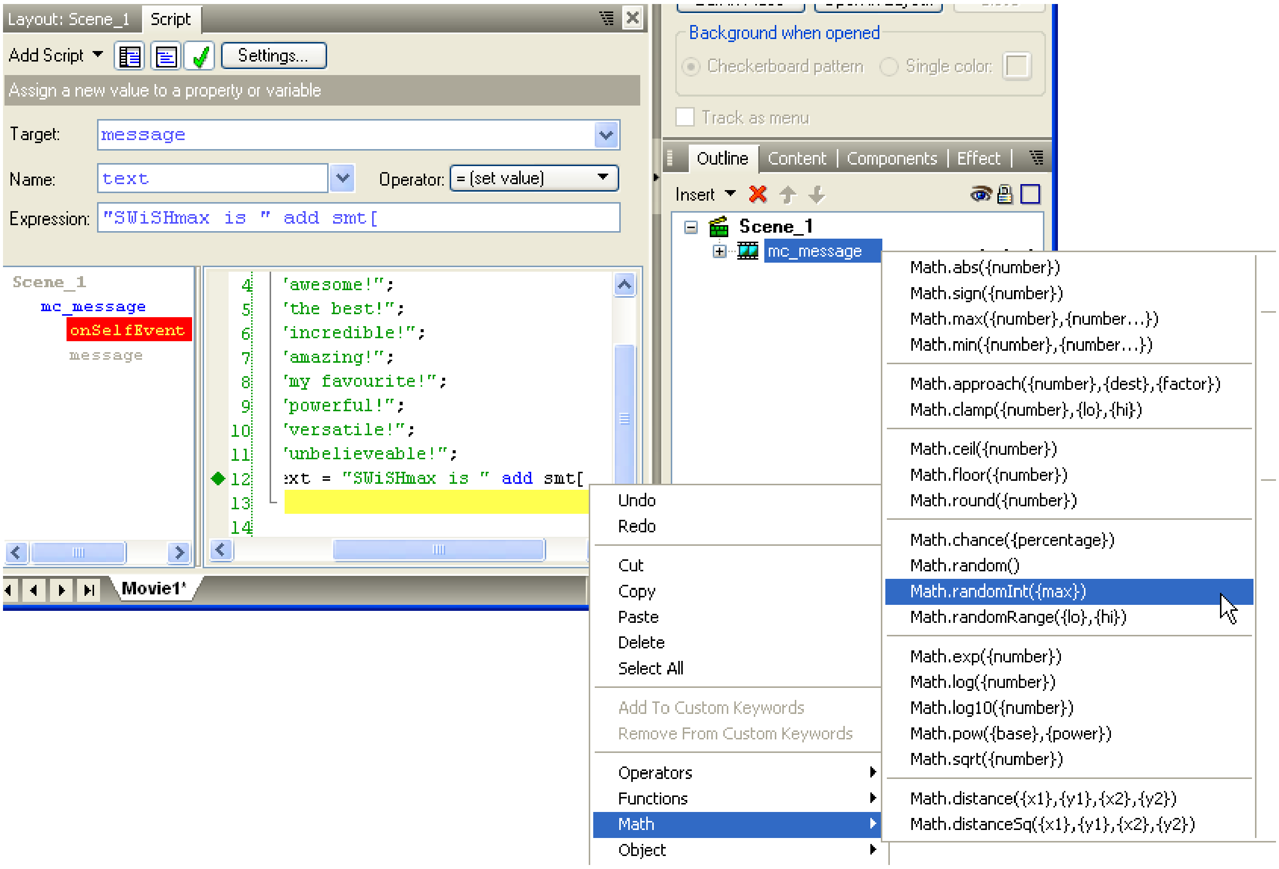
3. With the Movie Clip object selected in the 'Outline' panel, open the 'Script' panel and press the 'Add Script' button. Select **Events | Frame | onSelfEvent (load)** from the menu.

4. Enter the following script directly into the **onSelfEvent (load)** function via the Script Editor. You can also use cut and paste to save typing.

```
smt = new Array();
smt[0] = "awesome!";
smt[1] = "the best!";
smt[2] = "incredible!";
smt[3] = "amazing!";
smt[4] = "my favourite!";
smt[5] = "powerful!";
smt[6] = "versatile!";
smt[7] = "unbelieveable!";
```

Note: In the script above, a new empty array object was created and assigned to the variable "smt".
Note: Variables within "smt" are accessed using square brackets. The number inside the square brackets indicates the position (or index) within the array (please note, array indices always start from zero as the first element of the array). The message "awesome!" is defined for this position in the array.

5. When you have created several new values for the array, add the line
message.text = "SWiSH Max is " add smt[
then click the right mouse button and select **Math | Math.randomInt({max})** from the menu as shown below:



Assign a new value to a property or variable

Target: message

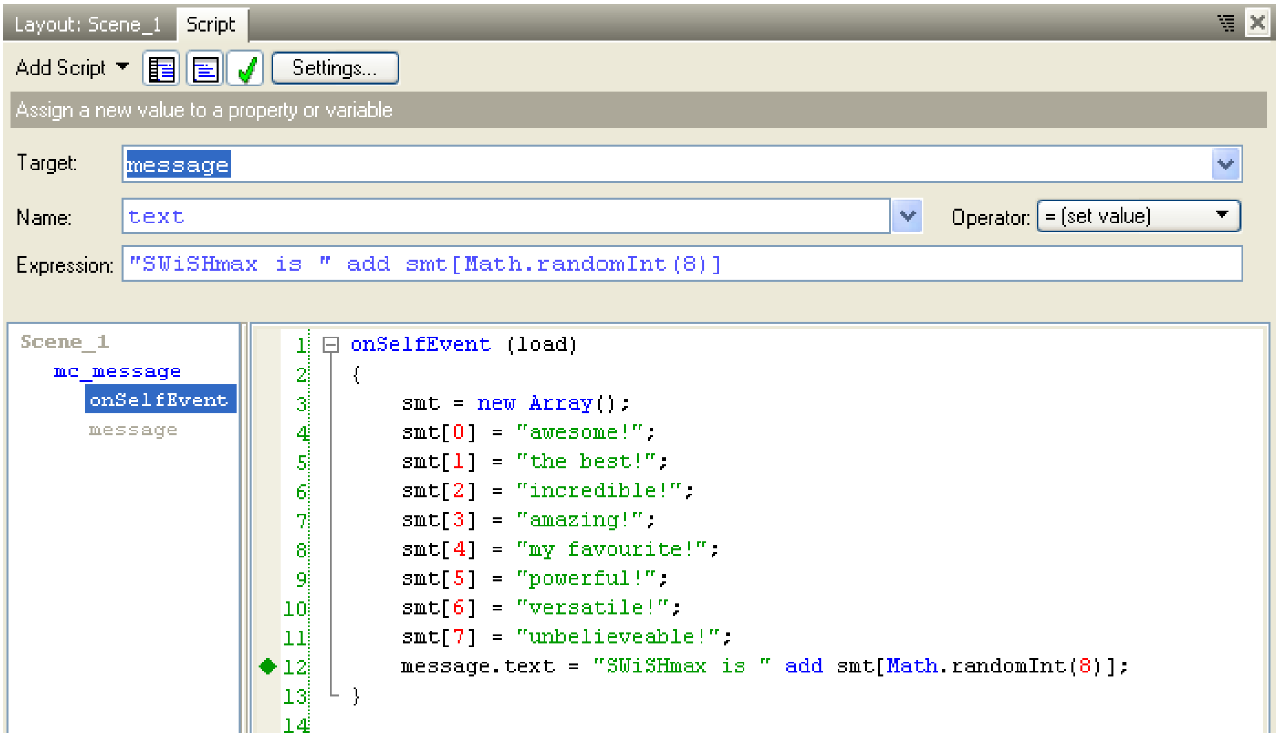
Name: text

Operator: = (set value)

Expression: "SWiSHmax is " add smt[

```
'awesome!";
'the best!";
'incredible!";
'amazing!";
'my favourite!";
'powerful!";
'versatile!";
'unbelieveable!";
ext = "SWiSHmax is " add smt[
```

6. When the Math.randomInt() script is added, the cursor will be inside the parentheses. Type the number "8" (which is one number higher than the last number in the array) into the parentheses. Then finish the script by adding the closing bracket "]". The 'Script' panel should appear similar to the image below:



Target: message

Name: text

Operator: = (set value)

Expression: "SWiSHmax is " add smt[Math.randomInt(8)]

```
onSelfEvent (load)
{
    smt = new Array();
    smt[0] = "awesome!";
    smt[1] = "the best!";
    smt[2] = "incredible!";
    smt[3] = "amazing!";
    smt[4] = "my favourite!";
    smt[5] = "powerful!";
    smt[6] = "versatile!";
    smt[7] = "unbelieveable!";
    message.text = "SWiSHmax is " add smt[Math.randomInt(8)];
}
```

8. Go back to the 'Layout' panel and press the 'Play' button in the Control Toolbar to preview the movie. If you get an error :

Cannot use constructor initialiser when exporting to SWF4 in statement: ...

Then either change the Export settings to SWF5 or later or remove the line:
smt = new Array();
which is not needed for SWF4.

Press the 'Play' button several times to see a random message displayed.

Analysis:
The Movie Clip "mc_message" contains on onSelfEvent (load) event which is triggered when the Movie Clip is first loaded. Inside the onSelfEvent (load) event a new Array object was defined. Array indices begin with 0 - zero - as the first position in the array and a value is defined for each position. Using the Math.randomInt() code, the movie will select a number between 0 and the number inserted into parentheses (note, the number in parentheses will not be one of the random numbers which is why we used a value one greater than the last position in the array).

The script that contains "smt[Math.randomInt(8)]" will create a random number for the array and the number that is picked will correspond with one of the positions in the array.

4.6.7.10 Mouse Dragging

Description
This is a step-by-step tutorial demonstrating how to make draggable Objects.

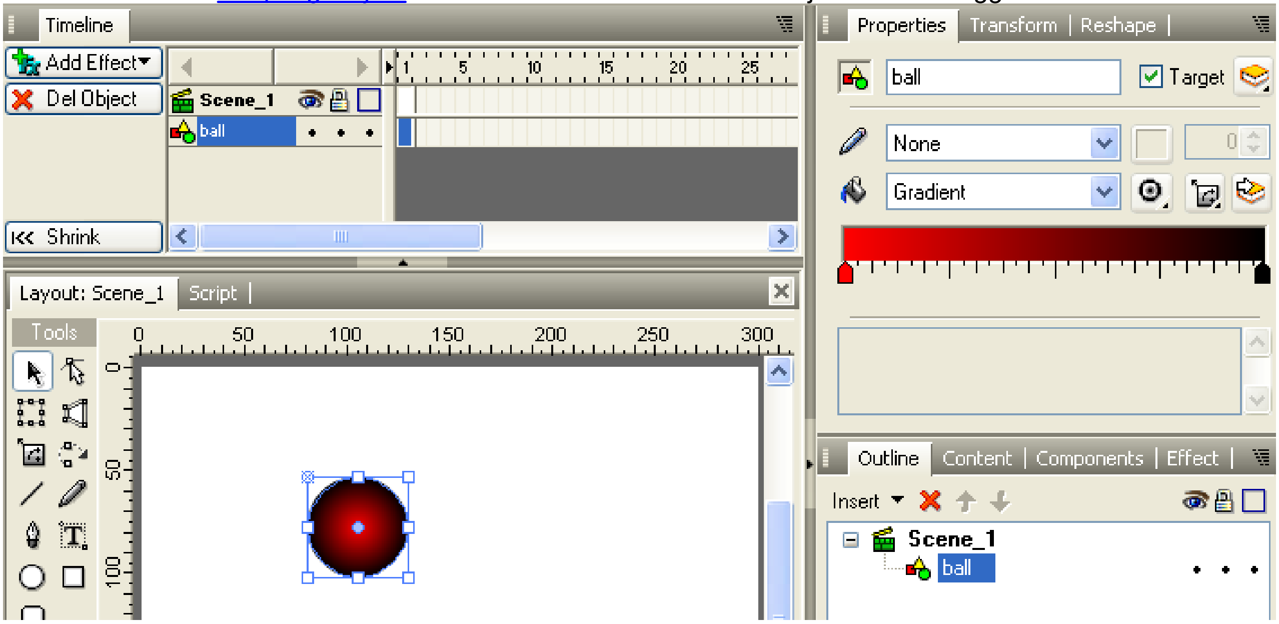
Aim
This tutorial introduces startDrag() and stopDrag() methods and onSelfEvent(press) and onSelfEvent(release) Events to start and stop dragging.

.swi file

"dragging.swi"

1. Start with a new project (File | New) and save the file as "mydragging.swi"

2. Draw a small circle and name it "ball" on the 'Shape' panel then tick the 'Target' checkbox so that it can be used as a Scripting Object. This circle will be used as the Object that is dragged

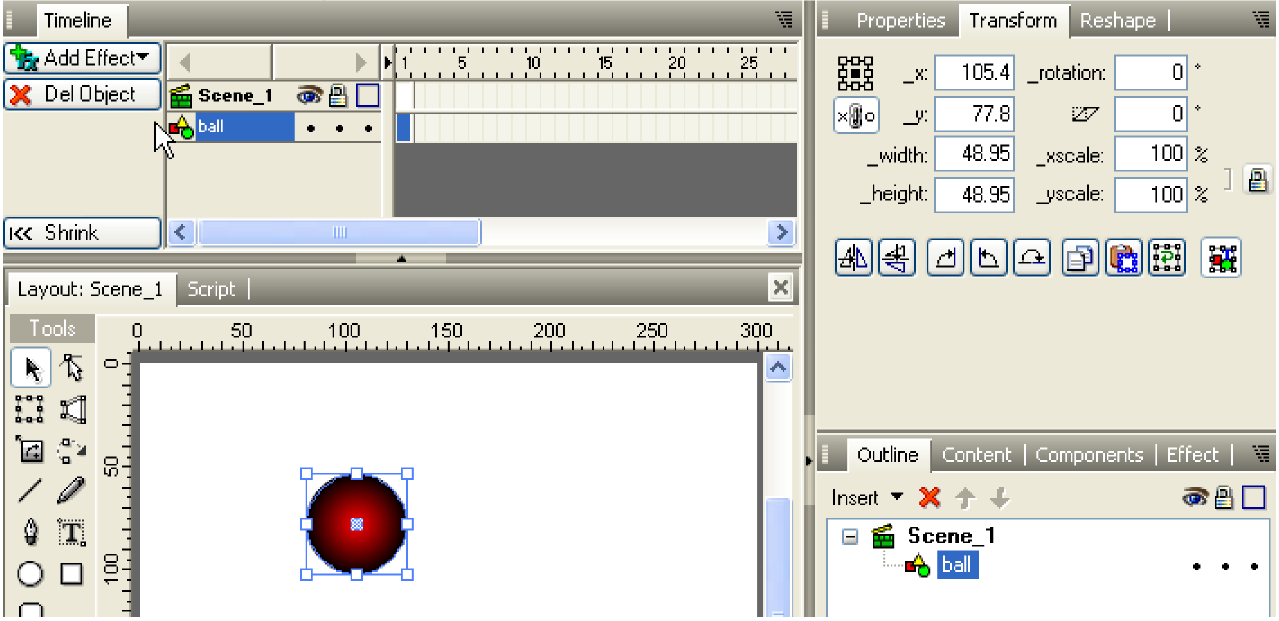


ball ☑ Target

None

Gradient

2a. Select the Transform Tab and press the button to lock the transformation point to the reference point. Then select the center point.



_x: 105.4 _rotation: 0 °

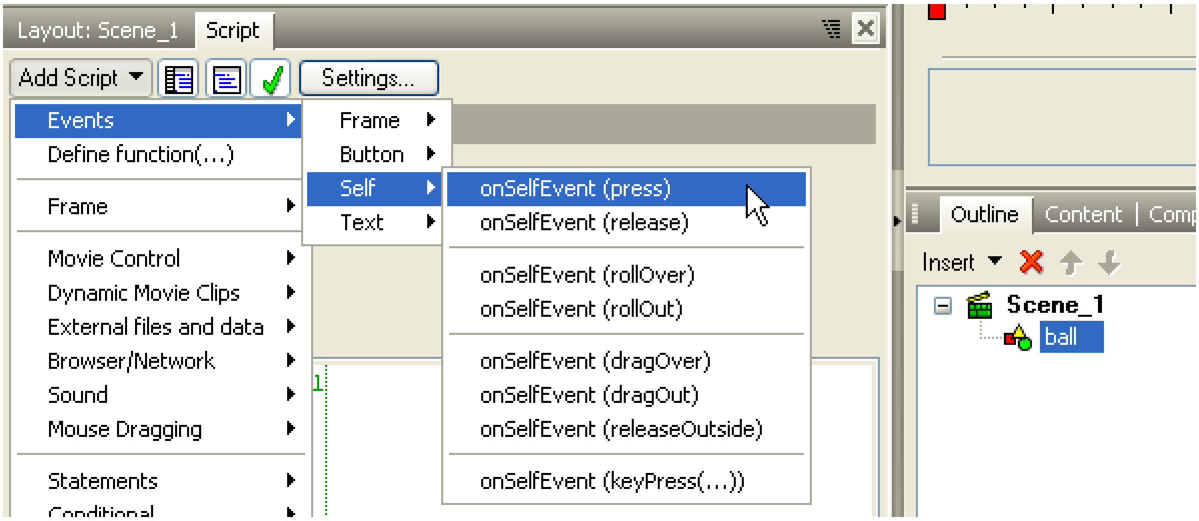
_y: 77.8 0 °

_width: 48.95 _xscale: 100 %

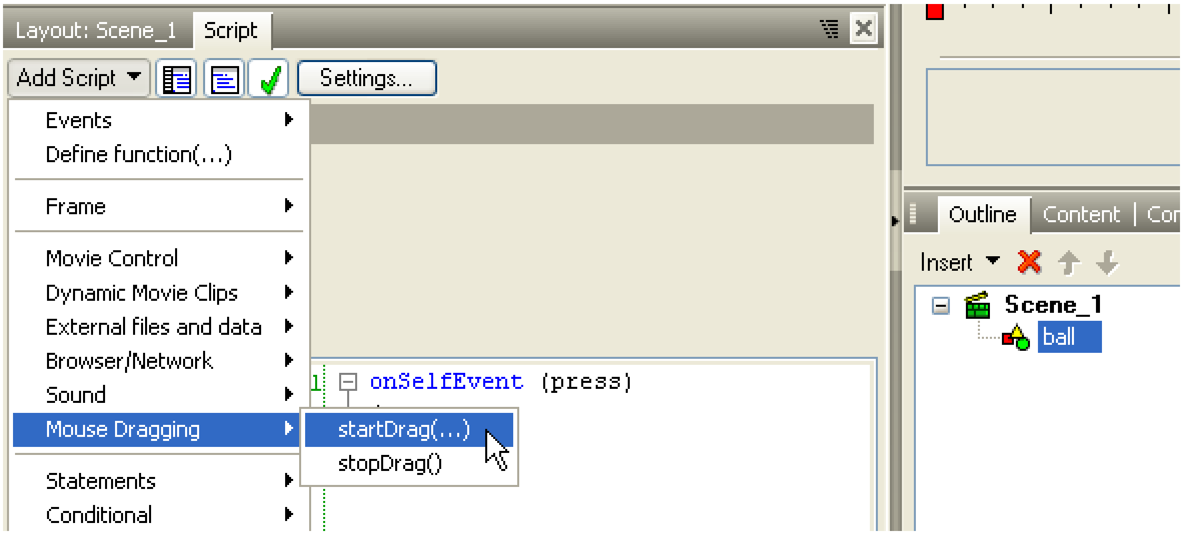
_height: 48.95 _yscale: 100 %

You should now see a x in the middle of the ball.

3. Select the Script tab and add an onSelfEvent(press) event for the ball.

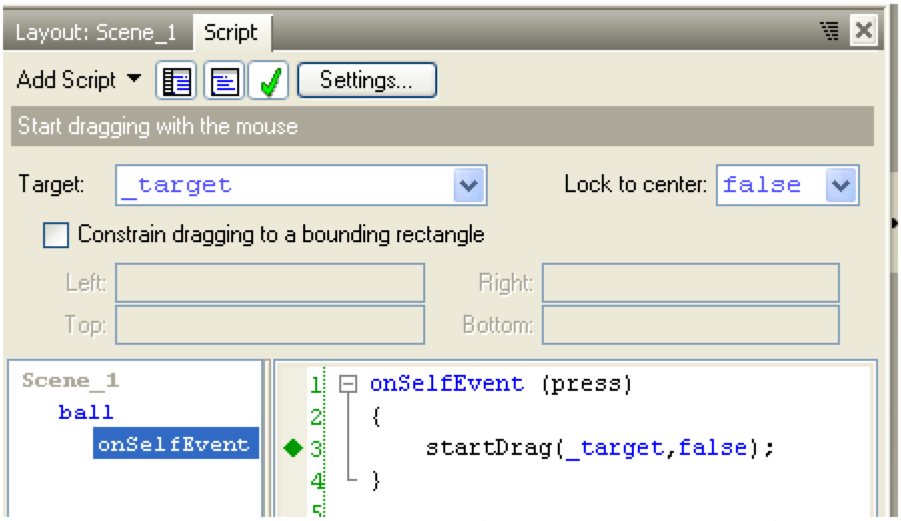


4. Right-click on the function name to add a Mouse Dragging | startDrag() command to the onSelfEvent(press) function.



```
onSelfEvent (press)
```

The resulting script should be as shown below:



Start dragging with the mouse

Target: _target

Lock to center: false

□ Constrain dragging to a bounding rectangle

Left:

Right:

Top:

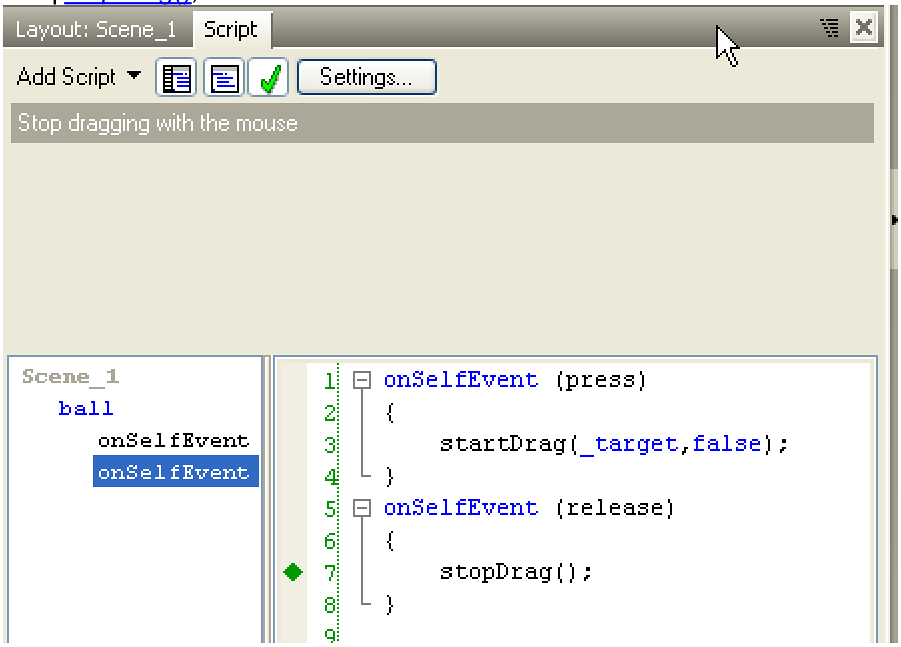
Bottom:

Scene_1
ball
onSelfEvent

```
onSelfEvent (press)
{
    startDrag(_target,false);
}
```

Leave the 'Target:' parameter blank (or with _target displayed). This will result in the current Object (ball) being the item intended to be dragged

5. In a similar way, create an onSelfEvent(release) function for the ball with the command Mouse Dragging | stopDrag();



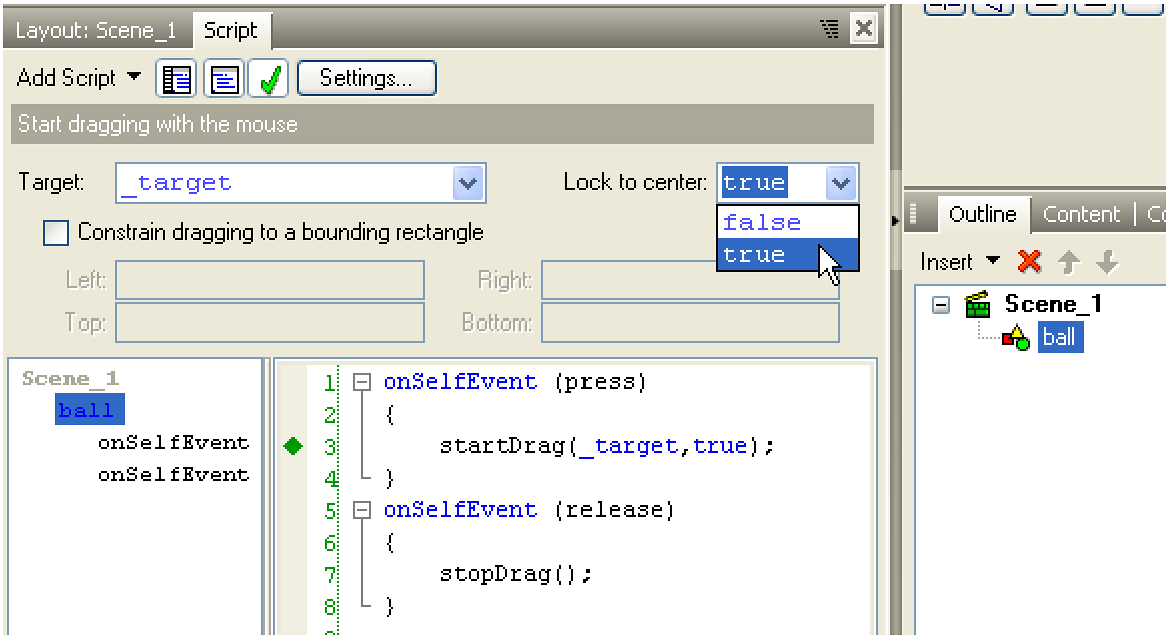
Stop dragging with the mouse

Scene_1
ball
onSelfEvent
onSelfEvent

```
onSelfEvent (press)
{
    startDrag(_target,false);
}
onSelfEvent (release)
{
    stopDrag();
}
```

6. Select the 'Layout' tab and press the 'Play' button. Move the mouse over the ball Object and press the left mouse button. While the button is held down, you should be able to drag the ball around the 'Layout' Panel. When the button is pressed, the mouse position within the Object stays in the location where the mouse was first pressed

7. Select the 'Script' tab and then select the startDrag() command. Select true in the 'Lock to center' combo box. This will change the command to startDrag(_target, true). Return to the 'Layout' Panel and observe the new behaviour (press play if it is not already running)



Start dragging with the mouse

Target: _target

Lock to center: true

false
true

□ Constrain dragging to a bounding rectangle

Left:
Right:
Top:
Bottom:

Scene_1
ball
onSelfEvent
onSelfEvent

```
onSelfEvent (press)
{
    startDrag(_target,true);
}
onSelfEvent (release)
{
    stopDrag();
}
```

Note: When the left mouse button is pressed over the Object, the mouse position within the Object now snaps to the Reference Point of the "ball" object.

Analysis
This tutorial demonstrates how to make an Object draggable according to mouse input. The onSelfEvent(press) and onSelfEvent(release) Events are used to start and stop dragging of the ball Object. The methods startDrag() and stopDrag() are used to start and stop the dragging. The Lock to center option can be used to force dragging from the Reference Point of the object.

4.6.7.11 Collision Detection

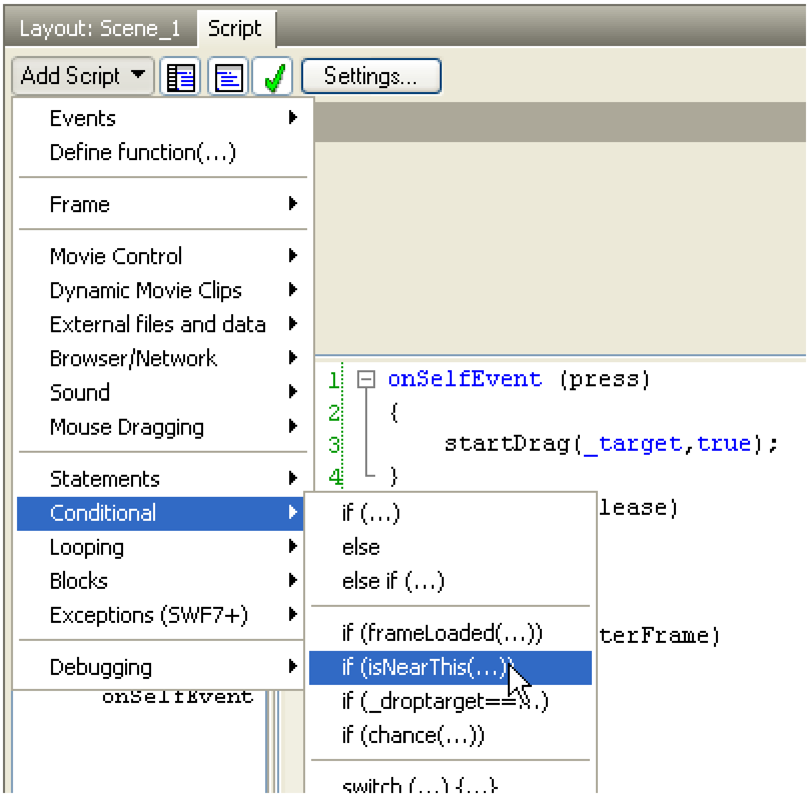
Description
This step-by-step tutorial builds on previous tutorials to demonstrate various methods of collision detection. Two methods isNearTarget() and isNearThis() provide a means of detecting proximity between two Objects.

Aim
This tutorial introduces isNearTarget() and isNearThis() methods.

.swi file
"collide.swi"

1. Continue from the previous tutorial or open the file "droptarget.swi". Save the file as "mycollide.swi"

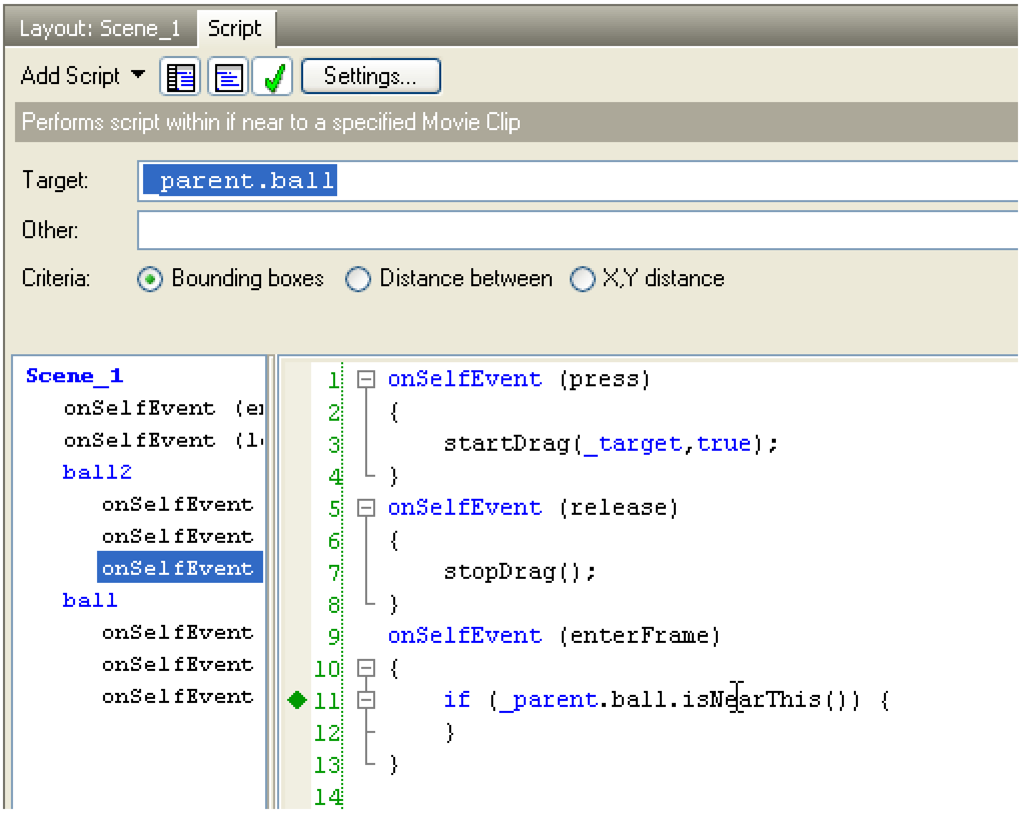
2. Select the blue ball (ball2) in the 'Outline' Panel. Delete the existing Actions from the onSelfEvent (enterFrame) Event then select the if (isNearThis()) conditional statement.



```
onSelfEvent (press)
{
    startDrag(_target,true);
}
```

```
lease)
terFrame)
onSelfEvent
```

Select _parent.ball from the pull-down list (_parent.ball is the other ball) and select the 'Bounding boxes hit' option



Performs script within if near to a specified Movie Clip

Target: parent.ball

Other:

Criteria: ● Bounding boxes ○ Distance between ○ X,Y distance

```
Scene_1
    onSelfEvent (e
    onSelfEvent (l
    ball2
        onSelfEvent
        onSelfEvent
        onSelfEvent
    ball
        onSelfEvent
        onSelfEvent
        onSelfEvent
```

```
onSelfEvent (press)
{
    startDrag(_target,true);
}
onSelfEvent (release)
{
    stopDrag();
}
onSelfEvent (enterFrame)
{
    if (_parent.ball.isNearThis()) {
    }
}
```

Note: As ball and ball2 are both Objects contained within the main Movie, ball can be referenced via either "_root.ball" or "_parent.ball" from inside the ball2 Object

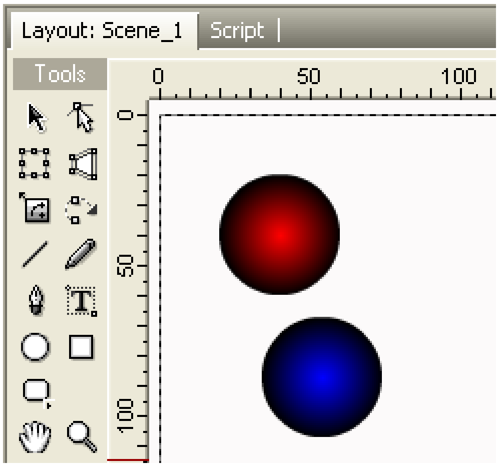
3. Complete the script for the onSelfEvent (enterFrame) event to be:

```
onSelfEvent (enterFrame)
{
 if (_parent.ball.isNearThis()) {
   _alpha = 50;
 } else {
   _alpha = 100;
 }
}
```

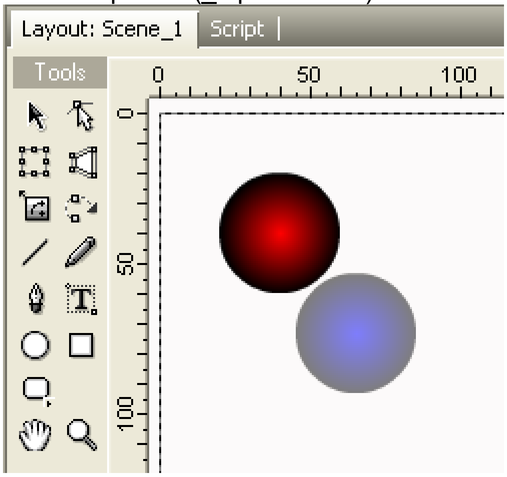
This script will change the _alpha (transparency) of the ball2 Object to 50% whenever it is near the other ball

4. Return to the 'Layout' Panel and press the 'Play' button. Drag the blue ball near the red ball from different directions noting when the blue ball becomes partially transparent.

In this example, the balls are not considered 'Near'.



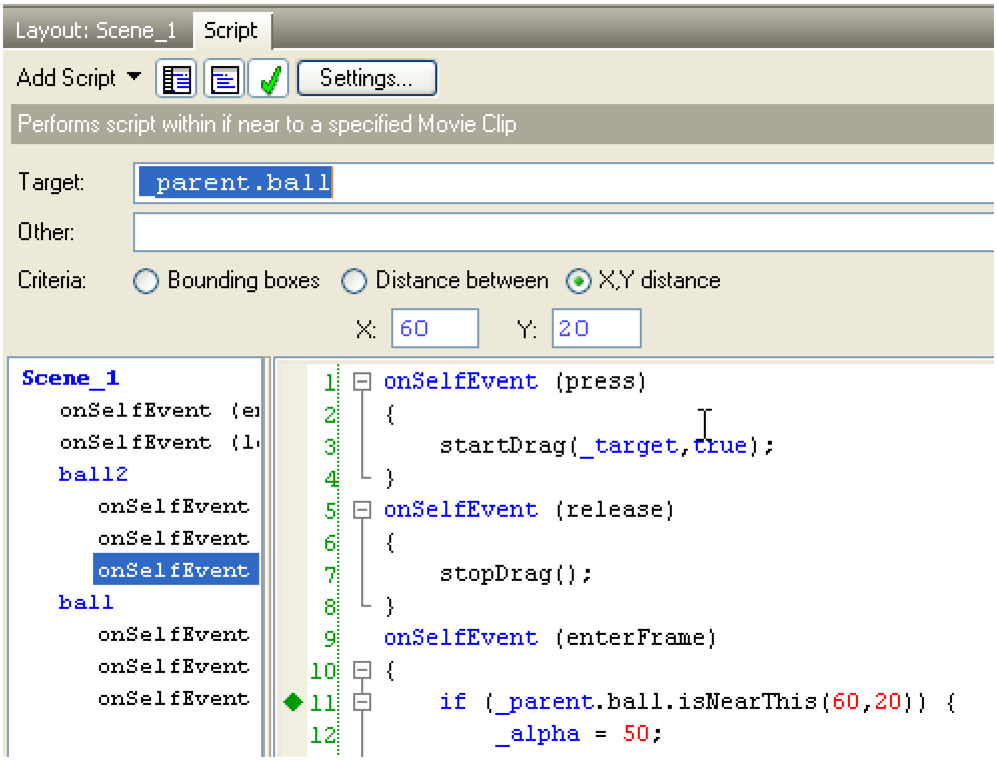
The balls are 'Near' in the example below. This is shown by the blue ball (ball2) becoming partially transparent (_alpha = 50%).



Note:

- The balls are 'Near' even though they are not over lapping. In this case, the definition of 'Near' is defined by the 'Bounding boxes hit' checkbox option. This means that the Objects are considered 'Near' if imaginary rectangles that enclose the Objects intersect
- Note that dragging the red ball towards the blue ball has the same effect (this is different to the previous example)

5. Press the 'Stop' button and return to the 'Script' Panel. Select the "X, Y" option for the if (isNearThis()) statement and enter values 60 for X and 20 for Y.



Performs script within if near to a specified Movie Clip

Target: _parent.ball

Other:

Criteria: ○ Bounding boxes ○ Distance between ● X,Y distance

X: 60 Y: 20

```
Scene_1
    onSelfEvent (e
    onSelfEvent (l
    ball2
        onSelfEvent
        onSelfEvent
        onSelfEvent
    ball
        onSelfEvent
        onSelfEvent
        onSelfEvent
```

```
onSelfEvent (press)
{
    startDrag(_target,true);
}
onSelfEvent (release)
{
    stopDrag();
}
onSelfEvent (enterFrame)
{
    if (_parent.ball.isNearThis(60,20)) {
        _alpha = 50;
```

The "if (isNearThis())" statement should now look like:

```
if (_parent.ball.isNearThis(60,20)) {
```

Note the shape of the area where the balls are considered 'Near'. Press 'Play' to preview the movie.

6. Press the 'Stop' button and return to the 'Script' Panel. Select the 'Distance between' option for the if (isNearThis()) statement. Enter a distance that is equal to the sum of the radius of each of the balls. In this example:

Ball has a width of 40 (radius = 20)
Ball2 has a width of 40 (radius = 20)
Distance is therefore 20 + 20 = 40.

The "if (isNearThis())" statement should now look like:

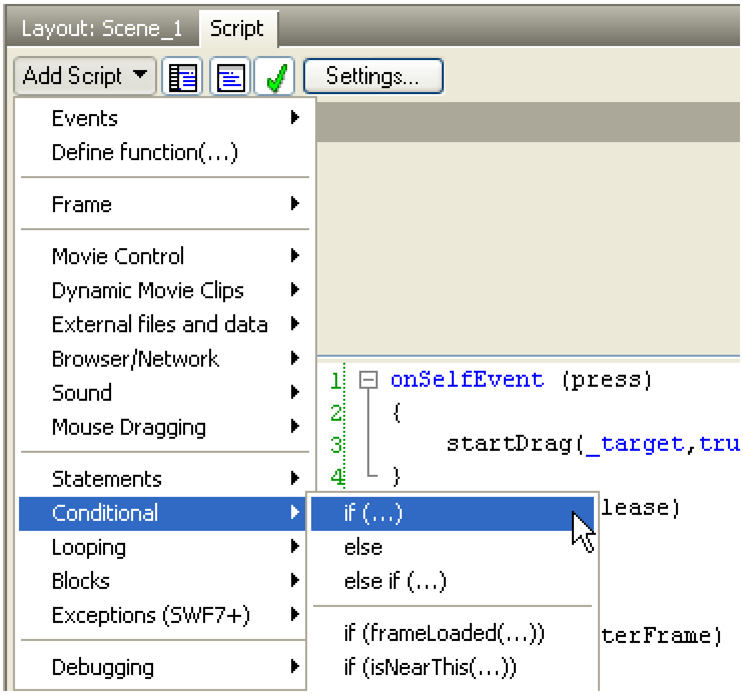
```
if (_parent.ball.isNearThis(40)) {
```

Press the 'Play' button and note that the blue ball changes transparency whenever the balls touch or overlap

7. Press the 'Stop' button and return to the 'Script' Panel. Select the ball Object in the 'Outline' Panel (red ball) and add the following script:

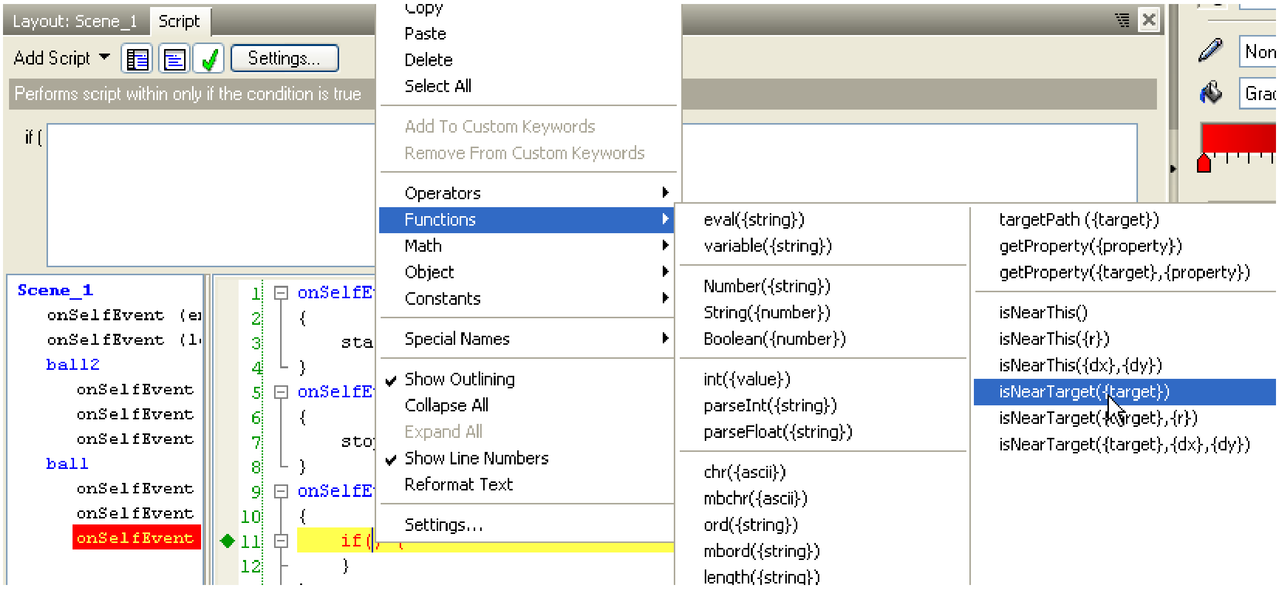
```
onSelfEvent (enterFrame)
{
  if (isNearTarget(_root.ball2._target)) {
    _alpha = 50;
  } else {
    _alpha = 100;
  }
}
```

To enter the isNearTarget() function you will need to enter a standard if (...) statement.



```
onSelfEvent (press)
{
    startDrag(_target,tru
}
lease)
terFrame)
```

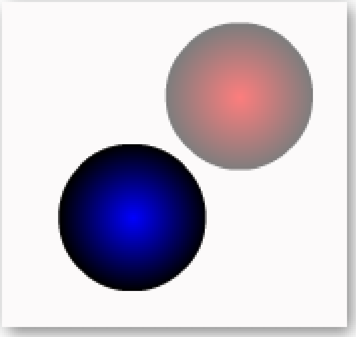
You can then enter the function name by typing it or by right clicking and selecting from the pop-up Menu



```
onSelfE
{
    sta
}
onSelfE
{
    sto
}
onSelfE
{
    if(
}
```

Now enter **_root.ball2._target** in the **Other:** field of the assist pane.

8. Press the 'Play' button and note that the blue ball changes transparency whenever the balls touch or overlap. Note that the red ball changes transparency when the bounding rectangles of the balls overlap.



The command "if (_root.ball2.isNearThis()) {" would have the same effect

9. Change the command within the if statement to:

```
if (isNearTarget(_root.ball2._target, 60, 20)) {
```

Press the 'Play' button and observe

Note: This behavior is identical to:

```
if (_root.ball2.isNearThis(60,20)) {
```

10. Change the command within the if statement to:

```
if (isNearTarget(_root.ball2._target, 40)) {
```

Press the 'Play' button and observe. Both balls should change transparency when they touch.

Note: This behaves is identical to:

```
if (_root.ball2.isNearThis(40)) {
```

Analysis

isNearTarget() and isNearThis() methods provide identical functionality using a different parameter interface. Both methods allow proximity to be defined via the size of:

- bounding rectangles (isNearTarget(target) or target.isNearThis())
- arbitrarily sized rectangle (isNearTarget(target, X, Y) or target.isNearThis(X, Y))
- the distance between the centers (isNearTarget(target, D) or target.isNearThis(D)).

Unlike the _target and _droptarget properties discussed in the previous tutorial, the 'Near' condition is not affected by which Object is being dragged.

4.6.7.12 Simple Gaming

Description

This is a step-by-step tutorial using the On Frame Event to show how Objects can be animated to provide simple gaming functionality.

Aim

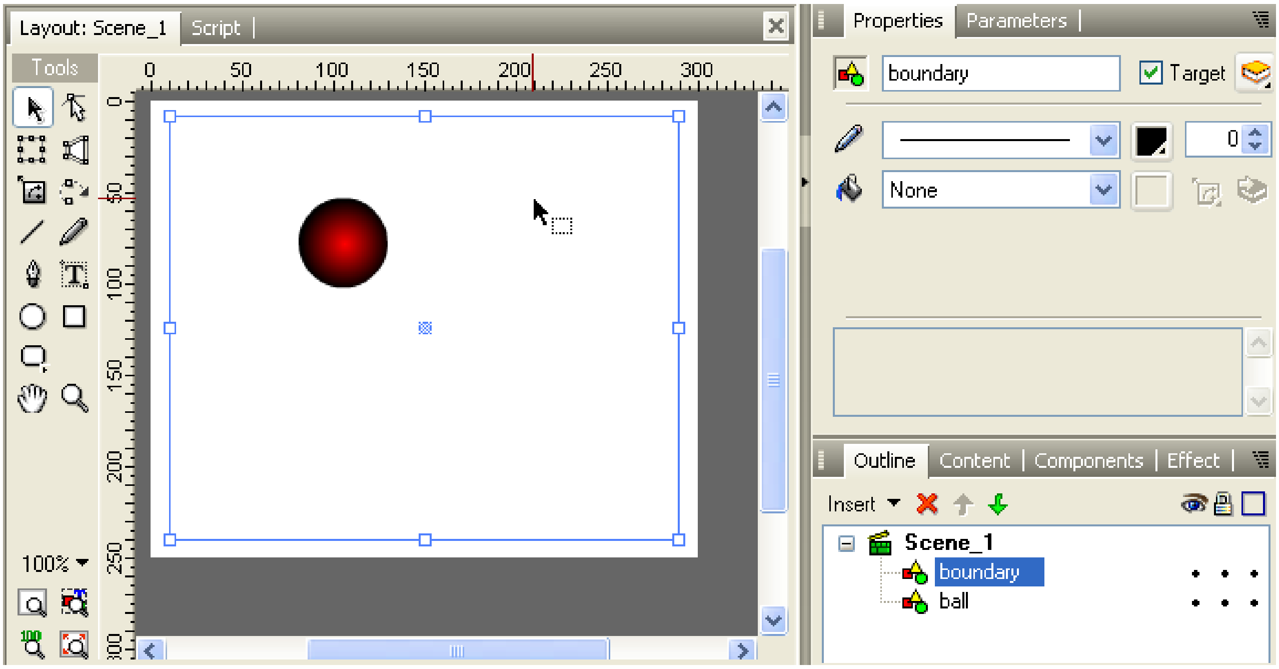
This tutorial uses the onSelfEvent (enterFrame) Event to animate Objects.

.swi file

"game.swi"

1. Open the "dragging.swi" file used in the Mouse Dragging tutorial and save the file as "mygame.swi"

2. Draw a rectangle to define the playing area. Name the rectangle "boundary" and tick the 'Target' checkbox



3. Rename the ball Object "bat" and change its Event scripts to be the following:

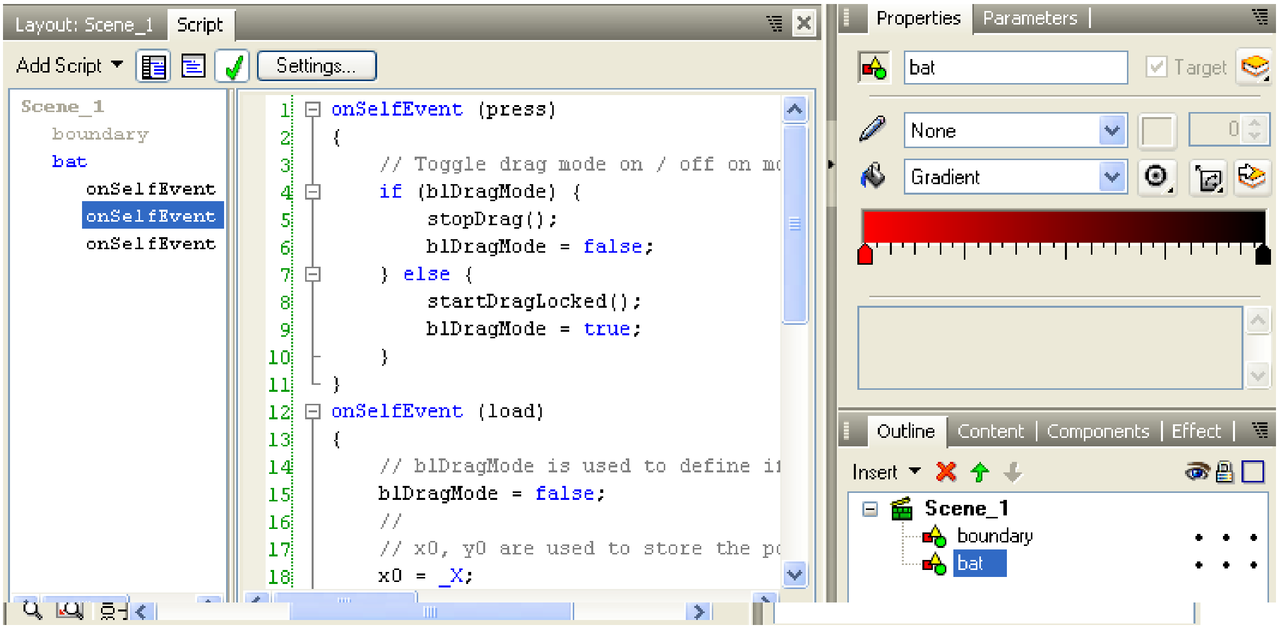
```
onSelfEvent (press)
{
        // Toggle drag mode on / off on mouse press.
        if (blDragMode) {
                stopDrag();
                blDragMode = false;
        } else {
                startDragLocked();
                blDragMode = true;
        }
}

onSelfEvent(load)
{
        // blDragMode is used to define if object is currently being dragged.
        blDragMode = false;
        //
        // x0, y0 are used to store the position of the object in the previous frame.
        x0 = _X;
        y0 = _Y;
        //
        // dx, dy are the x and y velocity of the object.
        // ie. dx is the number of pixels in the x direction that the object moved since the previous frame.
        dx = 0;
        dy = 0;
}

onSelfEvent (enterFrame)
{
        // Calculate and save the distance moved since the last frame. This is the velocity.
        dx = _X - x0;
        dy = _Y - y0;
        // Save the current position for calculation in the next frame
        x0 = _X;
        y0 = _Y;
}
```

Delete any previous onSelfEvent(release) {} event.

As the scripting is becoming more complex, it is useful to add comments using Add Script | Debugging | Comments. Comments are shown as a line starting with // or surrounded by /* */.

The **onSelfEvent (load)** function is used to declare variables and setup initial values.
The **onSelfEvent(press)** is used to toggle the dragging of the bat Object. Press the left mouse button once on the 'bat' object to start dragging. Press again to stop dragging. Note that the original **onSelfEvent(release)** has been deleted.
The **onSelfEvent (enterFrame)** function is called at the start of each Frame. This function is used to calculate and store the instantaneous speed of the bat. The speed is calculated by saving the number of pixels that the Object has moved since the function was previously called. The current position is then saved in x0, y0 for use when the function is called again on the next Frame.

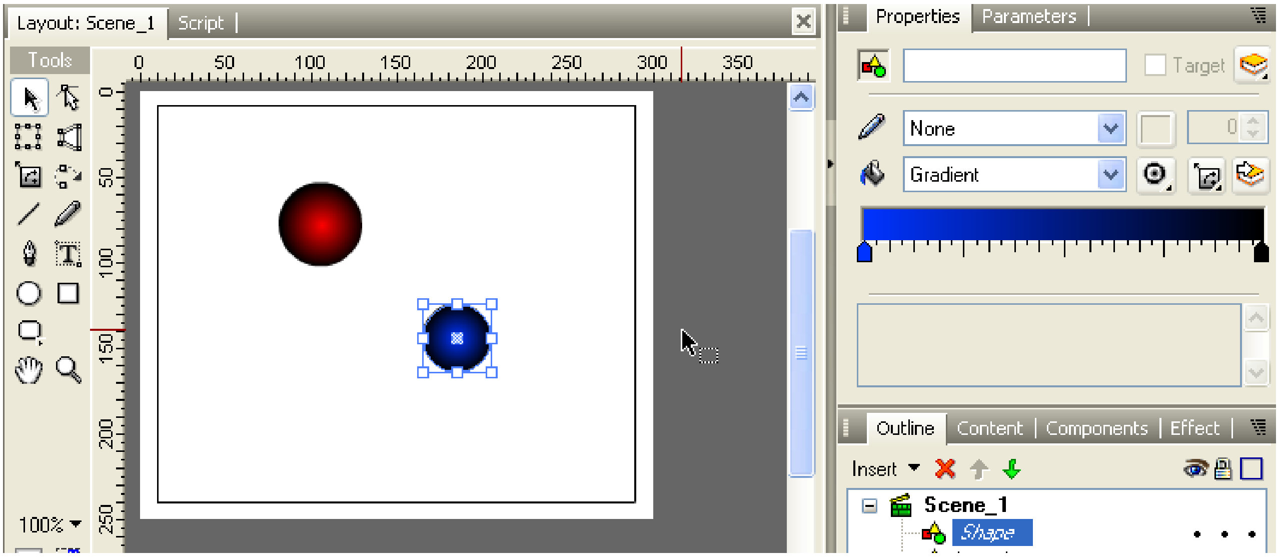
Note that in the image below, only a subsection of the script is shown.



bat

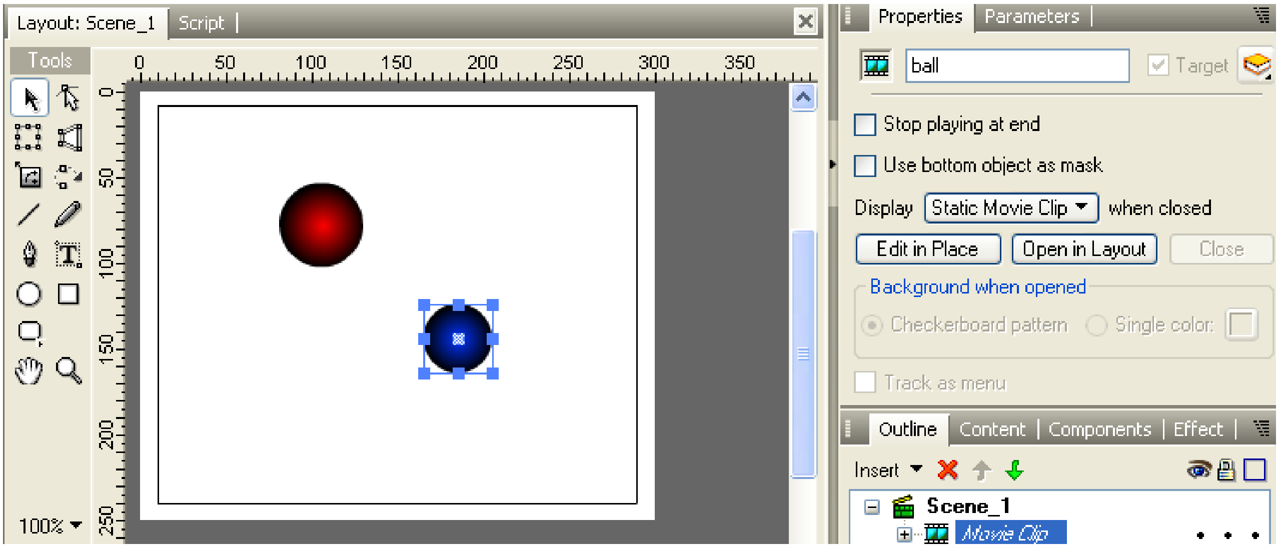
```
onSelfEvent (press)
{
    // Toggle drag mode on / off on mo
    if (blDragMode) {
        stopDrag();
        blDragMode = false;
    } else {
        startDragLocked();
        blDragMode = true;
    }
}
onSelfEvent (load)
{
    // blDragMode is used to define if
    blDragMode = false;
    //
    // x0, y0 are used to store the po
    x0 = _X;
```

4. Use the Circle Tool to create a new circle. Colour the circle blue. The diameter of this circle was 40



Note: If you create the circle by copying the bat Object, be sure to remove any script associated with the original Object

5. Use the Menu function Modify | Grouping | Group as Movie Clip to change the unnamed shape into a Movie Clip. Name the Movie Clip "ball". Using a Movie Clip for the ball allows multiple balls to be added later using references to a Library Symbol (instance).


ball

☑ Target

□ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip when closed

6. Add the following script to the 'ball' Movie Clip. The script below contains some hyperlinks to assist with understanding.

```
onSelfEvent (load)
{
            // Define variables and initial conditions.
            // dx is the pixels the object moves per frame in the x direction.
            // dy is the pixels the object moves per frame in the y direction.
            //
            // MinX, MaxX, MinY and MaxY are the minimum and maximum allowable positions for the object.
            // These are calculated based on the size of the boundary object and the radius of this shape.
            dx = 0;
            dy = 0;
            MinX = _root.boundary._X - _root.boundary._width / 2 + this._width / 2;
```

```
        MaxX = _root.boundary._X + _root.boundary._width / 2 - this._width / 2;
        MinY = _root.boundary._Y - _root.boundary._height / 2 + this._width / 2;
        MaxY = _root.boundary._Y + _root.boundary._height / 2 - this._width / 2;
        // batradius is the combined radius of the bat and the ball
        batradius = this._width / 2 + _root.bat._width / 2;
}

onSelfEvent (enterFrame)
{
        // Move ball to new position
        _X += dx;
        _Y += dy;
        //
        // Check for boundary collision. If collision detected, reverse velocity
        // Use of Math.abs is more robust than simply -dx, -dy.
        // This will work correctly if dx or dy > R but < 2 * R and ball straddles boundary.
        if (_X < MinX) {
                // Force dx to be positive.
                dx = Math.abs(dx);
        }
        if (_X > MaxX) {
                // Force dx to be negative.
                dx = -Math.abs(dx);
        }
        if (_Y < MinY) {
                // Force dy to be positive.
                dy = Math.abs(dy);
        }
        if (_Y > MaxY) {
                // Force dy to be negative.
                dy = -Math.abs(dy);
        }
        //
        // Check for collision with bat.
        if (_parent.bat.isNearThis(batradius)) {
                // We have collided with bat, change ball velocity.
                dx = CalcNewDx();
                dy = CalcNewDy();
        }
}
function CalcNewDx()
{
        // Function to calculate the new x velocity of the ball
        // This is not correct physics, (no bounce). Just take on the velocity of the bat
        return _root.bat.dx;
}
function CalcNewDy()
{
        // Function to calculate the new y velocity of the ball
        // This is not correct physics, (no bounce). Just take on the velocity of the bat
        return _root.bat.dy;
}
```

Again, many comments have been used to describe the script. To save time they can be omitted, but they do help explain the code.

Note: Comments do not affect the size of the exported .swf file hence they will not slow down the loading of the movie once it is published to the web. For this reason, there is no advantage in removing comments in the hope that it will improve the performance of your movie.

For this Object the **onSelfEvent (load)** function is used to declare and initialise variables. dx and dy are used to hold the Object's velocity. These are initialised to 0.

MinX, MinY, MaxX and MaxY are used to hold the boundary area that the ball is allowed to travel in. These points are based on the size of the boundary Object and are corrected to take into account the radius of the ball.

Batradius is used to hold the combined radius of the bat and the ball. This is calculated from bat._width / 2 + ball._width / 2. This radius is used when checking a collision condition. A collision has occurred when the two Objects are within Batradius of each other (ie. touching or overlapped).

The **CalcNewDx()** and **CalcNewDy()** functions can be entered via **Add Script | Define function (...)** (Leave all parameter fields blank). The functions are used to calculate the new x and y velocity of the ball after a collision with the bat. Functions are used here as they provide a way of grouping sections of code. This can be used to help keep each module small enough to be easily understood. For maximum readability, no function should extend beyond 1 page of display. Initially, the functions do not calculate the physics associated with the ball bouncing off the bat. The bat is assumed to be a 'sticky' bat, i.e. the ball simply adopts the current velocity of the bat.

The **onSelfEvent (enterFrame)** Event function is called on each Frame. It performs the following tasks:

- updates current ball position based on current dx, dy values
- checks for collision with boundary. Reverses dx, dy if collision is detected
- checks for collision with bat. Updates dx, dy according to values returned from CalcNewDx() and CalcNewDy() functions.

Note: When analysing the script, `a += b;` means `a = a + b;`.

7. Test the game by selecting the 'Layout' tab and pressing the 'Play' button. The balls should be stationary. Click on the red ball (bat object) with the mouse. The bat will now follow the mouse pointer until you click the mouse again. Use the bat to hit the blue ball. The blue ball should continue to bounce around the court until it meets the bat again

8. Press the 'Stop' button. Select the ball Movie Clip in the 'Layout' Panel and right click. **Select Library | Add to library**. Select the checkbox "Add to Library and create a link" then press OK. The original ball movie clip has now been added to the library and it is replaced with a link to that library reference. Now from the Main Menu go **Insert | Library Symbol...** and choose the ball movie clip. Press OK. You will now see an identical item named "*Instance of ball"* appear on the stage. If the new instance appears off the page, drag it back into the playing area. Press the 'Play' button. There are now two ball Objects that can be hit around the court. Note that the each of the references behave independently of each other as each of their variables and associated properties (dx, dy, _X, _Y etc.) are local to the individual Object.

Analysis

The onSelfEvent (enterFrame) Event Handling function can be used to animate Objects by altering their _X and _Y properties on a Frame-by-Frame basis.

The onSelfEvent (load) Event Handling function is useful for defining and initialising variables. If the variables x0 and y0 were defined and initialised within the **onSelfEvent (enterFrame)** function, their previously stored values would be lost when the function is called on the following Frame.

User defined functions (e.g. **CalcNewDx()** and **CalcNewDy()**) can be used to group sections of code and improve readability.

A Instance of a Movie Clip can behave independently (in a scripting sense) of the original Movie Clip and is a useful way of creating new game Objects.

4.6.7.13 Physics Properties

Description

Shows how various gaming functions can be simplified by using Physics Properties.

Aim

Introduce the concepts of velocity, acceleration and friction.
Introduce the available SWiSH Max specific Physics Properties and demonstrate how they can simplify such scripting.

.swi file

"game1.swi"

Step 1

Continue from the previous tutorial or open the file "game.swi"
Save the file as "mygame2.swi"

Step 2

Enable Physics properties for the movie. See here for instructions.

Enable Physics properties for the bat, and the ball Symbol within the library. See here for detailed instructions. Note that to alter the properties for the ball Library Symbol it is necessary to right click on either ball or Instance of ball, then select **Library | Edit Linked Symbol ball.** Once you have enabled the physics properties, you can close the Edit Linked Symbol layout panel by pressing the ☒ button on the layout panel. The main layout panel should remain.

Step 3
Modify the bat object's onSelfEvent (enterFrame) Event so that it calculates velocity based on pixels per second (as opposed to pixels per frame as it currently does). All physics properties are based on per second values.

```
onSelfEvent (enterFrame)
{
        // Calculate and save the distance moved since the last frame. This is the velocity.
        // The _delta property is used to convert these values to pixels per second.
        dx = (_X - x0) / _delta;
        dy = (_Y - y0) / _delta;
        // Save the current position for calculation in the next frame
        x0 = _X;
        y0 = _Y;
}
```

The physics property _delta represents the time period since the last frame. Dividing by this value converts the dx, dy velocities from pixel per frame into pixel per second velocities.
Note that the Physics properties must be enabled for the bat object to allow access to the _delta property.

Step 4
Modify the "ball" Movie Clip's onSelfEvent (enterFrame) Event so that it uses the physics properties _vX and _vY instead of dx and dy from the previous tutorial.

```
onSelfEvent (enterFrame) {
        // Move ball to new position (This is now done automatically by _vX and _vY properties).
        //
        // Check for boundary collision. If collision detected, reverse velocity
        if (_X < MinX) {
                // Force _vX to be positive.
                _vX = Math.abs(_vX);
        }
        if (_X > MaxX) {
                // Force _vX to be negative.
                _vX = -Math.abs(_vX);
        }
        if (_Y < MinY) {
                // Force _vY to be positive.
                _vY = Math.abs(_vY);
        }
        if (_Y > MaxY) {
                // Force _vY to be negative.
                _vY = -Math.abs(_vX);
        }
        //
        // Check for collision with bat.
        if (_parent.bat.isNearThis(batradius)) {
                // We have collided with bat, change ball velocity.
                _vX = CalcNewDx();
                _vY = CalcNewDy();
        }
}
```

Note that the lines

```
        // Move ball to new position
        _X += dx;
        _Y += dy;
```

Have now been removed. The dx, dy lines can also be removed from the ball.onSelfEvent (load) event handling function as they are no longer used.

Step 5
Test the game.
Select the layout tab and press play. The game should behave in a similar fashion to the previous example.

Step 6
Press the stop button.

Analysis
In this tutorial, physics properties were introduced. These are SWiSH Max specific. The properties _vX and _vY were used for Velocity, and _delta was used for measuring time. These properties, as used in this tutorial, determine how fast the ball moves - and in which direction. It is important that Physics Support be enabled for both the final Exported SWF and the on the Object's Export Panel in order for these scripts to work properly.

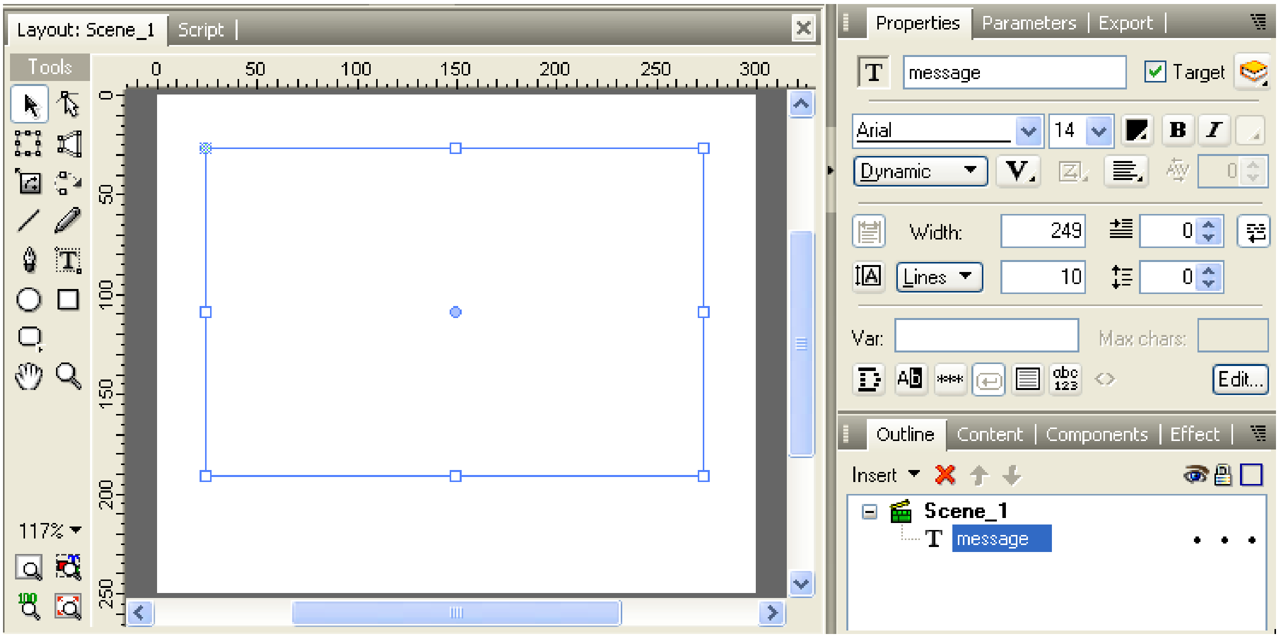
4.6.7.14 Data Transfer - text

Description
Demonstrates how data can be transferred to and from .htm, .asp, .php or text files that are resident on a server.

Aim
Introduces the methods loadVariables() and shows how you can load an external .txt file into your movie at runtime.

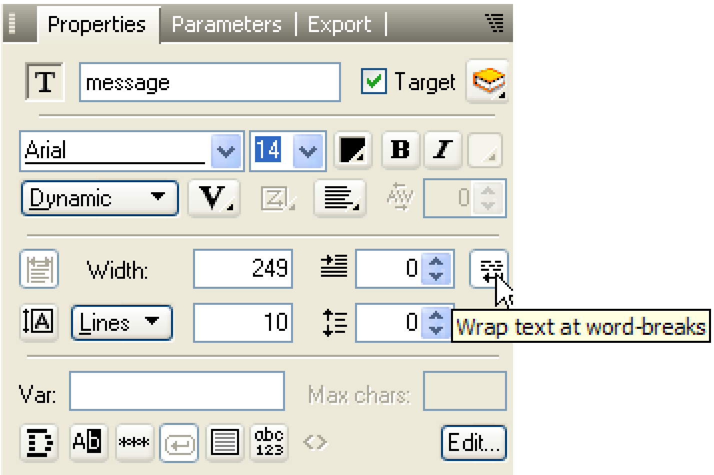
1. Create a New movie and save it as "mytxtmessage.swi".
Note: The Movie Properties must be set so that it exports as swf5 or later.

Then Insert a Dynamic Text object. In the 'Dimensions' options, untick the Auto-Height icon and choose 'Lines' rather than 'Height' - Set it for several lines to accommodate extra text. Name the text object 'message' and tick the 'Target' option as shown below:



message ☑ Target
Arial 14
Dynamic
Width: 249 0
Lines 10 0
Var:
Max chars:
Scene_1
message

2. In the 'Formatting' options, make sure the 'Word-Wrap' option is selected. Press the 'Black border with white background' option and assign a variable "mymessage" to this text object as shown below:



T message ☑ Target

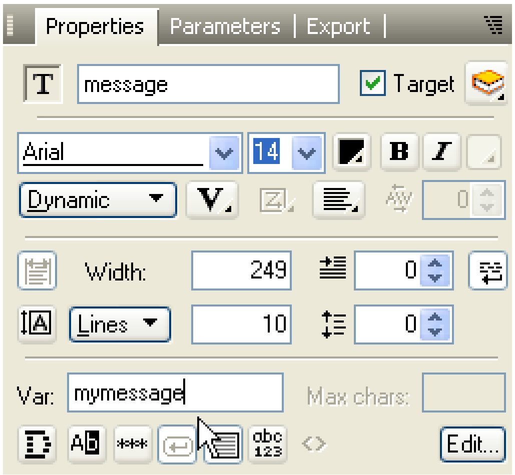
Arial 14

Dynamic 0

Width: 249 0

Lines 10 0

Var: Max chars:



T message ☑ Target

Arial 14

Dynamic 0

Width: 249 0

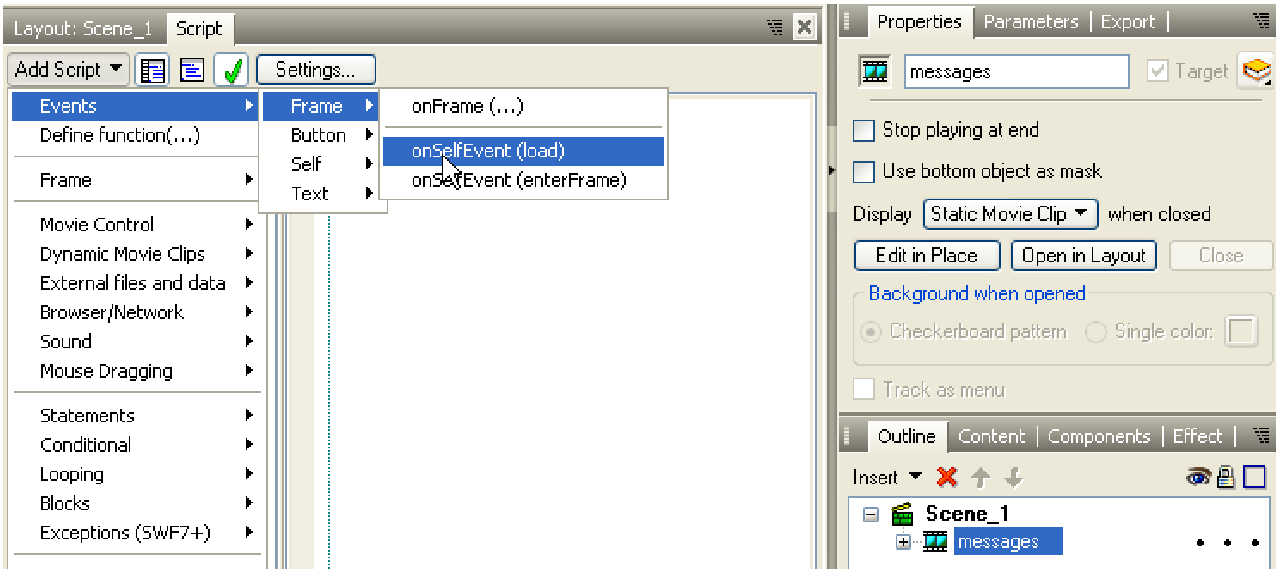
Lines 10 0

Var: mymessage Max chars:

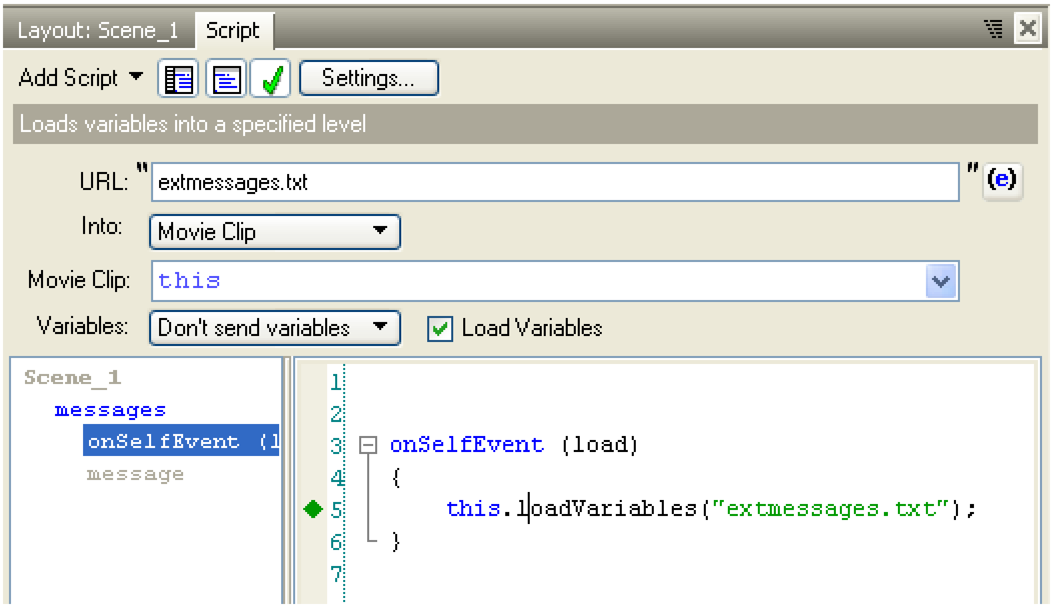
3. Use the Modify menu and select **Grouping | Group as Movie Clip**, then name the object "messages" on the 'Properties' panel.

4. With the Movie Clip object selected in the 'Outline' panel, open the 'Script' panel and press the 'Add Script' button. Select **Events | Frame | onSelfEvent (load)** from the menu.

5. Right-Click on the onSelfEvent (load) action and select **Add Script | External files and data | Load/unload Movie Clip | loadVariables(...)** from the menu.



6. In the loadVariables() options, type "extmessages.txt" in the URL field, and type "this" into the Movie Clip field. If the option "Load variables only" is not ticked, do so now. The 'Script' panel should look similar to this:



Loads variables into a specified level

URL: "extmessages.txt"

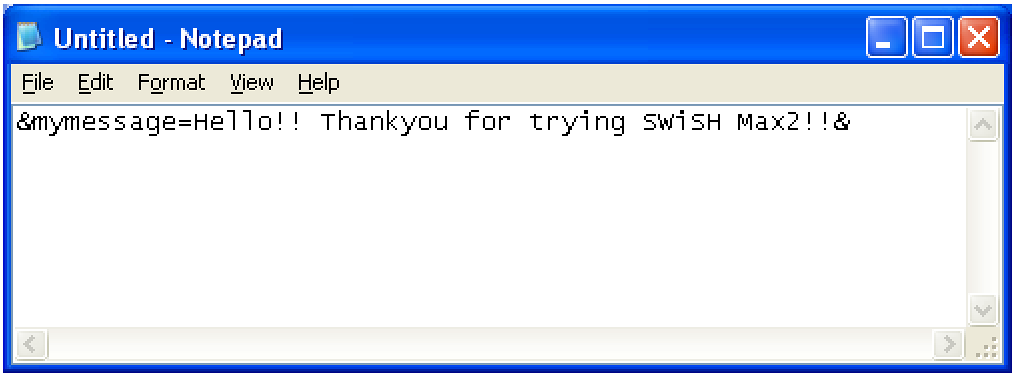
Into: Movie Clip

Movie Clip: this

Variables: Don't send variables ☑ Load Variables

```

onSelfEvent (load)
{
    this.loadVariables("extmessages.txt");
}

```

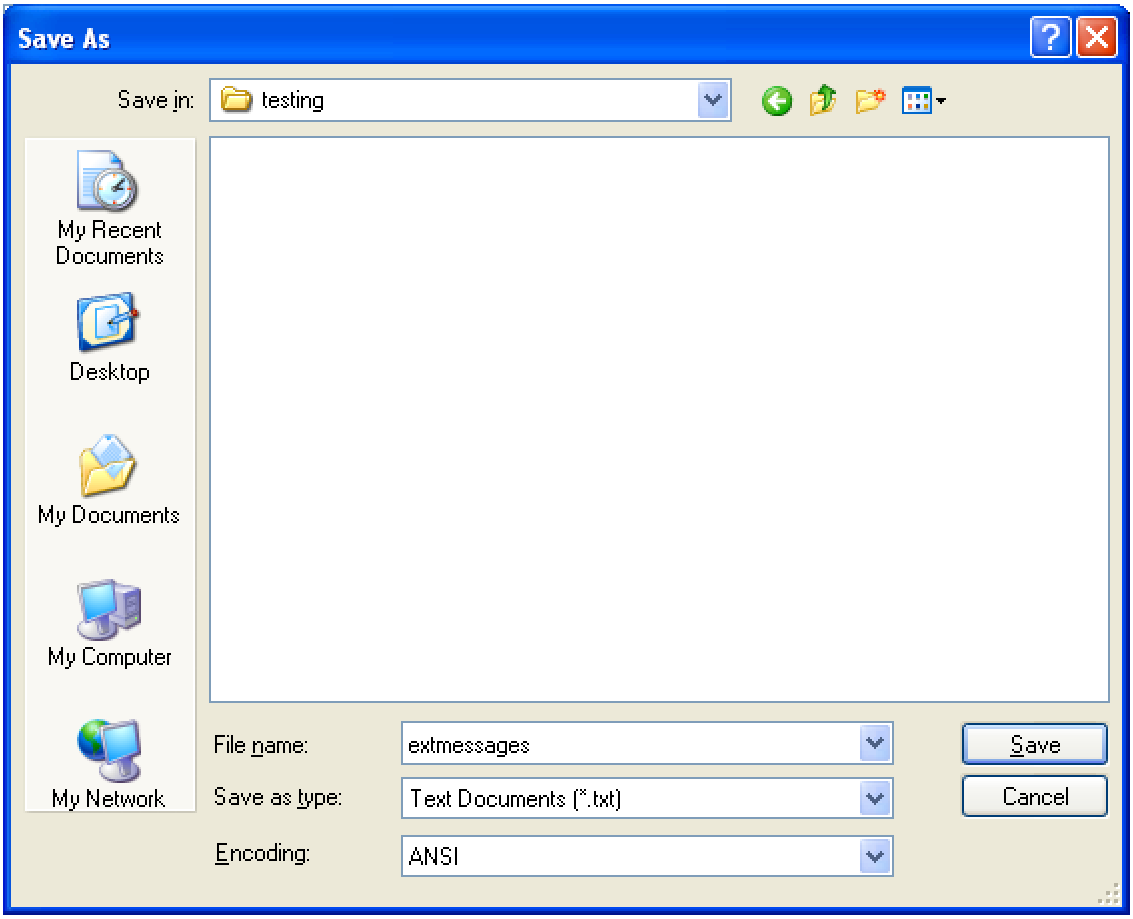
7. Next, open Windows Notepad (or any text-editing program) and type what is shown in the image below:



```
&mymessage=Hello!! Thankyou for trying SWiSH Max2!!&
```

Note: Notice that in the image above, the text is defined by two ampersand characters "&". When loading external .txt files into a SWF movie, you are not allowed to use the ampersand character in the text messages, because that character tells the Flash player that the text has ended. So, be sure to use "and" instead of "&" when typing the body of the text. Also, note that we have used the Variable assigned to the Dynamic text object - not the name.

8. After typing the body of the text message, save this file as "extmessages.txt" (which is the URL used above for the loadVariables() action) and note the folder you save this .txt file to.



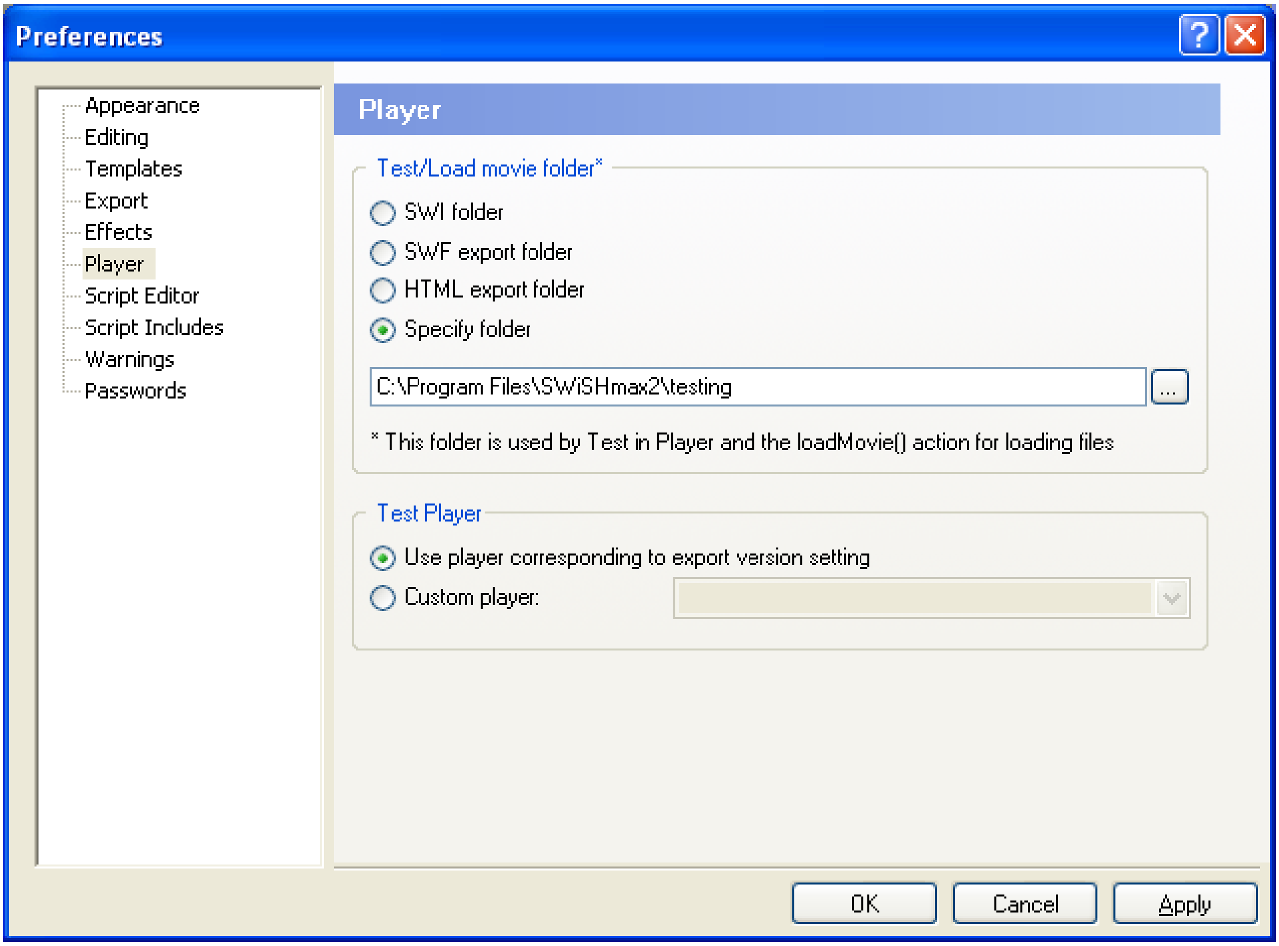
Save in: testing

File name: extmessages

Save as type: Text Documents (*.txt)

Encoding: ANSI

9. In SWiSH Max, open the Tools menu, select Preferences, then open the Player options. Tick the "Specify folder" option and press the browse button to locate the folder the "extmessages.txt" file was saved in on Step #8 above.



Player

Test/Load movie folder*

○ SWI folder
○ SWF export folder
○ HTML export folder
● Specify folder

C:\Program Files\SWiSHmax2\testing

* This folder is used by Test in Player and the loadMovie() action for loading files

Test Player

● Use player corresponding to export version setting
○ Custom player:

10. Make sure you are in the 'Layout' panel in SWiSH Max then press the 'Play' button in the Control Toolbar to preview the movie. The message typed in to the .txt file should be displayed inside the Dynamic text field.

Analysis
When the 'Play' button is pressed, the onSelfEvent (load) event is triggered for the Movie Clip "messages". The loadVariables() action was set to only load variables which will load the variables defined in the .txt file at the specified URL.

4.6.8 Script Reference

The scripting used within SWiSH Max is generally compatible with Adobe™ Flash Actionscript. Some areas of difference relate the the support of depreciated objects and naming conventions. The main areas of difference are described here.

Scripting commands can be divided into the following groups:

- operators: Arithmetic, Comparison, Equality, Logical
- functions and methods: General Functions, Movie Clip Object, String Object, Math Object, Date Object, Timer Object, Text Object
- Object properties: General Object Properties, Movie Clip Object Properties, Text Field Properties and Methods

- Constants and Object synonyms

Each of the Scripting commands is described in detail in the Script Reference Dictionary. This dictionary is arranged in alphabetical order for ease of reference.

Any Scripting Object can have Event Handler functions associated with it. These Event Handler functions execute script commands in response to defined events.

General definitions can be found in the Definitions Section.

Useful formulas and other notes can be found in the Useful Stuff section.

4.6.8.1 Arithmetic Operators

- + (addition)
- - (minus)
- * (multiplication)
- / (division)
- % (mod)
- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)
- << (bitwise left shift)
- >> (bitwise right shift)
- >>> (bitwise unsigned right shift)
- ++ (increment)
- -- (decrement)
- plus (addition)
- add (concat strings)

4.6.8.2 Comparison Operators

- < (less than)
- <= (less than or equal to)
- > (greater than)
- >= (greater than or equal to)
- lt (less than - string specific)
- le (less than or equal to - string specific)
- gt (greater than - string specific)
- ge (greater than or equal to - string specific)

4.6.8.3 Equality Operators

- == (equality)
- != (inequality)
- eq (equal - string specific)
- ne (not equal - string specific)

4.6.8.4 Logical Operators

- && (short-circuit AND)
- || (logical OR)
- ! (logical NOT)
- ?: (conditional)

4.6.8.5 Functions

Scripting functions:

- eval(string)
- variable(string)
- getProperty()
- Number(string)
- String(number)
- int(value)
- parseInt(string)
- parseFloat(string)
- isNearThis()
- isNearTarget()

4.6.8.6 Constants

Scripting constants:

- true
- false
- PI
- SQRT2
- SQRT1_2
- TAB
- NEWLINE
- $version

Note that additional constants can be defined using the const keyword.

4.6.8.7 Object Properties

All Scripting Objects inherit the following properties:

Property	Read Only	Physics Option	Description
_age	yes	yes	Age of Object since placed (in seconds)
_alpha	no	yes	Sets the Object's transparency (0 = transparent, 100 = solid)
_currentframe	yes	no	Returns the current frame number being played
_delta	yes	yes	Time elapsed since last form displayed (in seconds)
_droptarget	yes	no	Name of the Movie Clip/ Sprite that draggableInstance has been dragged over
_focusrect	no	no	Global property that defines whether a yellow rectangle appears around the button that has keyboard focus. Default value is true
_framesloaded	yes	no	The number of Frames that have currently been loaded
_height	yes	no	Returns the current height of the Movie Clip / Sprite or Object
_highquality	no	no	Specifies the level of anti-aliasing applied to the current Movie
_name	no	no	Current name of the Sprite/ Movie Clip or Object
_parent		no	Synonym for the Sprite/ Movie Clip/ Object that contains this Object, i.e. the parent
_rotation	no	yes	Sets/ Returns the current rotation angle of the Movie Clip / Sprite or Object
_soundbuftime	no	no	Specifies the number of seconds of streaming sound to pre-buffer
_target	yes	no	Target path of the Sprite/ Movie clip or Object

_time	yes	yes	Time since the start of the whole Movie (in seconds)
_totalframes	no	no	The number of Frames contained within the Sprite/ Movie Clip
_url	yes	no	Returns the URL of the .swf that the Movie Clip was downloaded from
_visible	no	no	Sets/ returns the current visibility status of the Sprite/ Movie Clip or Object
_width	yes	no	This read-only property returns the current width of the Movie Clip / Sprite or Object
_x	no	yes	Sets/ returns the current X coordinate of the Movie Clip / Sprite or Object
_xscale	no	yes	This property returns or sets the horizontal scale of the indicated Movie Clip / Sprite/ Object
_y	no	yes	Sets/ returns the current Y coordinate of the Movie Clip / Sprite or Object
_yscale	no	yes	This property returns or sets the vertical scale of the indicated Movie Clip / Sprite/ Object

4.6.8.8 Events

Multiple event handling systems have been incorporated in the Flash Player to handle situations where external events such as keyboard typing, mouse movement, and mouse button presses interact with your movie.

These event handling systems are:
1. Event Handlers
2. Event Methods. These are sometimes referred to as Event Callbacks, Event Handler Methods or Event Handler Callbacks.
3. Event Listeners

Event Handlers were used in earlier Flash implementations but this is gradually being phased out in favor of Event Methods. With the advent of Action Script 3, Event Handlers (Item 1.) will no longer be supported.

Event Handlers are easier to use than the Event Methods and Event Listeners, but the latter may provide additional flexibility in some situations. Each of the event systems are described in the following sections.

4.6.8.8.1 Event Handlers

Each Scripting Object can be supplied with an Event Handling routine so that specific script Actions occur in response to pre-defined Event conditions.

The following Event Handling routines are supplied:
- onFrame(): defines the Actions to occur in a specific Frame
- on(mouseEvent): used for buttons only. Defines Actions to be performed when a defined mouse or keyboard Action occurs. The Actions are always applied to the top level or _root Movie
- onSelfEvent (enterFrame): defines the Actions to occur at the start of each Frame
- onSelfEvent (load): defines the Actions to occur when the Object is loaded
- onSelfEvent (mouseEvent): defines Actions to be performed when a defined mouse or keyboard Action occurs
- onSelfEvent (changed) and on (changed) are used for Text Field Objects only. Defines the Actions to occur when the contained text changes.

4.6.8.8.2 Event Methods

Buttons, Movie Clips and Text Fields each have methods associated with them that allow the definition of Event Callback functions. These Event Callback functions are called when the associated event has occurred. In some cases the Callback function may contain a parameter.

Event Methods have the following advantages when compared to Event Handlers:
1. They can be defined for an individual object. ie. An object created from an instance of another object can

have its own specific handlers defined.
2. The handling function can be changed dynamically while the main movie is in progress. This could allow the default event handling to change after a series of other events has occurred.

The general format for defining a Callback function is as follows:

```
objectname.onEventMethod = function([optional parameter(s)]) {
        // function script
}       // end of callback function.
```

The definition must be performed within a script function. The onSelfEvent (load) function is a convenient function to use for this purpose.

Example
A callback function for the onPress event is defined for the movie clip mc. This function is called each time the mouse button is pressed while the mouse is over the movie clip mc.

If the callback function is defined in the _root script area then the following would be used:

```
onSelfEvent (load) {
        mc.onPress = function() {
                mc._xscale += 10;      // make movie clip 10% larger on each press
        }
}
```

Defining the callback function outside of the object has the advantage that the callback function can be re-defined at a later time if necessary.

Alternatively the script could be defined within the script area of the object mc as:

```
onSelfEvent (load) {
        this.onPress = function() {
                this._xscale += 10;    // make movie clip 10% larger on each press
        }
}
```

or

```
function onPress() {
        this._xscale += 10;
}
```

The available methods, a description of the event that they handle, the associated parameters and the objects that they apply to are listed in the table below.

Method	Description	Parameters	Objects
onChanged	Called when the text in the text field changes.		Text Field
onData	Called when the Movie Clip receives data from a loadVariables() or movieClip.loadVariables() function call.		Movie Clip
onDragOut	This is invoked when the mouse is clicked over the object then dragged outside while the mouse button remains pressed.		Button Movie Clip
onDragOver	This is invoked when the mouse is clicked outside the object then dragged onto the object while the mouse button remains pressed.		Button Movie Clip
onEnterFrame	Called when each frame within the movie clip is processed. This occurs at the global frame rate of the movie.		Movie Clip
onKeyDown	Called when the object has focus and a keyboard object is pressed.		Button Movie Clip
onKeyUp	Called when the object has focus and a keyboard object is released.	newfocus:Object	Button Movie Clip

onKillFocus	Called when the object loses keyboard focus.		Button Movie Clip Text Field
onLoad	Called when the movie clip is loaded and appears on the timeline.		Movie Clip
onMouseDown	Called when the mouse button is pressed.		Movie Clip
onMouseMove	Called when the mouse is moved.		Movie Clip
onMouseUp	Called when the mouse button is released.		Movie Clip
onPress	Called when the object is pressed.		Button Movie Clip
onRelease	Called when the object is released.		Button Movie Clip
onReleaseOutside	Called when the mouse button (initially pressed while over the object) is released outside of the object.		Button Movie Clip
onRollOut	Called when the pointer moves outside of the object.		Button Movie Clip
onRollOver	Called when the pointer moves over the object.		Button Movie Clip
onScroller	Called when one of the text field scroll properties changes.	scrolledField:Text Field	Text Field
onSetFocus	Called when the object receives keyboard focus.	oldfocus:Object	Button Movie Clip Text Field
onUnload	Called after the movie clip is removed from the timeline. Call occurs on the frame following the clip removal.		Movie Clip

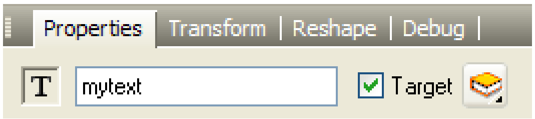
4.6.8.9 Definitions

This section defines the following features of SWiSH Max:

- Script Object
- Movie Clip (previously called Sprite)

4.6.8.9.1 Script Object

A Script Object is any drawing Object that is named and has the 'Target' checkbox ticked.



mytext ☑ Target

A Movie Clip is also a Script Object ('Target' checkbox is not required), as is the main Scene.

A Script Object can have Event Handler and private functions associated with it. Such an Object becomes the basic element of scripting.

All Script Objects have Object properties. In addition, TextField Objects can also have Text Field Properties and Methods.

When an object has the target option checked, it is wrapped in a Movie Clip. This conversion will cause grouped Objects and text to lose their Complex Effect behavior. In most cases this wrapping adds functionality as all properties and methods associated with Movie Clips can now apply to the object or shape.

In the case of Button and TextField objects, the wrapping prevents normal access to the native textfield and button properties and methods. In these situations, access to these native properties and methods can be obtained in a number of ways:
1. Set the expose SWF6 properties checkbox in the Movie Export Options.

2. Access the native properties or methods via the _button and _text properties.

Expanding on this explanation for a text object, when the '[x] target' is set, SWiSH Max2 wraps a text field in a movieclip. The inner text field is given the name "_text". So access to the text field in the movieclip wrapper is via the ._text member. Obviously, if the [x] target is NOT set, then access the text field directly. So the way to write a script changes depending on whether the [x] target set or not. As a 'convenience' the "[x] Expose SWF6 properties" setting generate a class for the wrapper. This class defines members with setter and getter functions that get at the inner _text. Consequently if this option is set it is not necessary to change how a script is written. The down side for this situation is:

- in some cases, it is still necessary to directly change the inner _text text field
- this adds performance overheads by calling the getter and setter each time there is access to a member or method for the text
- this adds to the SWF file size

Note: Group Objects have certain limitations when used as a target from within scripts. It is not recommended to use relative paths to target a Group Object or to execute actions from a Group Object. Instead, try to use absolute (or full) paths when targeting to or from a Group Object.

Note: There are reserved names that can **NOT** be use as the name or variable for scripted objects. The list of reserved words is as follows:
Math, String, Timer, Date, Key, length, text, htmltext, scroll, maxscroll, _root, _parent, this, do, else, for, if, not, playSound, preloadContent, return, setLabel, setLabelAnchor, stopSound, tellTarget, trace, var, while, with, return, true, false, PI, E, LN10, LN2, LOG2E, LOG10E, SQRT2, SQRT1_2, newline, tab, NULL

4.6.8.9.2 Movie Clip (Sprite)

Note: 'Sprite' is a deprecated term for 'Movie Clip'. Sprite was used in SWiSH Max version 1. Although sprite scripting commands still exist for backward compatibility, Movie Clip scripting commands should be used in their place as this is more compatible with Flash.

A Movie Clip is a Simple Object that has its own Timeline and is a collection of Objects with Effects.

SWiSH Max Movie Clips support most of the flash Movie Clip properties and methods. A list of supported items can be found in Movie Clip / Sprite Object.

Note: There are reserved names that you are NOT allowed to use as the name or variable for scripted objects. The list of reserved words is as follows:
Math, String, Timer, Date, Key, length, text, htmltext, scroll, maxscroll, _root, _parent, this, do, else, for, if, not, playSound, preloadContent, return, setLabel, setLabelAnchor, stopSound, tellTarget, trace, var, while, with, return, true, false, PI, E, LN10, LN2, LOG2E, LOG10E, SQRT2, SQRT1_2, newline, tab, NULL

4.6.8.10 SWiSH Max Specific

Some Scripting operators and functions are handled differently between Flash MX and SWiSH Max.

The main differences are explained in the following sections.

4.6.8.10.1 Sprites

Previous versions of SWiSH Max used the term Sprite in place of the Flash term Movie Clip. SWiSH Max now supports the name Movie Clip. Script commands containing the word Sprite are still supported for backward compatibility however the corresponding Movie Clip methods should be used in their place.

Sprites do not support all of the Movie Clip properties and methods. A list of supported items can be found in Movie Clip / Sprite Object.

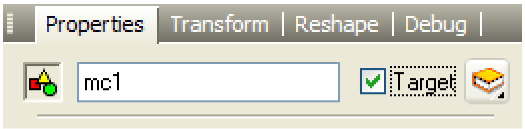
4.6.8.10.2 Shapes / Objects

Adobe Flash MX actionscript is associated with the base Movie and Movie Clips. Only the base Movie, Movie Clips and other supported Objects can have properties and Scripting applied.

SWiSH Max also allows Scripting and Object properties to be associated with simple Shape and Text Objects. This is an extension to Flash MX actionscript.

For Shape and Text Objects to have Scripting and dynamic properties, the Object must have a name and the 'Target' checkbox found under the 'Shape' tab must be checked.

The 'Target' checkbox is not visible until a name has been entered. Once the 'Target' box has been checked, the Object becomes a Scripting Object.



For buttons and text objects, some of the raw properties are obscured by this conversion to a Scripting Object. To access the raw properties and methods you must use _button and _text as appropriate. See Scripting Object for more information.

4.6.8.10.3 Variables and String Conversion

Variables can be used to store numbers and strings.

The exact item stored by the variable is converted, as appropriate, depending on the context in which the variable is used.

The operators +, <, <=, ==, !=, => and > can take numbers, strings or variables as their arguments.

In SWF4 export only, variables are assumed to be numeric and conversion is done as appropriate. In SWF5 or later, the type of the actual value stored in the variable is used.

The + operator has String Constant Precedence.
The <, <=, ==, !=, => and > operators have Numeric Constant Precedence.

The operators add, lt, le, eq, ne, ge, gt assume that variables are strings and have String Constant Precedence. The use of these depreciated commands in this way is a <%SWISHSCRIPT%> extension.

Consider the following trace statements:

```
a = 11;
b = "22";
trace(a + "987");      // string parameter causes conversion of 11 to "11" result is 11987
trace(a ADD 987);      // ADD operator converts both a and 987 to strings, result is 11987
trace(a + 987);        // both parameters are numbers, result is 998
trace("987" + b);      // both parameters are strings, result is 98722
trace(987 ADD b);      // ADD operator converts 987 to string, result is 98722
trace(987 + b);        // variable is assumed to be number. "22" converted to number, result is 1009
trace(a + b);          // variables assumed to be numbers. "22" converted to 22, result is 33
```

During conversion from string to number, 0 is returned if the string does not represent a number.

```
onSelfEvent (load) {
  five = 5;
  six = 06;
  fivestr = "5";
  sixstr = "06";
  if ("5" < "6") {
    trace("test1");
  }
  if ("5" < "06") {
    trace("test2");
  }
  if (5 < 06) {
    trace("test3");
  }
  if (5 lt 06) {
    trace("test4");
  }
  if (fivestr < sixstr) {
    trace("test5");
  }
  if (fivestr lt sixstr) {
    trace("test6");
  }
  if (five < six) {
    trace("test7");
  }
  if (five lt six) {
    trace("test8");
  }
}
```

returns

```
test1
test3
test4
test5
test7
test8
```

4.6.8.10.3.1 Conversion To Numeric

If the variable contains a string that can be interpreted as a numeric value, it is converted to that number:

e.g. "-2.2" is converted to -2.2.

If the string contains any other characters that indicate that it is not a number, it is converted to 0:

e.g. "-2.2a" is converted to 0.

4.6.8.10.3.2 Conversion To String

When a variable is converted to a string, the number is simply expressed as a string. However, leading 0s are stripped from the number when it is first assigned to a variable.

Note: Comparisons of numbers expressed as strings may yield some results that are not obvious.
"5" lt "06" returns false as '0' is alphabetically before '5'
"-2" lt "0" returns true but
"-2" lt "-0" returns false

Example

```
a = 06; // a contains the number 6
```

When the variable a is converted to a string, it is interpreted as "6", not "06"

4.6.8.10.3.3 String Constant Precedence

If one of the expressions is a string constant, the other expression is converted to a string.

4.6.8.10.3.4 Numeric Constant Precedence

If one of the expressions is a numeric constant, the other expression is converted to a numeric value.

4.6.8.10.4 Arrays

See the Array (Object) section for a discussion of SWiSH Max specific items.

4.6.8.10.5 Physics

To simplify gaming and other scripted movement, SWiSH Max supports additional properties that let you define the velocity, acceleration and friction of an Object or Movie Clip.

Some of the standard Movie Clip properties, have physics extensions.

To enable these properties you must turn on physics for the Movie and each individual Object that uses these properties. See Enable Movie Physics and Enable Object Physics for more information.

Example
The property _x has the SWiSH Max physics extensions _ax, _vx, _fx where:

- _ax represents the acceleration in pixels/second/second
- _vx represents the velocity in pixels/second
- _fx represents the movement friction percentage.

The table of properties with physics extensions and the properties representing those extensions is shown below:

Property	**Acceleration Property**	**Friction Property**	**Velocity Property**
_alpha	_aalpha (% / sec / sec)	_falpha (%)	_valpha (% / sec)
_rotation	_arotation (deg / sec / sec)	_frotation (%)	_vrotation (deg / sec)
_x	_ax (pixels / sec / sec)	_fx (%)	_vx (pixels / sec)
_xscale	_axscale (% / sec / sec)	_fxscale (%)	_vxscale (% / sec)
_Y	_aY (pixels / sec / sec)	_fY (%)	_vY (pixels / sec)
_yscale	_ayscale (% / sec / sec)	_fyscale (%)	_vyscale (% / sec)

The additional properties _age, _delta and _time are also provided to assist with age and movement calculations.

4.6.8.10.5.1 Acceleration

Acceleration is the rate at which the velocity changes.

An acceleration of 0 means that the Object travels at a constant velocity (i.e. the velocity does not change).

Example
If an object has an initial velocity of 10 pixels per second and accelerates at 1 pixel per second:

- after 1 second its velocity will be 11 pixels per second
- after 10 seconds it velocity will be 20 pixels per second.

The distance moved can be calculated from the formula D = v * t + a * t * t / 2;
where D is the distance moved, v is the initial velocity and t is the time in seconds.

In the above example, after 10 seconds the object would have moved 10 * 10 + 1 * 10 * 10 / 2 = 150 pixels.

Within <%SWISHSCRIPT%>, the concept of acceleration is also applied to % values. In this case, the acceleration is the rate at which the % velocity changes per second.

4.6.8.10.5.2 Friction

The friction property acts to simulate 'real' friction by reducing the velocity by a specified % per second.

It can be used to simulate things rolling to a stop.

It is different to acceleration, as it works by modifying velocity by a percent instead of a fixed value.

Example
An object has _vX = 100, _fX = 10:

- after 1 second, _vX = 90 (100 - 10%)
- after 2 seconds, _vX = 81 (90 - 10%)
- after 3 seconds, _vX = 72.9 (81 - 10%).

4.6.8.10.5.3 Velocity

Velocity is the rate at which the Object moves.

Example
If an object has a velocity of 10 pixels per second:

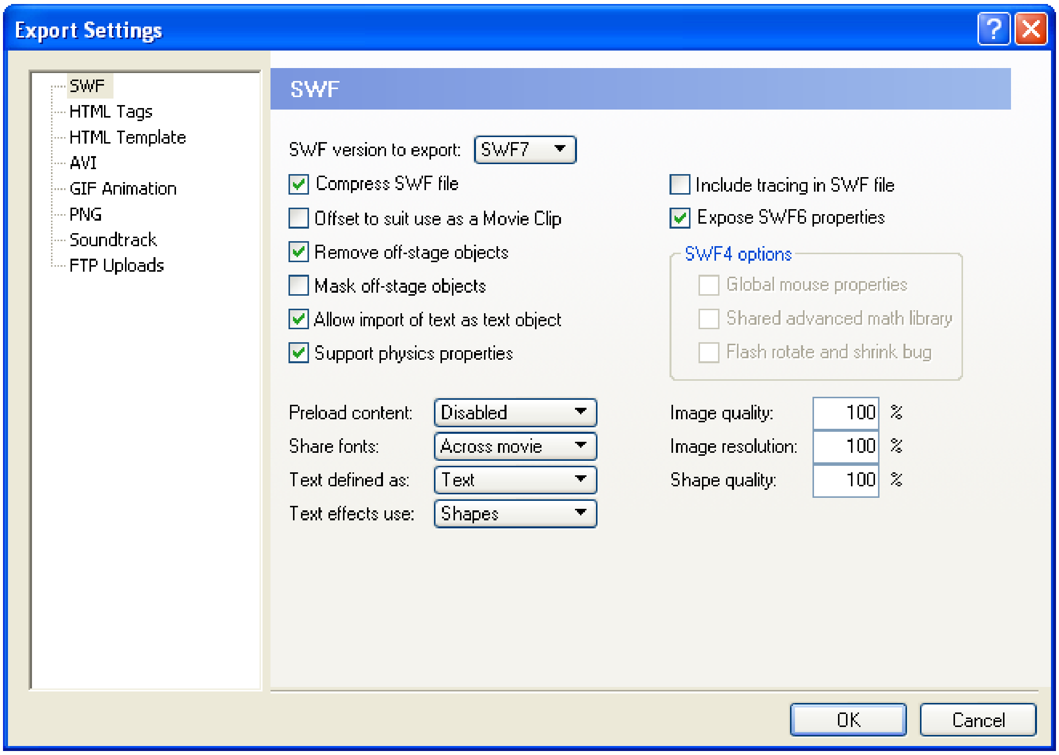
- after 1 second it will have moved 10 pixels
- after 10 seconds it will have moved 100 pixels.

Within <%SWISHSCRIPT%>, the concept of velocity is also applied to % values. In this case, the velocity is the rate at which the % changes per second.

4.6.8.10.5.4 Enable Movie Physics

To enable Movie physics, press the **Movie Properties...** button from the Properties Panel of the current Scene.
Then press the **Export Settings for Movie...** button.

The items below will appear.



SWF

SWF version to export: SWF7

☑ Compress SWF file
☐ Offset to suit use as a Movie Clip
☑ Remove off-stage objects
☐ Mask off-stage objects
☑ Allow import of text as text object
☑ Support physics properties

☐ Include tracing in SWF file
☑ Expose SWF6 properties

SWF4 options
☐ Global mouse properties
☐ Shared advanced math library
☐ Flash rotate and shrink bug

Preload content: Disabled
Share fonts: Across movie
Text defined as: Text
Text effects use: Shapes

Image quality: 100 %
Image resolution: 100 %
Shape quality: 100 %

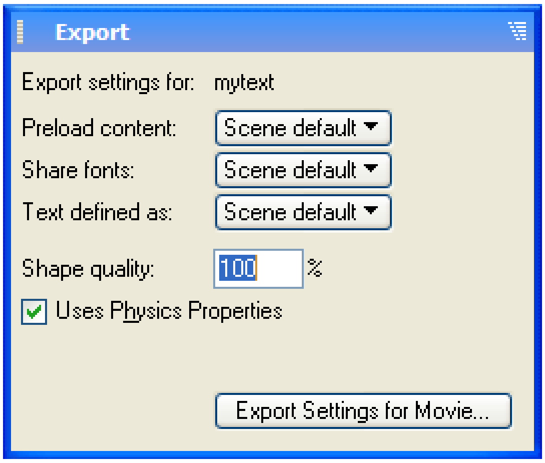
Check the 'Support physics properties' checkbox.

4.6.8.10.5.5 Enable Object Physics

It is necessary to enable the properties for each Object/ Movie Clip / Sprite that uses the physics properties. This is done from the Export Panel.

Note: The default panel layout does not show the Export panel. To see this panel use the menu item Window to enable the panel or use the key combination Ctrl+Shift+Alt+F12

The items below will appear.



Export settings for: mytext
Preload content: Scene default
Share fonts: Scene default
Text defined as: Scene default
Shape quality: 100 %
☑ Uses Physics Properties

Check the 'Uses physics properties' checkbox.

4.6.8.11 Script Reference Dictionary

The Script Reference Dictionary details each of the supported script commands.

The following symbols are used in the contents heading to indicate special notes about the script.

Symbol	Meaning
+	SWiSH Max-specific or special SWiSH Max-specific use
*	Deprecated (old style) function / operator

4.6.8.11.1 - (minus)

Player Required
SWF4 or later

Syntax
(Negation) - expression
(Subtraction) expression1 - expression2

Arguments
None.

Returns
Nothing.

Description
Operator (arithmetic); used for negating or subtracting.
Usage 1: When used for negating, it reverses the sign of the numerical expression.
Usage 2: When used for subtracting, it performs an arithmetic subtraction on two numerical expressions, subtracting expression2 from expression1. When both expressions are integers, the difference is an integer. When either or both expressions are floating-point numbers, the difference is a floating-point number.

Sample
Usage 1: The following statement reverses the sign of the expression 2 + 3.

```
-(2 + 3)
```

The result is -5.

Usage 2: The following statement subtracts the integer 2 from the integer 5.

```
5 - 2
```

The result is 3, which is an integer.

Usage 2: The following statement subtracts the floating-point number 1.5 from the floating-point number 3.25.

```
3.25 - 1.5
```

The result is 1.75, which is a floating-point number.

4.6.8.11.2 -- (decrement)

Player Required
SWF4 or later

Syntax
--expression
expression--

Arguments
expression can be a variable, number, element in an array, or the property of an Object.

Returns

--expression returns expression - 1
expression-- returns original value but the variable is decremented internally.

Description

Operator (arithmetic); a pre-decrement and post-decrement unary operator that subtracts 1 from the expression. The pre-decrement form of the operator (--expression) subtracts 1 from expression and returns the result. The post-decrement form of the operator (expression--) subtracts 1 from the expression and returns the initial value of expression (the value prior to the subtraction).

Sample

The pre-decrement form of the operator decrements x to 2 (x - 1 = 2), and returns the result as y:

```
x = 3;
y = --x;
//both y and x are equal to 2
```

The post-decrement form of the operator decrements x to 2 (x - 1 = 2), and returns the original value of x as the result y:

```
x = 3;
y = x--;
//y is equal to 3, x is equal to 2
```

4.6.8.11.3 ! (logical NOT)

Player Required

SWF4 or later

Syntax

!expression

Arguments

An expression that evaluates to a boolean value, usually a comparison expression, such as x < 5

Returns

Inverted boolean value of expression.

Description

Operator (logical); inverts the boolean value of a variable or expression. If expression is a variable with the absolute or converted value true, the value of !expression is false. If the expression x && y evaluates to false, the expression !(x && y) evaluates to true.
The following expressions illustrate the result of using the ! operator:
!true returns false
!false returns true.

Sample

In the following example, the variable happy is set to false. The if condition evaluates the condition !happy, and if the condition is true, the trace Action sends a string to the 'Debug' window:

```
happy = false;
if (!happy) {
    trace("don't worry, be happy");
}
```

4.6.8.11.4 != (inequality)

Player Required

SWF5 or later

Syntax

expression1 != expression2

Arguments

None.

Returns

false if (expression1 == expression2)

Description

Operator (inequality); tests for the exact opposite of the == operator. If expression1 is equal to expression2, the result is false. As with the == operator, the definition of equal depends on the data types being compared:

The definition of equal depends on the data type of the parameter:

- If both of the two values are strings, they are compared as strings. Otherwise, any strings are converted to numbers before being compared.
- If the values are numbers, then the values are compared

Note: For SWF4 only, comparing a variable and a string will assume the variable also contains a string, and will do a string comparison. For SWF5+ the actual type of value in the variable is used as above to determine how to compare.

Sample

The following example illustrates the result of the != operator:
5 != 8 returns true
5 != 5 returns false

This example illustrates the use of the != operator in an if statement:

```
a = "David";
b = "Fool" ;
if (a != b){
    trace("David is not a fool");
}
```

4.6.8.11.5 % (mod)

Player Required

SWF4 or later

Syntax

expression1 % expression2

Arguments

expression1, expression2 are floating-point or integer numbers. expression2 cannot be 0.

Returns

The integer remainder of expression1 / expression 2.

Description

x%y is emulated by the calculation x - (int(x/y) * y).

Sample

```
7 % 3 // returns 1
2.4 % 1.1 // returns 0.2
```

4.6.8.11.6 %= (mod assignment)

Player Required
SWF4 or later

Syntax
expression1 %= expression2

Arguments
None.

Returns
None.

Description
Identical to expression1 = expression1 % expression2.

Sample
```
Z = 7;
Z %= 3;
// Z now has the value of 1 (7%3)
```

4.6.8.11.7 & (bitwise AND)

Player Required
SWF5 or later

Syntax
expression1 & expression2

Arguments
None.

Returns
Nothing.

Description
Bitwise Operator; converts both expression1 and expression2 into 32-bit unsigned integers then a Boolean AND operation is performed for each bit of the integer parameters. A new 32-bit unsigned integer is the result.

4.6.8.11.8 &= (bitwise AND assignment)

Player Required
SWF5 or later

Syntax
expression1 &= expression2

Arguments
None.

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 & expression2 to expression1.

4.6.8.11.9 ~ (bitwise NOT)

Player Required
SWF5 or later

Syntax
~ expression

Arguments
None.

Returns
Nothing.

Description
Bitwise Operator; converts the expression to a 32-bit unsigned integer then inverts the bits. It also changes the sign of a number and subtracts 1 from it.

4.6.8.11.10 && (short-circuit AND)

Player Required
SWF4 or later

Syntax
expression1 && expression2

Arguments
None.

Returns
Boolean result of expression1 AND expression2.

Description
Operator (logical); performs a boolean operation on the values of one or both of the expressions. Evaluates expression1 (the expression on the left side of the operator) and returns false if the expression evaluates to false. If expression1 evaluates to true, expression2 (the expression on the right side of the operator) is evaluated. If expression2 evaluates to true, the final result is true; otherwise, it is false.

Sample
This example uses the && operator to perform a test to determine if a player has won the game. The turns variable and the score variable are updated when a player takes a turn or scores points during the game. The following script displays "You Win the Game!" in the 'Debug' window when the player's score reaches 75 or higher in 3 turns or less.

```
turns = 2;
score = 77;
winner = (turns <= 3) && (score >= 75);
if (winner) {
    trace("You Win the Game!");
} else {
    trace("Try Again!");
}
```

4.6.8.11.11 () (parentheses)

Player Required
SWF4 or later

Syntax
(expression1, expression2)

and
function(argument1, ..., argument*N*)

Arguments
None.

Returns
None.

Description
Forces a precedence of evaluation and groups arguments and expressions together.

Sample

```
(x + 3) * 5 // this changes standard precedence so that addition is done before multiplication.

call MyFunction(a,b,c) // groups arguments to be passed to the function MyFunction().
```

4.6.8.11.12 * (multiplication)

Player Required
SWF4 or later

Syntax
expression1 * expression2

Arguments
None.

Returns
result of expression1 * expression2

Description
Operator (arithmetic); multiplies two numerical expressions. If both expressions are integers, the product is an integer. If either or both expressions are floating-point numbers, the product is a floating-point number.

Sample
The following statement multiplies the integers 4 and 3:

```
4 * 3
```

The result is 12, which is an integer.

Sample
This statement multiplies the floating-point numbers 2.0 and 3.1416.

```
2.0 * 3.1416
```

The result is 6.2832, which is a floating-point number.

4.6.8.11.13 *= (multiplication assignment)

Player Required
SWF4 or later

Syntax
expression1 *= expression2

Arguments
None.

Returns
None.

Description
Identical to expression1 = expression1 * expression2.

Sample
```
Z = 7;
Z *= 3;
// Z now has the value of 21 (7 * 3)
```

4.6.8.11.14 , (comma)

Player Required
SWF4 or later

Syntax
expression1, expression2

Arguments
expression1, expression2 are any data (numbers, variables, strings, etc).

Returns
None.

Description
Tells the Flash Player to evaluate expression1 then expression2, etc.

Sample
```
call Myfunction(a+3, 4 * 2); // the values a+3 and 8 are passed to the function.
```

4.6.8.11.15 . (dot operator)

Player Required
SWF4 or later

Syntax
object.property
object.method
object.childObject
instance.variable
instance.child.variable

Arguments
object: Any instance of an Object.
property: Any property associated with an Object.
method: Any method associated with the Object.
instance: Any Instance of a Movie Clip (Sprite).
child: Any Instance of a Movie Clip that is a part of another Movie Clip (Sprite).
variable: Any variable in a Movie Clip.

Returns
Value of property or result of method operation or reference to child Object.

Description
Used to denote the hierarchical relationship among Objects, variables, methods, etc.

Sample
```
_root.Rect._X
_root.myMovieClip.play()
```

4.6.8.11.16 / (division)

Player Required
SWF4 or later

Syntax
expression1 / expression2

Arguments
expression: A number or a variable that evaluates to a number.

Returns
Floating-point result of expression1 / expression2.

Description
Operator (arithmetic); divides expression1 by expression2. The result of the division operation is a double-precision floating-point number.

Sample
The following statement divides the floating-point number 22.0 by 7.0 and then displays the result in the 'Debug' window.

```
trace(22.0 / 7.0);
```

The result is 3.1429, which is a floating-point number.

4.6.8.11.17 /* */ or // (comment delimiter)

Player Required
SWF4 or later

Syntax
// comment

or

/* multi
line
comment */

Arguments
Comment.

Returns
Nothing.

Description
Allows addition of comments to code. Comments can be used to explain complex areas of code or unusual coding practices.

Use of comments is highly recommended and will simplify future maintenance of the script.

Note: Comments do not affect the size of the exported .swf file. For this reason, they should **not** be removed in an attempt to improve the performance of your movie.

Sample

```
a += 3; // single line comment, add 3 to a
```

/* multi line
comment

this next section is complex...*/
a -=1;

4.6.8.11.18 /= (division assignment)

Player Required
SWF4 or later

Syntax
expression1 /= expression2

Arguments
None.

Returns
None.

Description
Identical to expression1 = expression1 / expression2.

Sample
```
a = 8;
a /= 4;
// a now has the value of 2, (8/4)
```

4.6.8.11.19 ?: (conditional)

Player Required
SWF4 or later

Syntax
expression1 ? expression2 : expression3

Arguments
expression1: An expression that evaluates to a boolean value, usually a comparison expression, such as x < 5.
expression2, expression3: Values of any type.

Returns
expression2 if expression1 is true, expression3 if expression1 is false.

Description
Operator; instructs the Flash Player to evaluate expression1, and if the value of expression1 is true, it returns the value of expression2; otherwise it returns the value of expression3.

Sample
The following statement assigns the value of variable x to variable z because expression1 evaluates to true:
```
x = 5;
y = 10;
z = (x < 6) ? x: y; // z assigned the value of x as (x<6) is true.
trace (z);
// returns 5
```

4.6.8.11.20 [] (array access operator)

Player Required
SWF4 or later

Syntax
myArray[n]

Arguments
Index of the array element to return.

Returns
Returns / allows access to the nth element of an array.

Description
Used to access the nth element of an array.
Note that all elements of the array should be initialised before access.

Sample
```
myArray = new Array;
myArray[0] = "first";
myArray[1] = "second";
myArray[2] = "third";
n = 1;
b = myArray[n];  // b now contains the value "second"
```

4.6.8.11.21 | (bitwise OR)

Player Required
SWF5 or later

Syntax
expression1 | *expression2*

Arguments
expression1 and *expression 2*: numbers

Returns
Nothing.

Description
Bitwise Operator; converts both expressions to 32-bit unsigned integers and returns a 1 in each bit position where the corresponding bits of either expression is equal to 1.

Sample
```
a=5;    // 101 binary
b=3;    // 011 binary
c=(a|b);       // 111 binary (7)
```

4.6.8.11.22 || (logical OR)

Player Required
SWF4 or later

Syntax
expression1 || expression2

Arguments
expression1,expression2: A boolean value or an expression that converts to a Boolean value.

Returns
Logical result of expression1 OR expression2.

Description
Operator (logical); evaluates expression1 and expression2. The result is true if either or both expressions evaluate to true; the result is false only if both expressions evaluate to false. You can use the logical OR operator with any number of operands; if any operand evaluates to true, the result is true.

With non-boolean expressions, the logical OR operator causes the Flash Player to evaluate the expression on the left; if it can be converted to true, the result is true. Otherwise, it evaluates the expression on the right and the result is the value of that expression.

Sample
The following example uses the || operator in an if statement. The second expression evaluates to true, so the final result is true:

```
x = 10
y = 250
start = false
if(x > 25 || y > 200 || start){
    trace('the logical OR test passed');
}
```

Sample
This example demonstrates how a non-boolean expression can produce an unexpected result. If the expression on the left converts to true, that result is returned without converting the expression on the right.

```
function fx1(){
    trace ("fx1 called");
    returns true;
}
function fx2(){
    trace ("fx2 called");
    return true;
}
if (fx1() || fx2()){
    trace ("IF statement entered");
}
// The following is sent to the 'Debug' window:
// fx1 called
// IF statement entered
```

4.6.8.11.23 |= (bitwise OR assignment)

Player Required
SWF5 or later

Syntax
expression1 |= expression2

Arguments
expression1 and *expression* 2: numbers or variables

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 | expression2 to expression1.

Sample

```
a=5;    // 101 binary
b=3;    // 011 binary
a|=b;   // a now equals 111 binary (7)
```

4.6.8.11.24 ^ (bitwise XOR)

Player Required
SWF5 or later

Syntax
expression1 ^ *expression2*

Arguments
expression1 and *expression 2*: numbers

Returns
Nothing.

Description
Bitwise Operator; converts both expressions to 32-bit unsigned integers and returns a 1 in each bit position where the corresponding bits of either expression - but not both of them - is equal to 1.

Sample

```
a=5;    // 101 binary
b=3;    // 011 binary
c=(a^b);        // c is equal to 110 binary (6 decimal)
```

4.6.8.11.25 ^= (bitwise XOR assignment)

Player Required
SWF5 or later

Syntax
expression1 ^= expression2

Arguments
expression1 and *expression 2*: numbers or variables

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 ^ expression2 to expression1.

Sample

```
a=5;    // 101 binary
b=3;    // 011 binary
a^=b;   // a is equal to 110 binary (6 decimal)
```

4.6.8.11.26 + (addition)

Player Required
SWF4 or later

Syntax
expression1 + expression2

Arguments
expression1,expression2 numeric constants, string constants or variables.

Returns
Result of the sum of expression1 + expression2 if neither expressions is a string (for SWF4 only, variables are assumed to be numeric).
Result of expression1 add expression2 if either expression is a string.

Description

If both expressions are integers, the sum is an integer. If either or both expressions are floating-point numbers, the sum is a floating-point number. If either expression is a string constant, the result is expression1 add expression2.

Sample

This statement below adds the integers 2 and 3 and displays the resulting integer, 5, in the 'Debug' window:

```
trace (2 + 3);
```

This statement below adds the floating-point numbers 2.5 and 3.25 and displays the result, 5.75, a floating-point number, in the 'Debug' window:

```
trace (2.5 + 3.25);
```

This statement below writes the string "1122" to the 'Debug' window"

```
trace(11 + "22"); // converts both to string because of "22"

onSelfEvent (load) {
    a = "10";
    b = "eats dog";
    c = "cat ";
    trace(1 + a);
    trace(a + b);
    trace("cat" + " eats dog");
    trace(a add b);
    trace(c add b);
}

Produces the output
11
10
cat eats dog
10eats dog
cat eats dog
```

See Also

plus, Variables and String Conversion for more information about automatic type conversion of variables.

4.6.8.11.27 ++ (increment)

Player Required

SWF4 or later

Syntax

++expression
expression++

Arguments

expression can be a variable, number, element in an array, or the property of an Object.

Returns

++expression returns expression + 1.
expression++ returns expression.

Description

Operator (arithmetic); a pre-increment and post-increment unary operator that adds 1 to expression. The expression can be a variable, element in an array, or property of an Object. The pre-increment form of the operator (++expression) adds 1 to expression and returns the result. The post-increment form of the operator (expression++) adds 1 to expression and returns the initial value of expression (the value prior to the addition).
The pre-increment form of the operator increments x to 2 (x + 1 = 2), and returns the result as y:

```
x = 1;
y = ++x
//y is equal to 2, x is equal to 2
```

The post-increment form of the operator increments x to 2 (x + 1 = 2), and returns the original value of x as the result y:

```
x = 1;
y = x++;
//y is equal to 1, x is equal to 2
```

Sample

The following example uses ++ as a post-increment operator to make a while loop run five times.

```
i = 0;
while(i++ < 5){
trace("this is execution " + i);
}
```

This example uses ++ as a pre-increment operator.

```
var a = [];
var i = 0;
while (i < 10) {
    a.push(++i);
}
trace(a.join()); // displays "1,2,3,4,5,6,7,8,9,10" in the 'Debug' window
```

The following example uses ++ as a post-increment operator.

```
var a = [];
var i = 0;
while (i < 10) {
a.push(i++);
          }
trace(a.join()); // displays "0,1,2,3,4,5,6,7,8,9" in the 'Debug' window.
```

4.6.8.11.28 += (addition assignment)

Player Required

SWF4 or later

Syntax

expression1 += expression2

Arguments

None.

Returns

None.

Description

Identical to expression1 = expression1 + expression2

Sample

```
a = 7;
a += 3;
// a now has the value of 10 (7 + 3)
```

4.6.8.11.29 < (less than)

Player Required

SWF4 or later

Syntax

expression1 < expression2

Arguments

expression1,expression2 can be numbers, strings or variables.

Returns

None.

Description

Operator (comparison): compares two expressions and determines whether expression1 is less than expression2; if so, the operator returns true. If expression1 is greater than or equal to expression2, the operator returns false.

String constants are converted to numeric unless both expressions are string constants.
Variables are assumed to be numeric. Possibly undergoing a conversion.

Mixed numeric / string comparisons can give unusual results.
If string comparison is intended, use the lt operator.

Sample

The following examples illustrate true and false returns for both numeric and string comparisons.

```
3 < 10;
// true

10 < 3;
// false

"a" < "cat";
// true

a = "a";
b = "b";
b < a;
// true.
```

Possibly not what is expected but caused by conversion of strings to numeric 0. Use the lt operator instead.

See Also

lt

4.6.8.11.30 <= (less than or equal to)

Player Required

SWF4 or later

Syntax

expression1 <= expression2

Arguments

expression1,expression2 can be numbers or strings.

Returns

Nothing.

Description

Operator (comparison): compares two expressions and determines whether expression1 is less than or equal to expression2; if so, the operator returns true. If expression1 is greater than expression2, the operator returns false.

String constants are converted to numeric unless both expressions are string constants.
Variables are assumed to be numeric. Possibly undergoing a conversion.

If string comparison is intended, use the le operator.

The definition of equal depends on the data type of the parameter:

- If both of the two values are strings, they are compared as strings. Otherwise, any strings are converted to numbers before being compared.
- If the values are numbers, then the values are compared

Note: For SWF4 only, comparing a variable and a string will assume the variable also contains a string, and will do a string comparison. For SWF5+ the actual type of value in the variable is used as above to determine how to compare.

Sample

The following examples illustrate true and false results for both numeric and string comparisons:

```
5 <= 10;
// true

2 <= 2;
// true

10 <= 3;
// false

"Allen" <= "Jack";
// true

a = "a";
b = "b";
b <= a;
```

`// true.` Possibly not what is exptected but caused by conversion of strings to numeric 0. Use the le operator instead.

See Also

le

4.6.8.11.31 << (bitwise left shift)

Player Required

SWF5 or later

Syntax

expression1 << *expression2*

Arguments

expression1: a number or expression for the left shift to be applied to
expression2: a number or expression that converts to an integer from 0 to 31

Returns

Nothing.

Description

Bitwise Operator; converts both expressions to 32-bit integers and shifts all of the bits in expression1 by the number of places specified by expression2. The bit positions that are emptied by the left shift are filled with a zero. Each value that is shifted by one position to the left is the same as multiplying it by 2.

Sample

```
a=5;    // 101 binary
b=3;    // 011 binary
c=(a<<b);      // c is equal to 101000 binary (40 decimal)
```

4.6.8.11.32 <<= (bitwise left shift and assignment)

Player Required

SWF5 or later

Syntax

expression1 <<= expression2

Arguments

expression1 and *expression* 2: numbers or variables

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 << expression2 to expression1.

Sample
```
a=5;    // 101 binary
b=3;    // 011 binary
a<<=b;  // a is equal to 101000 binary (40 decimal)
```

4.6.8.11.33 <> (inequality *)

Player Required
SWF2 or later

Syntax
a <> b

Description
This is old version of inequality syntax. Although recognized this should be replaced with != syntax.

See != for more information.

4.6.8.11.34 = (assignment)

Player Required
SWF4 or later

Syntax
expression1 = expression2

Arguments
expression1: A variable, element of an array, or property of an Object.
expression2: A value of any type.

Returns
Nothing.

Description
Assigns the value of expression2 to the item defined in expression1.

Sample
```
x=10;          // assigned the number 10 to the variable x
x="hello";     // assigns the string "hello" to the variable x
```

4.6.8.11.35 -= (negation assignment)

Player Required
SWF4 or later

Syntax
expression1 -= expression2

Description
Identical to expression1 = expression1 - expression2.

Sample
```
a = 7;
a -= 3;
```

```
// a now has the value of 4 (7 - 3)
```

4.6.8.11.36 == (equality)

Player Required
SWF5 or later

Syntax
expression1 == expression2

Arguments
expression1,expression2: A number, string, boolean value, variable or function.

Returns
Nothing.

Description
Operator (equality); tests two expressions for equality. The result is true if the expressions are equal.

The definition of equal depends on the data type of the parameter:

- If both of the two values are strings, they are compared as strings. Otherwise, any strings are converted to numbers before being compared.
- If the values are numbers, then the values are compared

Note: For SWF4 only, comparing a variable and a string will assume the variable also contains a string, and will do a string comparison. For SWF5+ the actual type of value in the variable is used as above to determine how to compare.

Sample

```
x = 5;
y = 6;
trace(x == y); // returns (0), false
```

These examples show the results of operations that compare mixed types.

```
x = "5"; y = "5";
trace(x == y); // this works as variables are converted to numbers.
// true
// this works as variables are converted to numbers.

x = "5"; y = "66";
trace(x ==y);
// false
// this works as variables are converted to numbers.

x = "chris"; y = "steve";
trace (x == y);
// true
// unexpected result caused by conversion of both strings constants to numeric (value = 0).
// use 'eq' operator instead.
```

See Also
eq

4.6.8.11.37 > (greater than)

Player Required
SWF4 or later

Syntax
expression1 > expression2

Arguments
expression1,expression2 can be numbers, strings or variables.

Returns
Nothing.

Description
Operator (comparison): compares two expressions and determines whether expression1 is greater than expression2; if so, the operator returns true. If expression1 is less than or equal to expression2, the operator returns false.

String constants are converted to numeric unless both expressions are string constants.
Variables are assumed to be numeric. Possibly undergoing a conversion.

Mixed numeric / string comparisons can give unusual results.
If string comparison is intended, use the gt operator.

Sample
The following examples illustrate true and false returns for both numeric and string comparisons.

```
31 > 10;
// true

10 > 31;
// false

"dog" > "cat";
// true, both string constants so comparison based on string comparison.

c = "cat";
d = "dog";
d > c;
// false. Probably not what was intended as both variables are converted to numeric 0. Use the gt
operator.
```

See Also
gt

4.6.8.11.38 >= (greater than or equal to)

Player Required
SWF4 or later

Syntax
expression1 >= expression2

Arguments
expression1, expression2 can be numbers, or strings.

Returns
Nothing.

Description
Operator (comparison): compares two expressions and determines whether expression1 is greater than or equal to expression2; if so, the operator returns true. If expression1 is less than expression2, the operator returns false.

String constants are converted to numeric unless both expressions are string constants.
Variables are assumed to be numeric. Possibly undergoing a conversion.

If string comparison is intended, use the ge operator.

The definition of equal depends on the data type of the parameter:

- If both of the two values are strings, they are compared as strings. Otherwise, any strings are converted to numbers before being compared.
- If the values are numbers, then the values are compared

Note: For SWF4 only, comparing a variable and a string will assume the variable also contains a string, and will do a string comparison. For SWF5+ the actual type of value in the variable is used as above to determine how to compare.

Sample

The following examples illustrate true and false returns for both numeric and string comparisons.

```
31 >= 31;
// true

10 >= 31;
// false

"dog" >= "cat";
// true

d = "dog";
c = "cat";

c >= d;
// true, possibly not what is intended as both variables are converted to numeric (0). Use ge
instead.
```

See Also

ge

4.6.8.11.39 >> (bitwise right shift)

Player Required

SWF5 or later

Syntax

expression1 >> *expression2*

Arguments

expression1: a number or expression for the right shift to be applied to
expression2: a number or expression that converts to an integer from 0 to 31

Returns

Nothing.

Description

Bitwise Operator; converts both expressions to 32-bit integers and shifts all of the bits in expression1 by the number of places specified by expression2. The bits shifted to the right are discarded. In order to maintain the sign of the original expression, the bits are filled with 0 or 1 based on the bit farthest to the left. Each value that is shifted by one position to the left is the same as dividing it by 2 and removing the remainder.

Sample

```
a=10;   // 1010 binary
b=2;
c=(a>>b);      // c is equal to 10 binary (2 decimal)
```

4.6.8.11.40 >>= (bitwise right shift and assignment)

Player Required

SWF5 or later

Syntax
expression1 >>= expression2

Arguments
expression1 and *expression 2*: numbers or variables

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 >> expression2 to expression1.

Sample

```
a=10;   // 1010 binary
b=2;
a>>=b); // a is equal to 10 binary (2 decimal)
```

4.6.8.11.41 >>> (bitwise unsigned right shift)

Player Required
SWF5 or later

Syntax
expression1 >>> *expression2*

Arguments
expression1: a number or expression for the right shift to be applied to
expression2: a number or expression that converts to an integer from 0 to 31

Returns
Nothing.

Description
Bitwise Operator; uses the same premise as the >> (bitwise right shift) operator with the exception that the sign of the original expression is not preserved, because all of the bits on the left are automatically filled with 0.

4.6.8.11.42 >>>= (bitwise unassigned right shift and assignment)

Player Required
SWF5 or later

Syntax
expression1 >>>= expression2

Arguments
expression1 and *expression 2*: numbers or variables

Returns
Nothing.

Description
Bitwise Operator (compound assignment); assigns the value of expression1 >>> expression2 to expression1.

4.6.8.11.43 and (logical AND *)

Player Required
SWF4 or later

Syntax

a and b

Description
This is old version of logical AND syntax. Although recognized this should be replaced with && syntax.

4.6.8.11.44 eq (string ==)

Player Required
SWF4 or later

Syntax
a eq b

Description
Similar to ==, however it ensures that string comparison is always used, even if either value is a number.

Sample
```
a = "hi";
b = "there";
trace(a == b);   // for SWF4, returns 1 (true) as both a and b are converted to numeric.
trace(a eq b);   // returns 0 (false) as strings are different which is the expected result.
```

See Also
==

4.6.8.11.45 ge (string >=)

Player Required
SWF4 or later

Syntax
a ge b

Description
Similar to >=, however it ensures that string comparison is always used, even if either value is a number.

Sample
```
a = "hi";
b = "there";
trace(a >= b); // For SWF4, returns 1 (true) as both a and b are converted to numeric and equate to
equal (both 0).
trace(a ge b); // returns 0 (false) as "hi" is before "there", which is the expected result.
```

See Also
>=

4.6.8.11.46 gt (string >)

Player Required
SWF4 or later

Syntax
a gt b

Description
This is old version of greater than syntax. Although recognized this should be replaced with > syntax.

Similar to >, however it ensures that string comparison is always used, even if either value is a number.

Sample
```
a = "hi";
b = "there";
```

```
trace(a > b);  // For SWF4, returns 0 (false) as both a and b are converted to numeric and equate to
equal (both 0).
trace(a ge b); // returns 0 (false) as "hi" is before "there", which is the expected result.
```

See Also

>

4.6.8.11.47 le (string <=)

Player Required

SWF4 or later

Syntax

a le b

Description

Similar to <=, however it ensures that string comparison is always used, even if either value is a number.

Sample

```
a = "zhi";
b = "there";
trace(a <= b); // For SWF4, returns 1 (true) as both a and b are converted to numeric and equate to
equal (both 0).
trace(a le b); // returns 0 (false) as "zhi" is after "there", which is the expected result.
```

See Also

<=

4.6.8.11.48 lt (string <)

Player Required

SWF4 or later

Syntax

a lt b

Description

This is old version of less than syntax. Although recognized this should be replaced with < syntax.

Similar to <, however it ensures that string comparison is always used, even if either value is a number.

Sample

```
a = "hi";
b = "there";
trace(a < b);  // For SWF4, returns 0 (false) as both a and b are converted to numeric and equate to
equal (both 0).
trace(a le b); // returns 1 (true) as "hi" is before "there", which is the expected result.
```

See Also

<

4.6.8.11.49 ne (string !=)

Player Required

SWF4 or later

Syntax

a ne b

Description
Similar to !=, however it ensures that string comparison is always used, even if either value is a number.

Sample

```
a = "hi";
b = "there";
trace(a != b); // For SWF4, returns 0 (false) as both a and b are converted to numeric and equate to
0
trace(a ne b); // returns 1 (true) as strings are different, which is the expected result.
```

See Also
!=

4.6.8.11.50 not (logical NOT *)

Player Required
SWF4 or later

Syntax
not a

Description
This is old version of logical NOT syntax. Although recognized this should be replaced with ! syntax.

4.6.8.11.51 or (logical OR *)

Player Required
SWF4 or later

Syntax
a or b

Description
This is old version of logical OR syntax. Although recognized this should be replaced with || syntax.

4.6.8.11.52 _aalpha +

See Physics Properties.

4.6.8.11.53 _age +

Player Required
SWF4 or later

Syntax
instanceName._age

Arguments
None.

Returns
Age of Object since placed (in seconds).

Description
Read-only property. Returns the age of the Object in seconds.

This is a SWiSH Max-specific Physics Property.

4.6.8.11.54 _alpha

Player Required
SWF4 or later

Syntax
instanceName._alpha
instanceName._alpha = value

Arguments
Can be set to a number between 0 and 100.

Returns
Current setting.

Description
The property both reads and sets a Movie Clip / Sprite's or Object's alpha transparency.
A value of 0 is transparent.
A value of 100 is opaque.

Elements with an alpha of 0 (i.e. invisible) can still detect Events.

Sample
```
onSelfEvent (load) {
    _root.rectangle._alpha = 40;      // make rectangle partly transparent
}
```

Physics Properties are also supported for this property.

4.6.8.11.55 _arotation +

See Physics Properties.

4.6.8.11.56 _ax +

See Physics Properties.

4.6.8.11.57 _axscale +

See Physics Properties.

4.6.8.11.58 _ay +

See Physics Properties.

4.6.8.11.59 _ayscale +

See Physics Properties.

4.6.8.11.60 _button +

Player Required
No specific player. SWiSH Max extension to allow access to button properties and methods.

Syntax
buttonscriptobject._button

Arguments
buttonscriptobject is a button Script Object (Target checkbox is ticked).

Returns
Values associated with the button property or method that is accessed.

Description
SWiSH Max button, shape and text objects are automatically wrapped in a Movie Clip when the object is

named and the target tickbox is checked. If access to the original button properties and methods are required, these must be accessed via the _button property.

Example:
The movie contains a button b1 that has the target checkbox ticked.
Within the button b1, the following script within the onSelfEvent (load) function defines the event handling function for onPress.

```
onSelfEvent (load)
{
    this._button.onPress = function () {
        trace("press");
    }
}
```

This event function will write "press" to the debug window each time the button is pressed.

Alternatively the script could be placed in the onSelfEvent (load) function of Scene_1 as follows:

```
onSelfEvent (load)
{
    b1._button.onPress = function () {
        trace("press");
    }
}
```

If the target checkbox was not ticked (the button is no longer a scripting object), then the following code placed within the onSelfEvent (load) function for the scene would do the same thing.

```
onSelfEvent (load)
{
    b1.onPress = function () {
        trace("press");
    }
}
```

Note that this function cannot be placed within the button b1 as the button is no longer a scripting object (target checkbox is not ticked).

4.6.8.11.61 _currentframe

Player Required
SWF4 or later

Syntax
instanceName._currentframe

Arguments
None.

Returns
The current Frame number of the named Instance / Movie Clip / Sprite.

Description
Read-only property returns the current Frame number of the named Instance / Movie Clip / Sprite.

Sample
```
if (_root._currentframe > 10) {
    trace("passed frame 10");
}
```

4.6.8.11.62 _delta +

Player Required
SWF4 or later

Syntax
instanceName._delta

Arguments
None.

Returns
Time elapsed since last Frame displayed (in seconds).

Description
Read-only property. Returns the time since the last Frame was displayed in seconds.

This is a SWiSH Max-specific Physics Property.

4.6.8.11.63 _droptarget

Player Required
SWF4 or later

Syntax
draggableInstance._droptarget

Arguments
draggableInstance is the name of the draggable Instance Movie Clip / Sprite or Object.

Returns
Name of the Movie Clip / Sprite that draggableInstance has been dragged over.
The return value is in "slash" notation.

Note: The property will only change in the Object that is being dragged

Description
This is a read-only property.

Sample
Assume a Movie contains two Movie Clip / Sprites, s1 and s2.
code below is for Movie Clip / Sprite s1...
no script is required for s2.
"zap" is displayed in the 'Debug' Panel if s1 is dragged over s2.

Note:

- If s2 was dragged over s1, s1._droptarget (this._droptarget) would not change, but s2._droptarget would show "/s1"
- To enter the if statement, use the if (_droptarget) option

```
onEnterFrame() {
    if (this._droptarget ==_parent.s2._target) {
        trace("zap");
    }
}

onSelfEvent (release) {
    stopDrag();
}

onSelfEvent (press) {
    this.startDragLocked();
}
```

See Also
_target

4.6.8.11.64 _falpha +

See Physics Properties.

4.6.8.11.65 _focusrect

Player Required
SWF6 or later

Syntax
_focusrect = Boolean;

Arguments
None.

Returns
None.

Description
Global property that defines whether a yellow rectangle appears around the button that has keyboard focus. Default value is true.

Sample

```
onSelfEvent (load) {
    _focusrect = false;       // turn off yellow rectangle around selected button.
}
```

4.6.8.11.66 _framesloaded

Player Required
SWF4 or later

Syntax
a = MovieClipName._framesloaded

Arguments
MovieClipName is the name of a Movie Clip or Sprite.

Returns
The number of Frames that have currently been loaded.

Description
Property (read-only); the number of Frames currently loaded from a streaming Movie.

This can be used to check if the contents of a specific Frame has been loaded or for checking the download progress of large Movies.

This property could be used to create a temperature bar that shows the % complete of download.

The following example uses the _totalframes property to calculate the % loaded

Sample

```
a = _framesloaded / _totalframes * 100;      // a contains % complete of download.
```

See Also

_totalframes

4.6.8.11.67 _frotation +

See Physics Properties.

4.6.8.11.68 _fx +

See Physics Properties.

4.6.8.11.69 _fxscale +

See Physics Properties.

4.6.8.11.70 _fy +

See Physics Properties.

4.6.8.11.71 _fyscale +

See Physics Properties.

4.6.8.11.72 _global

Player Required
SWF6 or later

Syntax
_global.*identifier*

Arguments
None

Returns
A reference to the global object that holds the core ActionScript classes.

Description
Objects, variables and classes defined and referenced in this way are available within the entire scope of the SWF file.
Note: The fully qualified variable name, _global.identifier must be used in all references to the global variable. Failure to do so will create a local variable that will obscure the global variable. ie. _global.i and i refer to different variables.

Sample
```
onSelfEvent (load)
{
    _global.i = 5;     // _global.i is available to all timelines and objects within the .swf file.
}
onSelfEvent (enterFrame)
{
    trace(_global.i++);
}
```

4.6.8.11.73 _height

Player Required
SWF4 or later

Syntax
MovieClipName._height

Arguments
MovieClipName is the name of a Movie Clip / Sprite or Object.

Returns
This read-only property returns the current height of the Movie Clip / Sprite or Object.

Description
This can be used to determine the current height of the referenced Object.
If scaling is applied (_yscale), this property is modified to show the revised size.

Sample
```
h = rect._height;      // h contains the current height of the Object rect.
```

4.6.8.11.74 _highquality

Player Required
SWF4 or later

Syntax
_highquality
_highquality = value

Arguments
value: Numeric, new value of global quality setting.

Returns
Current quality setting.

Description
Property (global); specifies the level of anti-aliasing applied to the current Movie. Specify 2 (BEST) to apply high quality with bitmap smoothing always on. Specify 1 (high quality) to apply anti-aliasing; this will smooth bitmaps if the Movie does not contain animation. Specify 0 (low quality) to prevent anti-aliasing.

Sample
```
_highquality = 1;      // apply anti-aliasing
```

See also
toggleHighQuality

4.6.8.11.75 _level

Player Required
SWF4 or later

Syntax
_level*N*

Arguments
N is any non-negative integer.

Returns
None.

Description
Synonym for the Movie loaded into level N.
The root Movie is loaded onto level 0.

Sample
```
_level2.stop() // will stop the Movie playing on level 2.
```
_level0 can be used in place of _root (for clarity, this is not recommended).

4.6.8.11.76 _lockroot

Player Required
SWF7 or later

Syntax
_lockroot = true;

Arguments
None

Returns
None.

Description
If set to true, then _root refers to the current movie clip even if the movie is loaded into another movie clip. If this property is not set or set to false then _root will refer to the _root of the loading movie.
This property allows movies that make reference to _root to function correctly when loaded into a movie clip.

Sample
```
Menu.swf loads page1.swf into a movie clip mc.
If _root references within page1.swf are intended to refer to the _root level of page1.swf then the
top level movie clip of page1.swf should contain the statement this._lockroot = true;
```

4.6.8.11.77 _name

Player Required
SWF4 or later

Syntax
mc._name
mc._name = value

Arguments
mc: The name of the Movie Clip / Sprite, Instance or Object.
value: A string containing the new name of the Movie Clip / Sprite or Object.

Returns
Current name of the Movie Clip / Sprite or Object.

Description
This property allows the Movie Clip / Sprite, Instance or Object to be read and modified.

Sample
```
onSelfEvent (load) {
    trace(this._name);  // show name of this Object / Sprite
}
```

See Also
_target

4.6.8.11.78 _parent

Player Required
SWF4 or later

Syntax
_parent.property
_parent.property = value

_parent._parent.property is also valid.

Arguments
property: Any property of the parent Object.
value: New value of the property.

Description
Synonym for the Movie Clip / Sprite / Object that contains this Object, i.e. the parent.

Sample

```
_parent.play();        // restart parent Movie Clip
```

See Also
_root, this

4.6.8.11.79 _quality

Player Required
SWF5 or later

Syntax
_quality

Arguments
property: Any property of the parent Object.
value: New value of the property.

Description
Property (global); used to set or retrieve the export (render) quality for the movie.

Note: Device fonts are unaffected by the _quality property.

The following values are acceptable for this property:

- LOW - Bitmap images are not smoothed and graphics are not anti-aliased.
- MEDIUM - Bitmap images are not smoothed and graphics are anti-aliased using only a 2 x 2 pixel grid.
- HIGH - Bitmap images are smoothed if the movie is static and graphics are anti-aliased using a 4 x 4 pixel grid (this is the default export setting)
- BEST - Bitmap images are always smoothed and graphics are anti-aliased using a 4 x 4 pixel grid.

4.6.8.11.80 _root

Player Required
SWF4 or later

Syntax
_root.property
_root.property = value

Arguments
property: Any property of the root (top level) Object.
value: New value of the property.

Description

Synonym for the Movie Clip / Sprite / Object that contains all other objects. i.e. the root. Note that the root has no parent Object.

Sample

```
_root.play();  // restart root Movie Clip
```

See Also

_parent, this

4.6.8.11.81 _rotation

Player Required

SWF4 or later

Syntax

MovieClipName._rotation
MovieClipName._rotation = value;

Arguments

MovieClipName is the name of Movie Clip / Sprite or Object.
value is the new rotation angle in degrees. Positive angle indicates clockwise rotation.

Returns

Rotation angle in degrees. Positive angle indicates clockwise rotation.

Description

Sets / Returns the current rotation angle of the Sprite or Object. This angle is specified in degrees.

Sample

```
_root.rect._rotation += 10;   // rotate rect Object 10 degrees clockwise.
```

Physics Properties are also supported for this property.

4.6.8.11.82 _soundbuftime

Player Required

SWF4 or later

Syntax

_soundbuftime
_soundbuftime = value

Arguments

value: Integer, specifying the number of seconds before the Movie starts to stream.

Returns

Current buffer setting.

Description

Property (global): Specifies the number of seconds of streaming sound to pre-buffer. Default is 5.

Sample

```
_soundbuftime = 10;    // buffer 10 seconds of audio.
```

4.6.8.11.83 _target

Player Required

SWF4 or later

Syntax
MovieClip._target

Arguments
MovieClip is the name of the Movie Clip / Sprite or Object.

Returns
Target path of the Movie Clip / Sprite or Object.

Description
This is a read-only property that returns the path name of the Movie Clip / Sprite or Object. The returned name is in 'slash' notation.
This is used in a comparison statement with the _droptarget property of another Movie Clip / Sprite or Object.
Unlike _name this returns the full path using slash notation.

Sample
Assume a Movie contains two MovieClips, s1 and s2 (s2 is smaller than s1 so it can be completely covered).
The code below is for MovieClip s1, no script is required for s2:

```
onEnterFrame() {
    if (this._droptarget ==_parent.s2._target) {
        trace("zap");
    }
}

onSelfEvent (release) {
    stopDrag();
}

onSelfEvent (press) {
    this.startDragLocked();
}
```

if s2 is a MovieClip owned by the main Movie Clip then its _target name is "/s2".

See Also
_droptarget

4.6.8.11.84 _text +

Player Required
No specific player. SWiSH Max extension to allow access to textfield properties and methods.

Syntax
textscriptobject._text

Arguments
textscriptobject is a text field Script Object.

Returns
Values associated with the textfield property or method that is accessed.

Description
SWiSH Max button, shape and text objects are automatically wrapped in a Movie Clip when the object is named and the target tickbox is checked. If access to the original text field properties and methods are required, these must be accessed via the _text property.

Example:
mytext is a dynamic text object with the target checkbox ticked. The example below shows how the

TextField properties background and backgroundColor can be accessed.

```
onSelfEvent (load)
{
    mytext._text.background = true;
    mytext._text.backgroundColor = 0xFF0000;
}
```

The code when run will turn the background of the text field red.
These Text field properties apply to flash player 6+

4.6.8.11.85 _time +

Player Required
SWF4 or later

Syntax
instanceName._time

Arguments
None.

Returns
Time since the start of the whole Movie (in seconds).

Description
Read-only property. Returns the age of the Movie in seconds.

This is a SWiSH Max-specific Physics Property.

4.6.8.11.86 _totalframes

Player Required
SWF4 or later

Syntax
a = MovieClipName._totalframes

Arguments
MovieClipName is the name of the Movie Clip or Sprite.

Returns
The number of Frames contained within the Movie Clip / Sprite.

Description
Property (read-only): the number of Frames contained within the Movie Clip / Sprite.

This property could be used to create a temperature bar that shows the % complete of download.

The following example uses the _framesloaded property to calculate the % loaded

Sample

```
a = _framesloaded / _totalframes * 100;      // a contains % complete of download.
```

See Also
_framesloaded

4.6.8.11.87 _url

Player Required
SWF4 or later

Syntax
a = MovieClipName._url

Arguments
MovieClipName is the name of the Movie Clip or Sprite.

Returns
The URL of the .swf file that the Movie Clip was downloaded from.

Description
Property (read-only): returns the URL of the .swf file that the Movie Clip was downloaded from.

Sample

```
a = _root._url
```

4.6.8.11.88 _width

Player Required
SWF4 or later

Syntax
MovieClipName._width

Arguments
MovieClipName is the name of the Movie Clip / Sprite or Object.

Returns
This read-only property returns the current width of the Sprite or Object.

Description
This can be used to determine the current width of the referenced Object.
If scaling is applied (_xscale), this property is modified to show the revised size.

Sample

```
w = rect._width;        // w contains the current width of the Object rect.
```

4.6.8.11.89 _valpha +

See Physics Properties.

4.6.8.11.90 _visible

Player Required
SWF4 or later

Syntax
_MovieClipName._visible
_MovieClipName._visible = value

Arguments
MovieClipName: Name of the Movie Clip / Sprite or Object.
value: Boolean value. True indicates visible.

Returns
Current visible status.

Description
Sets / Returns the current visibility status of the Movie Clip / Sprite or Object.
A button that is not visible cannot be clicked.

Sample
```
_root.rect._visible = false; // hide rectangle.
```

4.6.8.11.91 _vx +

See Physics Properties.

4.6.8.11.92 _vxscale +

See Physics Properties.

4.6.8.11.93 _vy +

See Physics Properties.

4.6.8.11.94 _vyscale +

See Physics Properties.

4.6.8.11.95 _x

Player Required
SWF4 or later

Syntax
MovieClipName._x
MovieClipName._x = value;

Arguments
MovieClipName is the name of the Movie Clip / Sprite or Object.
value is the new value of the distance from the left edge of the stage in pixels.

Returns
x coordinate in pixels.

Description
Sets / Returns the current X coordinate of the Sprite or Object. This is the distance from the left edge of the stage in pixels to the specified anchor point of the Object / Sprite.

Sample
```
_root.rect._x += 10;   // moves the Object rect 10 pixels to the right.
```

Physics Properties are also supported for this property.

4.6.8.11.96 _xmouse

Player Required
SWF5 or later

Syntax
myMovieClip._xmouse

Arguments
none

Returns
Nothing.

Description
Read-Only Property; Returns the X coordinate of the mouse's current location.

4.6.8.11.97 _xscale

Player Required
SWF4 or later

Syntax
MovieClipName._xscale
MovieClipName._xscale = percentage;

Arguments
MovieClipName is the name of the Movie Clip / Sprite or Object.
percentage is the new width of the Sprite / Object with respect to the original size as a percentage.
Negative percentage values can be used to horizontally flip the Movie Clip / Sprite / Object.

Returns
Current scale. 100 represents the original size.

Description
This property returns or sets the horizontal scale of the indicated Movie Clip / Sprite / Object.

Sample

```
_root.rect._xscale = 200;     // double the width of the Movie Clip / Sprite / Object.
```

Physics Properties are also supported for this property.

4.6.8.11.98 _y

Player Required
SWF4 or later

Syntax
MovieClipName._y
MovieClipName._y = value;

Arguments
MovieClipName is the name of the Movie Clip / Sprite or Object.
value is the new value of the distance from the top edge of the stage in pixels.

Returns
y coordinate in pixels.

Description
Sets / Returns the current Y coordinate of the Movie Clip / Sprite or Object. This is the distance from the top edge of the stage in pixels to the specified anchor point of the Object / Movie Clip / Sprite.

Sample

```
_root.rect._y += 10;   // moves the Object rect 10 pixels down.
```

Physics Properties are also supported for this property.

4.6.8.11.99 _ymouse

Player Required
SWF5 or later

Syntax

myMovieClip._ymouse

Arguments
none

Returns
Nothing.

Description
Read-Only Property; Returns the Y coordinate of the mouse's current location.

4.6.8.11.100 _yscale

Player Required
SWF4 or later

Syntax
MovieClipName._yscale
MovieClipName._yscale = percentage;

Arguments
MovieClipName is the name of the Movie Clip / Sprite or Object.
percentage is the new height of the Movie Clip / Sprite / Object with respect to the original size as a percentage.
Negative percentage values can be used to vertically flip the Movie Clip / Sprite / Object.

Returns
Current scale. 100 represents the original size.

Description
This property returns or sets the vertical scale of the indicated Movie Clip / Sprite / Object.

Example
```
_root.rect._yscale = 200;     // double the height of the Movie Clip / Sprite / Object.
```

Physics Properties are also supported for this property.

4.6.8.11.101 $version

Player Required
SWF5

Syntax
$version

Description
Property: A string that specifies the supported Flash Player version.
The version string is in the format
OS V, 0, ***R***, 0

Where ***OS*** is the operating system.

WIN	Windows Operating system
MAC	Apple / Macintosh OS
UNIX	UNIX / Linux and other derivative operating systems

V is the version number.
R is the revision number.
the number 0 appears after ***V*** and ***R*** items.

Samples

The value returned by the browser will depend on the installed Flash Player plugin "WIN 9,0,47,0" is a possible value.

See Also

$version, getVersion()

4.6.8.11.102 #include

Player Required

Not player related

Syntax

#include <*filename*>

Description

Allows inclusion of external script by reference to the name of the file containing the script.

When SWiSH Max is compiling the script, it reads script from the named file at point the point of the #include.

To avoid the possibility of infinitely nested includes, a named file is only included once.
The file is assumed to exist in the current SWI file folder for files, unless the full path is quoted.

The include directive is invoked at compile time. Therefore, if you make any changes to an included file, you must save the file and recompile any SWF files that uses it.

Note: this is an older style of notation. Current usage is **include <*filename*>**

See Also

include

4.6.8.11.103 add (concat strings) +

Player Required

SWF4 or later

Syntax

expression1 add expression2

Arguments

None.

Returns

Combined string containing both strings.

Description

The string expression2 is concatenated to the string in expression1. This operator converts expressions to strings before concatenation.
Note: In Flash, **add** is a deprecated term that operates identically to the + operator. (ie. it does not force conversion to string).

Sample

```
a = "cat ";
b = "eats dog";
c = a add b; // c now contains "cat eats dog"

trace(5 add 10); // displays "510".
```

See Also

+ operator

4.6.8.11.104 Array (Object)

An Array is an Object that is composed of a number of elements. An element can be any value (eg string) or Object. The types of the elements do not have to be all the same, but usually they will be. The elements are access using an index. The index is usually a numeric value starting from zero for the first element. The index is specified by a value within square brackets after an array variable name. The square brackets are called the array access operator.

Player Required

Supported Internally

Sample

```
dayArray = new Array;
dayArray[0] = "Monday";
dayArray[1] = "Tuesday";
dayArray[2] = "Wednesday";
dayArray[3] = "Thursday";
```

Arrays are a convenient way of handling tables of data. Access to the specific element is via an integer, which could be another variable. This property makes an array an extremely powerful tool, especially when used with looping constructs such as while, do while etc.

The first 3 days of the week could be conveniently traced with the following code:

```
while (i < 3) {
    trace(dayArray[i++]);
}
```

Creating Arrays

Arrays can be created in any of the following ways:

name = new Array;
name[*index*] = "*value*";
name = new Array(*length*);   (**Note:** won't work with SWF4 Movies)
name = new Array(*value1, value2...*);   (**Note:** won't work with SWF4 Movies)

For SWF5+ you need to create an array Object and assign it to a variable BEFORE you can start accessing or setting the variable as an array. Therefore simply going *name*[*index*] = "*value*"; won't work. For this reasons the RECOMMENDED WAY to do arrays that are compatible for both SWF4 and SWF5+ is:

***name* = new Array;**

Object Properties

An array can be have internal fields.
For example:

```
        person = new Array;
        person.push({first:"Tom", last:"Baker"});
        person.push({first:"Andrew", last:"Smith"});
```

These elements can then be referenced via:
person[0].first, person[0].last etc.

Flash MX Differences

The <%SWISHSCRIPT%> array object only support the MX array object methods when you specify SWF5 or higher. For SWF4, no array methods or properties are supported. Also the array constants and and special constructors are only supported for SWF5 or higher. In addition, using the 'new Array' as shown in the above example is not strictly required for SWF4.

4.6.8.11.104.1 Array.concat()

Player Required
SWF5 or later

Syntax
arrayName.concat(*value1, value2, ...*)

Arguments
value(s): Elements, numbers, or strings to be linked with the elements in the array specified by *arrayName*

Returns
Concatonated array.

Description
Method; takes the base array - *arrayName* - and links the *value(s)* to it and creates a new array. If an array is used in the *value* argument, only the elements of that array are added to the base array and the array specified in the *value* argument is unaffected.

Samples
```
onSelfEvent (load) {
    products = new Array("SWiSHlite", "SWiSH Max 2.0");
    more_products = new Array("SWiSH Max", "SWiSHpix");
    trace("Concatenated Array = " add products.concat(more_products));
    trace("Base Array = " add products);
    trace("Base Array with new elements = " add products.concat("Bob", 4));
}
```

The script above displays the following in the debug window:

```
Concatenated Array = SWiSHlite,SWiSH Max 2.0,SWiSH Max,SWiSHpix
Base Array = SWiSHlite,SWiSH Max 2.0
Base Array with new elements = SWiSHlite,SWiSH Max 2.0,Bob,4
```

4.6.8.11.104.2 Array.join()

Player Required
SWF5 or later

Syntax
arrayName.join({*sep*});

Arguments
sep: An optional character or string to be used as the separator for the string that returns the new array elements. Without this optional argument, a comma is used by default.

Returns
String representing the joined elements.

Description
Method; takes the individual elements of an array and converts them to strings. They are joined together by the character or string specified by the sep argument and returns a string containing the joined elements.

Samples
```
onSelfEvent (load) {
    days = new Array("Monday","Tuesday","Wednesday");
    trace(days.join());
}
// displays "Monday,Tuesday,Wednesday" in the debug window

onSelfEvent (load) {
    days = new Array("Monday","Tuesday","Wednesday");
    trace(days.join(":"));
}
// displays "Monday:Tuesday:Wednesday" in the debug window

onSelfEvent (load) {
    pets = new Array("fluffy","spot","Mr.Kitty");
    trace(pets.join(" and "));
}
// displays "fluffy and spot and Mr.Kitty" in the debug window
```

4.6.8.11.104.3 Array.length

Player Required
SWF5 or later

Syntax
arrayName.length

Arguments
None.

Returns
The length of the array.

Description
Property; contains the number of elements in an array.

Note: Using the length property for Arrays will only work with SWF5+ (will not work with SWF4). It is supported by the internal SWiSH Max player.

Samples

```
onSelfEvent (load) {
    products = new Array();
    products[0] = "SWiSHlite";
    products[1] = "SWiSH Max 2.0";
    trace(products.length);
}
// displays '2' in the debug window

onSelfEvent (load) {
    products = new Array();
    products[0] = "SWiSHlite";
    products[1] = "SWiSH Max 2.0";
    products[2] = "SWiSH Max";
    products[3] = "SWiSHsites";
    products[4] = "SWiSHpix";
    trace(products.length);
}
// displays '5' in the debug window

onSelfEvent (load) {
    pets = new Array("fluffy","spot","Mr.Kitty");
    trace(pets.length);
}
// displays '3' in the debug window
```

4.6.8.11.104.4 Array.pop()

Player Required
SWF5 or later

Syntax
arrayName.pop()

Arguments

None.

Returns
The element "popped" from the array.

Description
Method; removes the last element of an array and returns its value.

Samples

```
onSelfEvent (load) {
    pets = new Array("fluffy","spot","Mr.Kitty");
    trace(pets.pop());
}
// displays 'Mr.Kitty' in the debug window
```

Since this method actually removes the last element in the array, using it consecutively will produce different results:

```
onSelfEvent (load) {
    zoners = new Array();
    zoners[0] = "David M.";
    zoners[1] = "Hugh B.";
    zoners[2] = "Roger O.";
    zoners[3] = "David P.";
    trace("1st: " add zoners.pop());
    trace("2nd: " add zoners.pop());
    trace("3rd: " add zoners.pop());
}

// The script above will display the following in the debug window:
1st: David P.
2nd: Roger O.
3rd: Hugh B.
```

See also
Array.shift()

4.6.8.11.104.5 Array.push()

Player Required
SWF5 or later

Syntax
arrayName.push(*value1, value2, ...*)

Arguments
value: One or more values to add at the end of the array.

Returns
Returns the length of the array after the new *value(s)* have been added to it.

Description
Method; contains the length of an array after the new *value(s)* have been added to the end of it.

Samples

```
onSelfEvent (load) {
    products = new Array("SWiSHlite", "SWiSH Max 2.0");
    // array length is '2'
    trace("1st Length = " add products.push("SWiSH Max", "SWiSHsites"));
    // new array length of '4' is displayed in the debug window
    trace("2nd Length = " add products.push("SWiSHpix", "SWiSHstudio"));
    // new array length of '6' is displayed in the debug window
}
```

4.6.8.11.104.6 Array.reverse()

Player Required
SWF5 or later

Syntax
arrayName.reverse();

Arguments
None.

Returns
Nothing.

Description
Method; Reverses the order of the elements in an array.

Samples

```
onSelfEvent (load) {
    days = new Array("Sunday","Monday","Tuesday","Wednesday");
    trace("Initial Array Order = " add days.join());
    days.reverse();
    trace("Reverse Array Order = " add days.join());
}

// displays the following in the debug window:

Initial Array Order = Sunday,Monday,Tuesday,Wednesday
Reverse Array Order = Wednesday,Tuesday,Monday,Sunday
```

4.6.8.11.104.7 Array.shift()

Player Required
SWF5 or later

Syntax
arrayName.shift();

Arguments
None.

Returns
The first element from the specified array.

Description
Method; Removes the first element listed in the specified array and returns its value.

Samples

```
onSelfEvent (load) {
    days = new Array("Sunday","Monday","Tuesday","Wednesday");
    trace(days.shift());
    // displays 'Sunday' in the debug window
    trace(days.shift());
    // displays 'Monday' in the debug window
}
```

See also
Array.pop()

4.6.8.11.104.8 Array.slice()

Player Required
SWF5 or later

Syntax
arrayName.slice(*start {, end}*);

Arguments
start: A number indicating the index value to be used for the starting point of the slice. If a negative number is used for this argument, the starting point will be at the end of the array, for example: -1 indicates the last element in the array; -2 indicates the next to last element in the array; and so on.

end: An optional argument used to indicate the index value for the ending point of the slice. If you choose not to supply this optional argument, then all of the elements from the starting point to the last element are returned. If a negative number is used for this argument, the ending point will be at the end of the array, for example: -1 indicates the last element in the array; -2 indicates the next to last element in the array; and so on.

Returns
A new array containing all of the elements from the *start* index value up to the *end* index value.

Description
Method; Returns a new array containing all of the elements from the *start* index value up to the *end* index value. The element at the ending point is not included, and the original array is not affected by the slice.

Samples
```
onSelfEvent (load) {
    days = new Array();
    days[0] = "Sunday";
    days[1] = "Monday";
    days[2] = "Tuesday";
    days[3] = "Wednesday";
    days[4] = "Thursday";
    days[5] = "Friday";
    days[6] = "Saturday";
    trace(days.slice(3));
}
// displays "Wednesday,Thursday,Friday,Saturday" in the debug window


onSelfEvent (load) {
    days = new Array();
    days[0] = "Sunday";
    days[1] = "Monday";
    days[2] = "Tuesday";
    days[3] = "Wednesday";
    days[4] = "Thursday";
    days[5] = "Friday";
    days[6] = "Saturday";
    trace(days.slice(3,-2));
}
// displays "Wednesday,Thursday" in the debug window
```

4.6.8.11.104.9 Array sorting

Two methods exist to perform array sorting:
Array.sort() and Array.sortOn()

Both of the methods share a options object that allows definition of the type of sort. The options are passed as a bitmap.

Constant name	Value	Description
Array.CASEINSENSITIV	or 1	Case insensitive sort

E		
Array.DESCENDING	or 2	Descending sort (default is ascending)
Array.UNIQUESORT	or 4	Tests for uniqueness of elements. If array contains non unique members, returns 0 and does not modify the array.
Array.RETURNINDEXEDARRAY	or 8	Returns an indexed array indicating the sort order. The original array is not modified. This may result in more efficient sorting as the main array elements are not moved.
Array.NUMERIC	or 16	Considers the array elements to be numeric and does a numeric sort. eg. 5 is less than 100. (alphabetic sort would place 100 < 5)

So for a case insensitive search in descending order you would pass the option 3 (or Array.CASEINSENSITIVE | Array.DESCENDING)

Player Required
SWF5 or later

Syntax
arrayName.sort(*compareFunction:Object[optional]*, *options:Number [optional]*);

Arguments
compareFunction is an optional parameter that defines a function that does the comparison. This allows non standard ordering to be implemented.
The function should return -1 if A < B, 0 if A == B and 1 if A > B.
The function should be of the form: fn(A,B)

options is an optional parameter that defines additional sort options.

Returns
Generally modifies the array and returns nothing.
Exceptions:
option = 4, returns 0 if array contains non unique members.
option = 8, returns index array.

Description
Method; sorts the elements of an array using the < (less than) comparison operator. This can be used to sort elements of an array numerically or alphabetically.
Note: numerical values are sorted alphabetically. (10 appears before 5 because 1 < 5)

Samples
```
onSelfEvent (load) {
    myNumbers = new Array("5","2","1","3","10");
    myNumbers.sort();
    trace(myNumbers.join());
}
// displays "1,10,2,3,5" in the debug window

onSelfEvent (load) {
    team = new Array("John","Brian","Alison","Stephan");
    team.sort();
    trace(team.join());
}
// displays "Alison,Brian,John,Stephan" in the debug window

onSelfEvent (load) {
    myNumbers = new Array("one","two","three","five","six");
    myNumbers.sort(order);
    trace(myNumbers.join());
}
function order(a,b)
{
    // sort strings according to length, if same length, sort alphabetically.
    if (a.length < b.length)
    {
        return -1;
    }
    if (a.length > b.length)
    {
        return 1;
    }
    return (a > b);    // alphabetic sort.
}
// displays "one,six,two,five,three". sort according to length, if same length then alphabetic.
```

Player Required

SWF5 or later

Syntax

arrayName.sortOn(*field, options*);

Arguments

field: a string identifying a field in an element of the array. All elements will be sorted based on the value of this field.
options: is an optional parameter that defines additional sort options.

Returns

Generally modifies the array and returns nothing.
Exceptions:
option = 4, returns 0 if array contains non unique members.
option = 8, returns index array.

Description

Method; sorts the elements in an array based on the specified *field* from the array. If the elements being sorted do not contain the specified *field*, then the sortOn will behave like the standard arrayName.sort method.

Notes

If no sort options are specified, sorting is case sensitive (Z precedes a).
Sorting is ascending (a preceds b).
Sorting is based on a string sort (10 preceeds 2).

Samples

```
onSelfEvent (load) {
        person = new Array;
        person.push({first:"Tom", last:"Baker"});
        person.push({first:"Andrew", last:"Smith"});

        person.sorton("first");       // sorts the array by first names
        for (i=0; i<person.length; i++) {
                trace("First: " add person[i].first add " Last:" add person[i].last);
        }

        for (i=0; i<person.length; i++) {
                trace("First: " add person[i].first add " Last:" add person[i].last);
        }

        person.sorton("last");        // sorts the array by last names
}
```

See Also

Array.sort()

4.6.8.11.104.10 Array.splice()

Player Required

SWF5 or later

Syntax
arrayName.splice(*start,deletecount,{value...}*);

Arguments
start: The array's index value where the splice begins.
deletecount: The number of elements (including the element specified by the *start* argument) to be deleted. If this value is set to zero, then no elements are deleted.
value(s): Optional elements to be added to the array at the index value specified by the *start* argument.

Returns
An array of the elements removed from the original array.

Description
Method; inserts or deletes elements from an array.

Samples
```
onSelfEvent (load) {
    nickNames = new Array("Snookie","Piggy","Shorty","Slim");
    nickNames.splice(3,0,"Tiny");
    trace(nickNames.join());
}
// displays "Snookie,Piggy,Shorty,Tiny,Slim" in the debug window

onSelfEvent (load) {
    nickNames = new Array("Snookie","Piggy","Shorty","Slim");
    nickNames.splice(3,1);
    trace(nickNames.join());
}
// displays "Snookie,Piggy,Shorty" in the debug window
```

4.6.8.11.104.11 Array.toString()

Player Required
SWF5 or later

Syntax
arrayName.toString();

Arguments
None.

Returns
A string.

Description
Method; outputs a string containing the individual elements in the specified Array. The elements will be concatenated and separated by commas.

Sample
```
onSelfEvent (load) {
  days = new Array("Sunday","Monday","Tuesday","Wednesday");
  trace(days.toString());
}
// outputs "Sunday,Monday,Tuesday,Wednesday" to the debug window
```

4.6.8.11.104.12 Array.unshift()

Player Required
SWF5 or later

Syntax
arrayName.unshift(*value1, value2, ...*);

Arguments
value(s): One or more new elements to be added at the beginning of the array.

Returns
The updated length of the base array.

Description
Method; places the new elements specified by the *value* argument at the beginning of the array and returns the updated length of the array.

Samples

```
onSelfEvent (load) {
    days = new Array("Wednesday","Thursday","Friday","Saturday");
    trace(days.length);
    // displays '4' in the debug window
    week = days.unshift("Sunday","Monday","Tuesday");
    trace(week);
    // displays '7' in the debug window
}
```

4.6.8.11.105 beginFill()

Player Required
SWF6 or later

Syntax
myMovieClip.beginFill({*rgb* {,*alpha*})

Arguments
rbg: specifies the color of the fill in a hex color value (eg. Black is 0x000000, white is 0xFFFFFF, etc).
alpha: specifies the opacity of the fill color. This argument is optional and defaults to 100 (opaque) if not provided. Acceptable values are 0 through 100.

Returns
Nothing.

Description
Method; drawing method indicating the start of a new path. If the current position is not equal to the position previously defined by the moveTo method, the path is closed and filled - similar to the endFill method.

See Also
beginFill()
beginGradientFill()
moveTo()
lineTo()
endFill()

Example
The following code will cause a orange square to be drawn.

```
onSelfEvent (load)
{
    this.beginFill(0xFF8800,100);
    this.moveTo(-50,-50);
    this.lineTo(-50,50);
    this.lineTo(50,50);
    this.lineTo(50,-50);
    this.endFill();
}
```

4.6.8.11.106 beginGradientFill()

Player Required
SWF6 or later

Syntax
myMovieClip.beginFill(*fillType, colors, alphas, ratios, matrix*)

Arguments
fillType: either "linear" or "radial"
colors: an array of hex color values that will be used in the gradient fill (eg. 0x000000 is black, 0xFFFFFF is white, etc.)
alphas: an array of opacity levels that correspond with the color arrays. Acceptables values range from 0 (transparent) through 100 (opaque).
ratios: An array of color distribution ratios. Acceptables values range from 0 through 255.
matrix: A transformation matrix that is an object with either of the following two sets of properties.

- a, b, c, d, e, f, g, h, i, which can be used to describe a 3 x 3 matrix of the following form:

$$\begin{pmatrix} \text{a (width)} & \text{b (0)} & \text{c (0)} \\ \text{d (0)} & \text{e (height)} & \text{f (0)} \\ \text{g (x start)} & \text{h (y start)} & \text{i (1)} \end{pmatrix}$$

If a matrixType property is omitted then the remaining arguments are all required; the function fails if any are missing. This *matrix* rotates, scales, and skews the unit gradient which is defined at (-1,-1) and (1,1).

The above use of the values a..i assumes a non rotated gradient running from left to right.

This gradient can be scaled, moved or rotated by multiplying by the following matrixes:
scale matrix:

$$\begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

translate matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{pmatrix}$$

rotate matrix:

$$\begin{pmatrix} \cos(r) & \sin(r) & 0 \\ -\sin(r) & \cos(r) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- matrixType, x, y, w, h, r.

The properties indicate the following: matrixType is the string "box".
x and y are the horizontal (x) and vertical (y) positions relative to the anchor point of the parent Movie Clip /

Sprite for the top-left corner of the gradient.
w; is the width of the gradient.
h; is the height of the gradient
r; is the rotation of the gradient in radians.

If a matrixType property exists then it must equal "box" and the remaining arguments are all required. The function fails if any of these conditions are not met.

Returns
Nothing.

Description
Method; drawing method indicating the start of a new drawing path. This method will fail if any of the following occur:

- The number of elements in the colors, alphas, and ratios arrays are not equal
- The fillType argument is not defines as either "linear" or "radial"
- Any field for the matrix agrument are missing or invalid.

Example
The script below demonstrates the drawing of a 100x80 rectangle around the center point of an empty movie clip.
The gradient is created that goes from Green to Red.

```
onSelfEvent (load)
{
    var colors = [0x00FF00,0xFF0000];     // colors green, red
    var alphas = [100,100];
    var ratios = [0x00,0xFF];
    matrix = {a:100,b:0,c:0,d:0,e:80,f:0,g:0,h:0,i:1};
    this.lineStyle(1,0);
    this.beginGradientFill("linear",colors,alphas,ratios,matrix);
    // draw rectangle
    this.moveTo(-50,-40);     // TLHC
    this.lineTo(50,-40);      // TRHC
    this.lineTo(50,40);         // BRHC
    this.lineTo(-50,40);         // BLHC
    this.lineTo(-50,-40);     // back to TLHC
    this.endFill();
}
```

Note: Use matrix = {a:71,b:71,c:0,d:-56.8,e:56.8,f:0,g:0,h:0,i:1}; to rotate the gradient by 45 degrees. This matrix was obtained by multiplying the matrix above by the rotation matrix specified above. (cos(45 deg) = 0.71, sin(45 deg) = 0.71).

See Also
beginFill()
beginGradientFill()
moveTo()
lineTo()
endFill()
Matrix Multiplication
Transformation Matrix

4.6.8.11.107 Boolean (Object)

A boolean Object is used to store a boolean (true / false) value.
The constants true and false are provided to allow assignment of those values.

Player Required

Supported Internally

Sample

```
myBool = true;
myBool = false;
myBool = (a < 5);      // true if a was previously assigned a value < 5
```

4.6.8.11.108 break

Player Required
SWF4 or later

Syntax
break

Arguments
None.

Returns
Nothing.

Description
Action; used within a conditional loop or within a block of statements associated with a particular case in a switch action. When used within a loop, the break action tells the script to abort the loop and execute the statement immediately following the loop. When used within a switch action, the break action tells the script to skip the remaining statements in that case and jump to the next statement after the enclosing switch action.

Note: Within SWiSH Max, the break action cannot be used to break out of a with or tellTarget action.

See Also
while, for, switch, do while

4.6.8.11.109 Button (Object)

A Button Object is a Scripting Object that supports the additional Event Handling function on(). Events can also be handled using the Event Methods associated with a button.

When using Event Methods it is necessary to consider if the button has the "target" checkbox ticked. (See Script Object and _button for more information).

If the Target checkbox is ticked, then the event methods must be accessed via the SWiSH Max specific property _button. Otherwise, the methods can be accessed directly. eg. myButton.onEventFunction() = { function script }

4.6.8.11.109.1 enabled

Player Required
SWF6 or later

Syntax
buttonName.enabled

Arguments
None.

Returns

The current enabled status of the button.

Description
Property; a Boolean value used to enable or disable a button. By default, this value is set to *true*.

Sample
```
onFrame (10) {
    button1.enabled = false; // disables the specified button at Frame 10
}

onFrame (20) {
    button1.enabled = true; // re-enables the specified button at Frame 20
}
```

4.6.8.11.109.2 tabEnabled

Player Required
SWF6 or later

Syntax
buttonName.tabEnabled

Arguments
None.

Returns
The current tabEnabled status of the button.

Description
Property; determines whether a Button, Movie Clip / Sprite, or TextField object is to be included in automatic or custom tab ordering. By default this property is set to true and will be included in tab ordering. If the property is set to false, then the object is not included in automatic or custom tab ordering. If a Movie Clip / Sprite has the tabEnabled property set to false, the child Movie Clip / Sprites inside of it could still be included in tab ordering - unless the tabChildren property is also set to false.

Sample
```
onSelfEvent (load) {
    button1.tabEnabled = false;
}
// disables tab ordering for this button
```

4.6.8.11.109.3 tabIndex

Player Required
SWF6 or later

Syntax
buttonName.tabIndex

Arguments
None.

Returns
The current tabIndex value.

Description
Property; allows you to define the tab ordering for a Button, Movie Clip / Sprite, or TextField object. If any object has a tabIndex value, then automatic tab ordering is disabled. The tab ordering will be determined by the non-negative integer used in the tabIndex property. A object with a tabIndex value of 1 will become active in the tab ordering before an object with a tabIndex value of 2.

Note: If the tabIndex property is not defined - the order will be from top to bottom. You can re-order fields in this manner to avoid the SWF6 restriction.

Sample

```
onSelfEvent (load) {
    button1.tabIndex = 1;
    button2.tabIndex = 2;
    button3.tabIndex = 3;
    myMovieClip1.tabIndex = 4;
    myTextField1.tabIndex = 5;
}
```

4.6.8.11.109.4 trackAsMenu

Player Required

SWF6 or later

Syntax

buttonName.trackAsMenu

Arguments

None.

Returns

The current trackAsMenu status of the button.

Description

Property; determines if a Button object or Movie Clip / Sprite can receive mouse release events. By default, the value of trackAsMenu is set to false.

4.6.8.11.109.5 useHandCursor

Player Required

SWF6 or later

Syntax

buttonName.useHandCursor

Arguments

none

Returns

The current useHandCursor status of the button.

Description

Property; determines whether the specified button uses a hand cursor or not. The default value of *true* tells the button to use the hand cursor while a value of *false* tells it to use the arrow cursor. This property can be changed at any time

Sample

```
onSelfEvent (load) {
    buttonName.useHandCursor = false;
}
// uses the standard mouse pointer when over the named button

onFrame(10) {
    buttonName.useHandCursor = true;
}
// displays the hand cursor when over the named button
```

4.6.8.11.110 call *

Player Required
SWF4 or later

Syntax
call(frame);

Arguments
frame: The name or number of a Frame in the current Timeline.

Returns
Nothing.

Description
Action: executes the script in the called Frame without moving the playhead to that Frame. Local variables will not exist once the script is finished executing.

Depreciated
For script readability reasons, it is recommended that you use the function Action instead.

4.6.8.11.111 call function

Player Required
SWF5 or later

Syntax
object.function([parameter1, [parameter2, [parameter3, [parameter4]]]])

Arguments
object: Name of the Object / Movie Clip / Sprite containing the function.
function: Name of the function being called.
parameter1: Optional input parameter.
parameter2: Optional input parameter.
parameter3: Optional input parameter.
parameter4: Optional input parameter.

Returns
Value defined by return Action.

Description
Allows you to call a user-defined function. Functions are useful when the same processing needs to be applied to different variables or Objects.

Sample
```
onSelfEvent (load) {
    a = 0;
    r = 3;
    a = _root.area(r);
    trace(a);
}

function area(p) {
    return Math.PI * p * p;
}

// On completion, a contains the area for a radius of 3 (pi * r^2), 28.2743346691132
```

See Also
function

4.6.8.11.112 case

Player Required
SWF4 or later

Syntax
case *expression*: *statements*

Arguments
expression Any expression.

statements Any statements.

Returns
Nothing.

Description
Keyword; used to define a condition for the switch action. The *statements* will be executed only if the *expression* strictly equals the expression argument of the switch action.

See Also
switch, default

4.6.8.11.113 chr()

Player Required
SWF4 or later

Syntax
chr(number)

Arguments
number: Integer representing an ASCII code number.

Returns
The character corresponding to the code number.

Description
String function. Converts ASCII code numbers to characters.

Sample
```
myVar = chr(65);       // myVar now contains the character "A"
```

See Also
ord

4.6.8.11.114 clear()

Player Required
SWF6 or later

Syntax
myMovieClip.clear()

Arguments
None.

Returns
Nothing.

Description
Methods; clears all of the previously assigned drawing commands for the specified MovieClip..

4.6.8.11.115 Color (Object)

The Color object is used to alter the RGB properties of a specified Movie Clip / Sprite or other scripted object.

4.6.8.11.115.1 new Color()

Player Required
SWF5 or later

Syntax
new Color(*target*);

Arguments
target: the target path of a Movie Clip / Sprite or instance.

Returns
Nothing.

Description
Constructor; creates a new Color Object and can be used to manipulate the color of the target Movie Clip / Sprite.

Sample

```
myColor = new Color(_parent.myMovieClip);
```

4.6.8.11.115.2 getRGB()

Player Required
SWF5 or later

Syntax
colorObject.getRGB();

Arguments
None.

Returns
An integer representing the color.
The integer should be interpreted as a hex number of the format
0xRRGGBB.
ie. the least significant byte contains the Blue level (0-255)
The next most significant byte contains the Green level (0-255)
The next most significant byte contains the Red level (0-255)

Description
Method; returns the numeric values of the Color Object as defined by the last setRGB method.

4.6.8.11.115.3 setRGB()

Player Required
SWF5 or later

Syntax
colorObject.setRGB(*0xRRGGBB*);

Arguments
0xRRGGBB; defines either the hexadecimal or Red/Green/Blue color to be set. *RR*, *GG*, and *BB* need to be two hexidecimal digits each. 0x tells the script that the number is hexidecimal.

Returns
Nothing.

Description
Method; defines the RGB color for the specified Color Object.

Sample
```
onSelfEvent (load) {
  myColor = new Color(myMovieClip);
  myColor.setRGB(0xFF6600);
}
// sets the color object to a value of Red=255, Green=102, Blue=0 or #FF6600 (Orange)
```

4.6.8.11.116 const

Player Required
SWF5 or later

Syntax
const constName [: typeName1] [= value1] [...,constNameN [: typeNameN] [= valueN]]

Arguments
constName An identifier.

typeName A type keyword, one of the default types or a custom class member.

value The value assigned to the constant.

Returns
Nothing.

Description
Action: used to declare local or Timeline constants.

If you declare constants inside a function, the constants are local. They are defined for the function and expire at the end of the function call. Otherwise the constants are interpreted as Timeline constants.

The const statement supports optional types. These types may be used by the compiler to test whether the values assigned to constants are of the expected type, and that the constant is not used where a different type of value is required.

Sample:

```
function f ( x ) {
    const root2 : Number = 1.414213562;
    return y*y;
}
```

4.6.8.11.117 continue

Player Required
SWF4 or later

Syntax
continue

Arguments
None.

Returns
Nothing.

Description
Action; used in conditional loops and behaves differently based on which type of loop is used.

If continue is used within a while loop, it tells the script to skip the rest of the loop and return to the beginning of the loop - where the condition is tested.

If continue is used within a do..while loop, it tells the script to skip the rest of the loop and jump to the bottom of the loop, where the condition is tested.

If continue is used within a for loop, it tells the script to skip the rest of the loop and jump to the evaluation of the loop's post-expression.

If continue is used within a for...in loop, it tells the script to skip the rest of the loop and jump to the top of the loop and continue with the next value in the enumeration.

4.6.8.11.118 createEmptyMovieClip()

Player Required
SWF6 or later

Syntax
myMovieClip.createEmptyMovieClip(*instanceName, depth*)

Arguments
instanceName: a string defining the name of the new MovieClip instance.
depth: a number indicating the depth level at which the new MovieClip will reside. Use this. getNextHighestDepth() (requires flash 7 or later) to obtain the next available level.

Returns
Reference to newly created movie clip.

Description
Method; creates a new empty MovieClip. The new MovieClip becomes a child of the existing Sprite/MovieClip. The anchor point of the empty MovieClip is set to top-left. If any of the arguments are omitted, the function will fail.

4.6.8.11.119 createEmptySprite *

Player Required
SWF6 or later

Syntax
myMovieClip.createEmptySprite(*instanceName, depth*)

Arguments
instanceName: a string defining the name of the new Sprite instance.
depth: the depth level at which the new Sprite will reside.

Returns
Reference to newly created movie clip.

Description
Method; creates a new empty Sprite. The new Sprite becomes a child of the existing Sprite/MovieClip. The anchor point of the empty Sprite is set to top-left. If any of the arguments are omitted, the function will fail.

See Also
createEmptyMovieClip()

4.6.8.11.120 createTextField()

Player Required
SWF6 or later

Syntax
myMovieClip.createTextField(*instanceName, depth, x, y, width, height*)

Arguments
instanceName: a string defining the name of the new text instance.
depth: the depth level at which the new Movie Clip / Sprite will reside.
x: The x position of the text field (relative to the containing movie clip)
y: The y position of the text field (relative to the containing movie clip)
width: Width of the text field in pixels.
height: Height of the text field in pixels.

Returns
Reference to newly created text field.

Description
Method; creates a new empty text field. If any of the arguments are omitted, the function will fail.
The text field can be further modified using the TextFormat object.
If the depth level already exists, the text will replace the object at that level.
Use MovieClip.getNextHightestDepth() to insert the text field without overwriting any existing object.

See Also
TextFeildObject

4.6.8.11.121 curveTo()

Player Required
SWF6 or later

Syntax
myMovieClip.curveTo(*controlX, controlY, anchorX, anchorY*)

Arguments
controlX: integer value that defines the horizontal position relative to the registration point of the parent Movie Clip / Sprite.
controlY: integer value that defines the vertical position relative to the registration point of the parent Movie Clip / Sprite.
anchorX: integer value that defines the horizontal position relative to the registration point of the parent Movie Clip / Sprite of the anchor point.
anchorY: integer value that defines the vertical position relative to the registration point of the parent Movie Clip / Sprite of the anchor point.

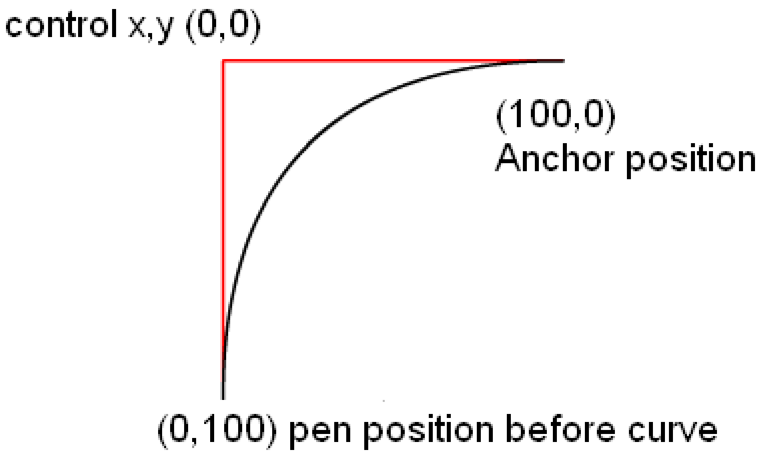
Returns
Nothing.

Description
Method; draws a curve from the current drawing position to the *anchor* points using the *control* points. After the line is drawn, the drawing position is set to the position defined by the anchor points. Lines are drawn underneath any objects placed inside the Movie Clip / Sprite from SWiSH Max (such as images, shapes, buttons, etc.). If no drawing position is set by a moveTo method, then the drawing position defaults to 0,0. If any of the arguments are omitted, the function will fail.

Example
The following script which was placed into the load event of an empty Movie Clip draws the two tangent lines in red then a curve between them.

In this example, the control point is 0,0 (apex of the red lines).
The pen position before the curveTo() command was 0,100 (bottom of vertical red line).
The anchor position (destination point) is (100,0) (RHS of horizontal red line).

```
onSelfEvent (load)
{
    this.moveTo(100,0);
    this.lineStyle(1,0xFF0000);
    this.lineTo(0,0);    // draw horizontal red line
    this.lineTo(0,100);  // draw vertical line
    this.lineStyle(1,0); // black line
    this.curveTo(0,0,100,0); // note pen was at 0,100
}
```



4.6.8.11.122 Date (Object)

This Object allows you access to the various date properties and constants. By using or creating a new Date Object, you can retrieve the Local or Universal Time and/or create a unique Date Object.

Local Time
Local Time is often referred to in the methods used for the Date Object. Local Time is determined by the operating system running the Flash Player.

The various Date Object Methods are listed below:

Method	Description
new Date()	The Date Object
Date.getDate()	Retrieves the day of the month
Date.getDay()	Retrieves the day of the week
Date.getFullYear()	Retrieves the full four-digit year.
Date.getHours()	Retrieves the current hour.
Date.getMilliseconds()	Retrieves the current millisecond.
Date.getMinutes()	Retrieves the current minute.
Date.getMonth()	Retrieves the current month.
Date.getSeconds()	Retrieves the current second.
Date.getTime()	Retrieves the number of milliseconds that have passed since midnight January 1, 1970 (universal time)
Date.getTimezoneOffset()	Retrieves the difference between the local time of the system running the Flash player and Universal Time.
Date.getUTCDate()	Retrieves the day of the month according to Universal Time.
Date.getUTCDay()	Retrieves the day of the week according to Universal Time.
Date.getUTCFullYear()	Retrieves the full four-digit year according to Universal Time.
Date.getUTCHours()	Retrieves the hour according to Universal Time.
Date.getUTCMilliseconds()	Retrieves the millisecond according to Universal Time.
Date.getUTCMinutes()	Retrieves the minute according to Universal Time.
Date.getUTCMonth()	Retrieves the month according to Universal Time.
Date.getUTCSeconds()	Retrieves the second according to Universal Time.
Date.getYear()	Retrieves the current two-digit year. The year is the full (four-digit) year minus 1900
Date.setDate()	Sets the current date.
Date.setFullYear()	Sets the full four-digit year
Date.setHours()	Sets the hour.
Date.setMilliseconds()	Sets the millisecond.
Date.setMinutes()	Sets the minute.
Date.setMonth()	Sets the month.
Date.setSeconds()	Sets the second.
Date.setTime()	Sets the date for the Date Object specified since 12:00am on January 1, 1970
Date.setUTCDate()	Sets the date according to Universal Time.
Date.setUTCFullYear()	Sets the full four-digit year according to Universal Time.
Date.setUTCHours()	Sets the hour according to Universal Time.
Date.setUTCMilliseconds()	Sets the millisecond according to Universal Time.
Date.setUTCMinutes()	Sets the minute according to Universal Time.
Date.setUTCMonth()	Sets the month according to Universal Time.
Date.setUTCSeconds()	Sets the second according to Universal Time.
Date.setYear()	Sets the year. Two-Digit numbers are added to 1900 and used as the year; four-digit numbers are used as written.
Date.toString()	Converts the date object to a string in readable format.

4.6.8.11.122.1 new Date()

Player Required
SWF5 or later

Syntax

DateObj = new Date(*year, month {, date {, hour {, min {, sec {, ms }}}}}*);

Arguments
year: This is a four-digit number used to specify the year.
month: This is a number from 0 to 11 used to represent the month (0 being January, 1 being February, etc.)
date: This is a number from 1 to 31 used to represent the day of the month. This is an optional argument.
hour: A number from 0 to 23 used to represent the current hour (0 being midnight, 23 being 11:00pm). This is an optional argument.
min: A number from 0 to 59 used to represent the current minute. This is an optional argument.
sec: A number from 0 to 59 used to represent the current minute. This is an optional argument.
ms: A number from 0 to 999 used to represent the current millisecond. This is an optional argument.

Returns
an integer

Description
Object; This is the used to create or define a new Date Object. It can be used to create a new Date Object based on the current Local Time, or it can be used to create a reference Date Object based on the arguments assigned to it.

4.6.8.11.122.2 getDate()

Player Required
SWF5 or later

Syntax
DateObj.getDate()

Arguments
none

Returns
an integer

Description
Method; returns the day of the month of the Date Object specified. The result is an integer from 1 to 31 and is according to Local Time.

4.6.8.11.122.3 getDay()

Player Required
SWF5 or later

Syntax
DateObj.getDay()

Arguments
none

Returns
an integer

Description
Method; returns an integer representing the day of the week of the Date Object according to Local Time. The result is an integer from 0 to 6 (0 being Sunday, 1 being Monday, 2 being Tuesday, etc.)

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getDay());
}

// If it is Sunday on your system clock - "0" will be returned in the debug window.
```

4.6.8.11.122.4 getFullYear()

Player Required
SWF5 or later

Syntax
DateObj.getFullYear()

Arguments
none

Returns
an integer

Description
Method; returns an integer (a four-digit number) representing the full year of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getFullYear());
}

// If it is 2003 on your system clock - "2003" will be returned in the debug window.
```

4.6.8.11.122.5 getHours()

Player Required
SWF5 or later

Syntax
DateObj.getHours()

Arguments
none

Returns
an integer

Description
Method; returns an integer (from 0 to 23; 0 being midnight, 23 being 11:00pm) representing the current hour of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getHours());
}

// If it is 5:00pm on your system clock - "17" will be returned in the debug window.
```

4.6.8.11.122.6 getMilliseconds()

Player Required
SWF5 or later

Syntax
DateObj.getMilliseconds()

Arguments
none

Returns
an integer

Description
Method; returns an integer (from 0 to 999) representing the current millisecond of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getMilliseconds());
}

// returns a number between 0 and 999 in the debug window
```

4.6.8.11.122.7 getMinutes()

Player Required
SWF5 or later

Syntax
DateObj.getMinutes()

Arguments
none

Returns
an integer

Description
Method; returns an integer (from 0 to 59) representing the current minute of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getMinutes());
}

// If it is 5:38pm on your system clock - "38" will be returned in the debug window.
```

4.6.8.11.122.8 getMonth()

Player Required
SWF5 or later

Syntax
DateObj.getMonth()

Arguments
none

Returns
an integer

Description
Method; returns an integer (from 0 to 11; 0 being January, 1 being February, etc.) representing the current second of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getMonth());
}

// If it is March on your system clock - "2" will be returned in the debug window.
```

4.6.8.11.122.9 getSeconds()

Player Required
SWF5 or later

Syntax
DateObj.getSeconds()

Arguments
none

Returns
an integer

Description
Method; returns an integer (from 0 to 59) representing the current second of the Date Object according to Local Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getSeconds());
}

// If it is 5:38:23 pm on your system clock - "23" will be returned in the debug window.
```

4.6.8.11.122.10 getTime()

Player Required
SWF5 or later

Syntax
DateObj.getTime()

Arguments
none

Returns
an integer

Description

Method; returns an integer representing the number of milliseconds that have passed since midnight January 1, 1970 (universal time) for the Date Object.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getTime());
}
```

4.6.8.11.122.11 getTimezoneOffset()

Player Required
SWF5 or later

Syntax
DateObj.getTimezoneOffset()

Arguments
none

Returns
an integer

Description
Method; returns an integer representing the difference between the local time of the system running the Flash player and Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getTimezoneOffset());
}

// If your system clock is set at GMT-5 (Eastern USA, Canada) ... "300" will be displayed in the
debug window (60 minutes * 5 hours offset)
```

4.6.8.11.122.12 getUTCDate()

Player Required
SWF5 or later

Syntax
DateObj.getUTCDate()

Arguments
none

Returns
an integer

Description
Method; returns an integer (1 - 31) representing the current day of the month for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCDate());
}
```

4.6.8.11.122.13 getUTCDay()

Player Required
SWF5 or later

Syntax
DateObj.getUTCDay()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 6; 0 being Sunday, 1 being Monday, etc.) representing the current day of the week for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCDay());
}
```

4.6.8.11.122.14 getUTCFullYear()

Player Required
SWF5 or later

Syntax
DateObj.getUTCFullYear()

Arguments
none

Returns
an integer

Description
Method; returns an integer (4-digit number) representing the current full year for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCFullYear());
}
```

4.6.8.11.122.15 getUTCHours()

Player Required
SWF5 or later

Syntax
DateObj.getUTCHours()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 23; 0 being midnight, 23 being 11:00pm, etc.) representing the current hour for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCHours());
}
```

4.6.8.11.122.16 getUTCMilliseconds()

Player Required
SWF5 or later

Syntax
DateObj.getUTCMilliseconds()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 999) representing the current millisecond for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCMilliseconds());
}
```

4.6.8.11.122.17 getUTCMinutes()

Player Required
SWF5 or later

Syntax
DateObj.getUTCMinutes()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 59) representing the current minute for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCMinutes());
}
```

4.6.8.11.122.18 getUTCMonth()

Player Required
SWF5 or later

Syntax
DateObj.getUTCMonth()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 12) representing the current month for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCMonth());
}
```

4.6.8.11.122.19 getUTCSeconds()

Player Required
SWF5 or later

Syntax
DateObj.getUTCSeconds()

Arguments
none

Returns
an integer

Description
Method; returns an integer (0 - 59) representing the current second for the Date Object according to Universal Time.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getUTCSeconds());
}
```

4.6.8.11.122.20 getYear()

Player Required
SWF5 or later

Syntax

DateObj.getYear()

Arguments
none

Returns
an integer

Description
Method; returns an integer representing the year for the Date Object according to Local Time. The year is the full (4-digit) year minus 1900.

Sample

```
onFrame (1) {
    theDate = new Date();
    trace(theDate.getYear());
}

// If the year on your system is 2003 ... "103" will be displayed in the debug window.
```

4.6.8.11.122.21 setDate()

Player Required
SWF5 or later

Syntax
DateObj.setDate(*date*)

Arguments
date: an integer from 1 to 31

Returns
an integer

Description
Method; this method will set the day of the month for the Date object specified. The day of the month is set according to Local Time.

4.6.8.11.122.22 setFullYear()

Player Required
SWF5 or later

Syntax
DateObj.setFullYear(*year {,month{,date}}*)

Arguments
year: This is a four-digit number used to specify the year.
month: This is a number from 0 to 11 used to represent the month (0 being January, 1 being February, etc.) This is an optional argument.
date: This is a number from 1 to 31 used to represent the day of the month. This is an optional argument.

Returns
an integer

Description
Method; this method sets the year for the Date Object specified. All arguments (year, month, date) are set according to Local Time.

4.6.8.11.122.23 setHours()

Player Required
SWF5 or later

Syntax
DateObj.setHours(*hour*)

Arguments
hour: A number from 0 to 23 (0 being midnight, 23 being 11:00pm)

Returns
an integer

Description
Method; This method sets the hour for the Date Object specified according to Local Time.

4.6.8.11.122.24 setMilliseconds()

Player Required
SWF5 or later

Syntax
DateObj.setMilliseconds(*ms*)

Arguments
ms: A number from 0 to 999 used to represent the current millisecond.

Returns
an integer

Description
Method; This method sets the current millisecond for the Date Object specified according to Local Time.

4.6.8.11.122.25 setMinutes()

Player Required
SWF5 or later

Syntax
DateObj.setMinutes(*min*)

Arguments
min: A number from 0 to 59 used to represent the current minute.

Returns
an integer

Description
Method; This method sets the current minute for the Date Object specified according to Local Time.

4.6.8.11.122.26 setMonth()

Player Required
SWF5 or later

Syntax

DateObj.setMonth(*month {, date}*)

Arguments
month: A number from 0 to 11 used to represent the current month (0 being January, 1 being February, etc.)
date: A number from 1 to 31 used to represent the current day of the month. This is an optional argument.

Returns
an integer

Description
Method; This method sets the current month for the Date Object specified according to Local Time.

4.6.8.11.122.27 setSeconds()

Player Required
SWF5 or later

Syntax
DateObj.setSeconds(*sec*)

Arguments
sec: A number from 0 to 59 used to represent the current second.

Returns
an integer

Description
Method; This method sets the current second for the Date Object specified according to Local Time.

4.6.8.11.122.28 setTime()

Player Required
SWF5 or later

Syntax
DateObj.setTime(*ms*)

Arguments
ms: An integer. 0 is used to represent 0:00 GMT 1970 Jan 1.

Returns
an integer

Description
Method; This method sets the date for the Date Object specified since 12:00am on January 1, 1970. The new time is returned in milliseconds.

4.6.8.11.122.29 setUTCDate()

Player Required
SWF5 or later

Syntax
DateObj.setUTCDate(*date*)

Arguments
date: A number from 1 to 31.

Returns
an integer

Description
Method; This method is used to set the date for the Date Object specified according to Universal Time.

4.6.8.11.122.30 setUTCFullYear()

Player Required
SWF5 or later

Syntax
DateObj.setUTCFullYear(*year {, month {, date}}*)

Arguments
year: This is a four-digit number used to specify the year.
month: This is a number from 0 to 11 used to represent the month (0 being January, 1 being February, etc.) This is an optional argument.
date: This is a number from 1 to 31 used to represent the day of the month. This is an optional argument.

Returns
an integer

Description
Method; This method is used to set the full four-digit year for the Date Object specified according to Universal Time.

4.6.8.11.122.31 setUTCHours()

Player Required
SWF5 or later

Syntax
DateObj.setUTCHours(*hour {, min {, sec {, ms}}}*)

Arguments
hour: A number from 0 to 23 used to represent the current hour (0 being midnight, 23 being 11:00pm)
min: A number from 0 to 59 used to represent the current minute. This is an optional argument.
sec: A number from 0 to 59 used to represent the current minute. This is an optional argument.
ms: A number from 0 to 999 used to represent the current millisecond. This is an optional argument.

Returns
an integer

Description
Method; This method is used to set the current hour for the Date Object specified according to Universal Time.

4.6.8.11.122.32 setUTCMilliseconds()

Player Required
SWF5 or later

Syntax

DateObj.setUTCMilliseconds(*ms*)

Arguments
ms: A number from 0 to 999 used to represent the current millisecond.

Returns
an integer

Description
Method; This method is used to set the current millisecond for the Date Object specified according to Universal Time.

4.6.8.11.122.33 setUTCMinutes()

Player Required
SWF5 or later

Syntax
DateObj.setUTCMinutes(*min {, sec {, ms}}*)

Arguments
min: A number from 0 to 59 used to represent the current minute.
sec: A number from 0 to 59 used to represent the current minute. This is an optional argument.
ms: A number from 0 to 999 used to represent the current millisecond. This is an optional argument.

Returns
an integer

Description
Method; This method is used to set the current minute for the Date Object specified according to Universal Time.

4.6.8.11.122.34 setUTCMonth()

Player Required
SWF5 or later

Syntax
DateObj.setUTCMonth(*month {, date}*)

Arguments
month: A number from 0 to 11 used to represent the current month (0 being January, 1 being February, etc.)
date: A number from 1 to 31 used to represent the current day of the month. This is an optional argument.

Returns
an integer

Description
Method; This method is used to set the current month for the Date Object specified according to Universal Time. It can also be used to set the day of the month (date), but that is optional.

4.6.8.11.122.35 setUTCSeconds()

Player Required
SWF5 or later

Syntax
DateObj.setUTCSeconds(*sec {, ms}*)

Arguments
sec: A number from 0 to 59 used to represent the current minute.
ms: A number from 0 to 999 used to represent the current millisecond. This is an optional argument.

Returns
an integer

Description
Method; This method is used to set the current second for the Date Object specified according to Universal Time.

4.6.8.11.122.36 setYear()

Player Required
SWF5 or later

Syntax
DateObj.setYear(*year*)

Arguments
year: A two-digit number between 0 and 99, or a full four-digit number used to represent the year.

Returns
an integer

Description
Method; This method sets the current year for the Date Object specified according to Local Time. If the *year* argument is a two digit number between 0 and 99, then that number is added to 1900 then used as the year. If the *year* argument is a four-digit number, then that number by itself is used to represent the year.

4.6.8.11.122.37 toString()

Player Required
SWF5 or later

Syntax
DateObj.toString()

Arguments
none

Returns
a string

Description
Method; This method will return a string value for the Date Object specified in a human readable format.

4.6.8.11.123 default

Player Required
SWF6 or later

Syntax
default : *statements*

Arguments
statements; Any statements.

Returns
Nothing.

Description
Keyword; used to define the default case for a switch action. The *statements* are only executed if the expression argument of the switch action is not strictly equal to any of the expression arguments following each case keyword for the given switch action.

See Also
switch

4.6.8.11.124 do...while

Player Required
SWF4 or later

Syntax
do {
        statements;
} while (condition);

Arguments
statements: The statements that are to be executed while the condition is true.
condition: An expression that evaluates to a boolean value.

Returns
Nothing.

Description
This Action executes at least once and repeats while the condition remains true.

Sample

```
onSelfEvent (load) {
    a = 1;
    do {
        trace(a);
    } while (a++ < 5);
}
// displays "1,2,3,4,5" to the 'Debug' window.
```

See Also
while and for.

4.6.8.11.125 duplicateMovieClip()

Player Required
SWF4 or later

Syntax
MovieClipName.duplicateMovieClip(newname, depth)

Arguments
MovieClipName: The target path of the Movie Clip / Sprite to duplicate.

newname: A unique identifier for the duplicated Movie Clip / Sprite.

depth: A unique depth level for the duplicated Movie Clip / Sprite. The depth level is a stacking order for duplicated sprites. This stacking order is much like the stacking order of layers in the Timeline; Sprites with a lower depth level are hidden under clips with a higher stacking order. You must assign each duplicated Movie Clip a unique depth level to prevent it from replacing Movie Clips of the same name on occupied depths.

Returns
Reference to duplicated movie clip (SWF6 and later).

Description
Method: Creates an Instance of a Movie Clip / Sprite while the Movie is playing. The playhead in duplicate Movie Clips always starts at Frame 1, regardless of where the playhead is in the original (or "parent") Movie Clip. Variables in the parent Movie Clip are not copied into the duplicate Movie Clip. If the parent Movie Clip is deleted, the duplicate Movie Clip is also deleted. Use the removeMovieClip() method to delete a Movie Clip Instance created with duplicateMovieClip() method.

Sample
This example duplicates the existing Movie Clip, mc1, and creates a new Movie Clip, mc2, on the same level. The position of both Movie Clip / Sprites is adjusted so that they can be seen. To remove the duplicated Movie Clip, use mc2.removeMovieClip().

```
onSelfEvent (load) {
    mc1.duplicateMovieClip("mc2",1);
    mc1._X += 50;
    mc1._Y += 50;
    mc2._X += 150;
    mc2._Y += 50;
}
```

See also
removeMovieClip() and Movie Clip Methods.

4.6.8.11.126 duplicateSprite() +*

NOTE
This is a deprecated function. Use duplicateMovieClip() instead.

Player Required
SWF4 or later

Syntax
SpriteName.duplicateSprite(newname, depth)

Arguments
SpriteName: The target path of the Movie Clip / Sprite to duplicate.

newname: A unique identifier for the duplicated Movie Clip / Sprite.

depth: A unique depth level for the duplicated Movie Clip / Sprite. The depth level is a stacking order for duplicated sprites. This stacking order is much like the stacking order of layers in the Timeline; Sprites with a lower depth level are hidden under clips with a higher stacking order. You must assign each duplicated Movie Clip a unique depth level to prevent it from replacing Movie Clips of the same name on occupied depths.

Returns
Reference to duplicated movie clip (SWF6 and later).

Description

Method: Creates an Instance of a Sprite / Movie Clip while the Movie is playing. The playhead in duplicate

Movie Clips always starts at Frame 1, regardless of where the playhead is in the original Movie Clip. Variables are dynamically created / loaded content in the original Movie Clip are not copied into the duplicate Movie Clip. If the parent of the Movie Clip is deleted, the duplicate Movie Clip is also deleted. Use the removeSprite() method to delete a Movie Clip Instance created with duplicateSprite() method.

Sample

This example duplicates the existing Sprite, s1, and creates a new Sprite, s2, on the same level. The position of both Sprites is adjusted so that they can be seen. To remove the duplicated Sprite, use s2. removeSprite().

```
onSelfEvent (load) {
    s1.duplicateSprite("s2",1);
    s1._X += 50;
    s1._Y += 50;
    s2._X += 150;
    s2._Y += 50;
}
```

See also

removeSprite() and Movie Clip Methods.

4.6.8.11.127 endFill()

Player Required

SWF6 or later

Syntax

myMovieClip.endFill()

Arguments

controlX: integer value that defines the horizontal position relative to the registration point of the parent Movie Clip / Sprite.
controlY: integer value that defines the vertical position relative to the registration point of the parent Movie Clip / Sprite.
anchorX: integer value that defines the horizontal position relative to the registration point of the parent Movie Clip / Sprite of the anchor point.
anchorY: integer value that defines the vertical position relative to the registration point of the parent Movie Clip / Sprite of the anchor point.

Returns

Nothing.

Description

Method; applies a fill to the lines and curves added since the last beginFill or beginGradientFill. If the current drawing position is not the same as the position specified by a previous moveTo method, the path is closed with a line and filled.

See Also

beginFill()
beginGradientFill()
moveTo()
lineTo()
endFill()

Example

The following code will cause a orange square to be drawn.

```
onSelfEvent (load)
{
    this.beginFill(0xFF8800,100);
    this.moveTo(-50,-50);
    this.lineTo(-50,50);
    this.lineTo(50,50);
    this.lineTo(50,-50);
    this.endFill();
}
```

4.6.8.11.128 else

Player Required
SWF4 or later

Syntax
else {...statement(s)...}

Arguments
condition: A boolean expression that evaluates to true or false.
statement(s): An alternative series of statements to run if the condition specified in the if statement is false.

Returns
Nothing.

Description
Action: Specifies the statements to run if the condition in the if statement returns false.

Note: any given if conditional may only contain a single else action.

See also
if; else if

4.6.8.11.129 else if

Player Required
SWF4 or later

Syntax

```
if (condition){
   statement(s);
} else if (condition){
   statement(s);
}
```

Arguments
condition: An expression that evaluates to true or false.
statement(s): An alternative series of statements to run if the condition specified in the if statement is false.

Returns
Nothing.

Description
Action: Used to introduce second (or more) conditional test(s) if initial condition equates to false. Once a condition evaluates to true, after the associated statements have been executed, control passes to the end of the if / else if / else if sequence. This can be used to prevent too many nested levels if standard else construct is used.

Note: An if conditional can contain multiple else if actions and one optional else action after the last else if listed.

Sample

```
if (this._X > maxX) {
    this._X = maxX;
    dx = -dx;  // bounce of right wall
} else if (this._X < minX) {
    this._X = minX;
    dx = -dx;
}
```

The above example is easier to read than the following (equivalent code)

```
if (this._X > maxX) {
    this._X = maxX;
    dx = -dx;  // bounce of right wall
} else {
    if (this._X < minX) {
        this._X = minX;
         dx = -dx;
    }
}
```

See also
if; else

4.6.8.11.130 eval(string)

Player Required
SWF4 (Slash notation only)
SWF5 or later (Dot notation only)

Syntax
eval(expression);

Arguments
expression: A string containing the name of a variable.

Returns
Values as described in Description.

Description
Function: Accesses variables, properties, Objects or Movie Clips by name. If the expression is a variable or a property, the value of the variable or property is returned. If the expression is an Object or Movie Clip / Sprite, a reference to the Object or Movie Clip / Sprite is returned.

If the element named in the expression cannot be found, undefined is returned.

You can also use the eval function to dynamically set and retrieve the value of a variable or Instance name.

Note: The Actionscript eval Action is not the same as the JavaScript eval function, and cannot be used to evaluate statements.

Sample
The following example uses the eval function to determine the value of the expression "piece" + x. Because the result is a variable name, piece3, the eval function returns the value of the variable and assigns it to y:

```
Item5 = "test";
x = 5;
y = eval("Item" add x);
trace(y);
// Output: test
```

Flash MX Differences
Flash MX can use eval to return references to Objects, properties and Movie Clips as well.
<%SWISHSCRIPT%> eval() can only return references to variables.

See Also

variable()

4.6.8.11.131 External Media Class +

When the "Use automatic scripting" option of the External Media Object is selected, the following properties, methods and scripting functions become available for scripting use. These properties, methods and callback functions allow the video to be controlled in various ways.

Properties

bufferLength	A number that specifies the number of seconds of data currently in the buffer. (Read-only).
bufferTime	The number of seconds to buffer in memory before beginning playback of a video stream.
bytesLoaded	The extent of downloading in number of bytes for an HTTP download. (Read-only).
bytesTotal	The total number of bytes downloaded for an HTTP download. (Read-only).
cuePoints	The cue points array read from the FLV file. (Read-only).
currentFps	A number that specifies the number of frames per second being displayed. (Read-only).
lastFrameTime	The timestamp for the last frame. (Read-only).
lastKeyFrameTime	The timestamp for the last frame. (Read-only)
loadedPercentage	The bytes downloaded as a percentage of the total bytes for an HTTP download. (Read-only).
metadata	The metadata object read from the FLV file. (Read-only).
muted	Mute status for the sound
playheadPercentage	The current playhead time or position, measured as a percentage of duration, which can be a fractional value.
playheadTime	The current playhead time or position, measured in seconds, which can be a fractional value.
playing	True if the video is plaing, false if it is paused or stopped. (Read-only)
seekToPrevOffset	The number of seconds that the seekToPrevNavCuePoint() method uses when it compares its time against the previous cue point.
totalTime	The total playing time for the video in seconds. (Read-only).
videoHeight	The height of the video read from the FLV file. (Read-only).
videoWidth	The width of the video read from the FLV file. (Read-only).
volume	Sound volume.

Methods

close()	Closes the video stream and FCS connection.
load(url)	Loads the FLV file but does not begin playing. After resizing (if needed) the FLV file is paused.
pause()	Pauses the video stream.
play()	Begins playing the video stream.
play(url)	Loads the url then begins playing the video stream.
seek(time)	Seeks to a given time in the file, given in seconds, with decimal precision up to milliseconds.
seekPercentage(pct)	Same as seek, but uses percentage of duration
seekSeconds(time)	Same as seek
seekToNextNavCuePoint() or seekToNextNavCuePoint(time)	Seeks to next navigation cue point.
seekToPrevNavCuePoint() or	Seeks to Prev navigation cue point.

seekToPrevNavCuePoint(time)	
stop()	Stops playing the video stream.

Callback functions

onCuePoint	Called when cue point data read (after default cue point processing)
onMetaData	Called when metadata data read (after default metadata processing)
onNotFound	Called when FLV file not found or not loaded
onPlayStart	Called when playing starts
onPlayStop	Called when playing stops because of end of file
onSeekDone	Called when seeking is successful
onStatus	Called when status changes (after default cue point processing)

These functions are called to indicate a specific status or indicate that a previous command (seek) is complete.

Callback function example

The following is the onSelfEvent (load) function for the external media object. This function defines callback functions for onPlayStart and onPlayStop which will display "Play start" and "Play stop" in the debug window when the movie playback starts and then finishes.

```
onSelfEvent (load)
{
    onPlayStart = function () {
        trace("Play start");
    }
    onPlayStop = function () {
        trace("Play stop");
    }
}
```

4.6.8.11.132 false

Player Required
SWF5 or later

Syntax
false

Description
A unique boolean value that represents the opposite of true.

4.6.8.11.133 Filters

Player Required
SWF8 or later

Syntax
As per flash documentation.

Description
Flash player 8 and later provides a number of filters:
BevelFilter
BlurFilter
ColorMatrixFilter

ConvolutionFilter
DisplacementMapFilter
DropShadowFilter
GlowFilter
GradientBevelFilter
GradientGlowFilter

Filters work with SWiSH Max however the full path to the class (typically flash.filters) must be specified.

eg. Typical examples in Flash documentation:

```
import flash.filters.BlurFilter;
var filter:BlurFilter = new BlurFilter(30,30,2);
```

would be coded in SWiSH Max as

```
var filter:flash.filters.BlurFilter = new flash.filters.BlurFilter(30,30,2);
```

or

```
var filter:Object = new flash.filters.BlurFilter(30,30,2);
```

Example
The following code will increase the blur over successive frames that is applied to the current movie clip.

```
onSelfEvent (enterFrame)
{
    var blurX:Number = _currentframe/10;
    var blurY:Number = _currentframe/10;
    var quality:Number = 1;
    var filter:Object = new flash.filters.BlurFilter(blurX, blurY, quality);
    this.filters = [filter];
}
onFrame (300)
{
    stop();
}
```

For more information, see the sample movie filter_effects.swi in the Intermediate samples section.

4.6.8.11.134 focusEnabled

Player Required
SWF6 or later

Syntax
myMovieClip.focusEnabled

Arguments
None.

Returns
Current status of the focusEnabled property.

Description
Property; if the value of this property is false or undefined, the Movie Clip cannot receive input focus unless it has mouse or self events attached to it. Otherwise, if the value is true is can receive input focus regardless.

4.6.8.11.135 for()

Player Required
SWF5 or later

Syntax
for (*initial* ; *condition* ; *end of loop*) {
    statements;
}

Arguments
statements: The statements that are to be executed while the condition is true.

initial: This clause can be used to set up initial conditions and Arguments. Multiple statements can be entered if each statement is separated by a comma.

condition: An expression that evaluates to a boolean value. The loop continues while this is true.
end of loop: A statement that is executed at the end of the loop. This is normally used to increment a loop counter. Multiple statements can be entered if each statement is separated by a comma.

Returns
Nothing.

Description
This Action executes and repeats while the condition remains true. If the condition is initially false, no statements are executed.
This is a more useful statement than while or do while if a loop is to be executed a defined number of times.
It is ideal for initializing arrays.

Sample
```
for (i = 0; i <= 5 ; i++) {
    trace(i);
}
// displays "0,1,2,3,4" in the 'Debug' window.
```

See Also
while, do...while and for.

4.6.8.11.136 for...in

Player Required
SWF5 or later

Syntax
for(*var* in *object*){
    statement(s);
}

Arguments
var: a variable assigned to represent each property of an object or array element.

object: The name of an object to be repeated.

statement(s): An instruction to be executed for each loop.

Returns
Nothing.

Description

Action; loops through individual properties of an object or array elements and initiates the *statement(s)* for each one.

Certain properties cannot be listed by the for or for...in actions. As an example, Movie Clip / Sprite properties such as _x and _y are cannot be listed.

Sample

```
onFrame(1) {
  n = 1;
  products = new Array();
  products[0] = "SWiSHstudio";
  products[1] = "SWiSHsites";
  products[2] = "SWiSHpix";
  products[3] = "SWiSH Max";
  products[4] = "SWiSH Max 2.0";
  products[5] = "SWiSHlite";
  for (var1 in products) {
    trace ("Product" add n add " = " add products[var1]);
    n += 1;
  }
}
```

The above script displays the following in the debug window:

```
Product1 = SWiSHlite
Product2 = SWiSH Max 2.0
Product3 = SWiSH Max
Product4 = SWiSHpix
Product5 = SWiSHsites
Product6 = SWiSHstudio
```

4.6.8.11.137 fscommand()

Player Required

SWF4 or later

Syntax

fscommand("command", "Arguments")

Arguments

command: A string passed to the host application for any use or a command passed to the stand-alone Flash Player.

Arguments: A string passed to the host application for any use or a value passed to the Flash Player.

Returns

Nothing. However fscmd_ret may be assigned a value. This depends on the command called.

Description

Action: Allows the Flash Movie to communicate with:

- The .exe Flash Player.
- The program hosting the Flash Player, such as a Web browser.
- Other programs that can host ActiveX controls such as Visual Basic or Visual C++.

Usage 1: To send a message to the Flash Player, you must use predefined commands and Arguments. The following table shows the values you can specify for the command and Arguments of the fscommand Action to control a Movie playing in the stand-alone Flash Player (including projectors):

Command	Arguments	Purpose
quit	None	Closes the projector
fullscreen	true or false	Specifying true sets the Flash Player to full-screen mode. Specifying false returns the player to normal Menu view

allowscale	true or false	Specifying false sets the player so that the Movie is always drawn at its original size and never scaled. Specifying true forces the Movie to scale to 100% of the player
showmenu	true or false	Specifying true enables the full set of context Menu items. Specifying false dims all the context Menu items except About Flash Player
exec	Path to app.	Executes an application from within the projector
trapallkeys	true or false	Specifying true sends all key events, including accelerator keys, to the onClipEvent(keyDown/keyUp) handler in the Flash Player

Usage 2: To use the fscommand Action to send a message to a scripting language such as JavaScript in a Web browser, you can pass any two Arguments in the command and Arguments. These Arguments can be strings or expressions and are used in a JavaScript function that 'catches', or handles, the fscommand Action.

In a Web browser, the fscommand Action calls the JavaScript function moviename_DoFScommand in the HTML page containing the Flash Movie. The moviename is the name of the Flash Player as assigned by the NAME attribute of the EMBED tag or the ID property of the OBJECT tag. If you assign the Flash Player the name myMovie, the JavaScript function called is myMovie_DoFScommand.

Usage 3: In Visual Basic, Visual C++ and other programs that can host ActiveX controls, fscommand sends a VB event with two strings that can be handled in the environment's programming language.

Samples

In the following example, the fscommand Action sets the Flash Player to scale the Movie to the full monitor screen size when the button is released.

```
on(release){
    fscommand("fullscreen", true);
}
```

4.6.8.11.138 function()

Player Required

SWF5 or later

Syntax

function name (*[arg [: type] [, arg [: type]]...]*) [*: type*] {
  statements;
}

-OR-

function (*[arg [: type] [, arg [: type]]...]*) [*: type*] {
  statements;
}

Arguments

name: Name of the function being defined.
arg: Optional function argument names
type: Optional type of the argument and return value
statements: The script to execute when the function is called

Returns

The first syntax does not return a value.
The second returns a function object.

Description

The first syntax defines a named function. You can call the function with a function call (add link to call function here).

The second syntax is an expression that defines an unnamed function object and returns a reference to it. You can assign that function object to a variable and call the function using that variables. This syntax is often used to set up event handlers in SWF6+.

Both syntaxes support optional argument and return value types. These types may be used by the <%SWISHSCRIPT%> compiler to test whether the values passed to and returned by a function are of the specified types. Types can include Number, Boolean or String.

When the function is called, the script is executed. The function argument names become variables whose values are the values passed to the function. The return value of the called function will be the value used in the return statement.

Sample

Example 1: First Syntax

```
onSelfEvent (load) {
    a = 0;
    r = 3;
    a = _root.area(r);
    trace(a);
}

function area(p) {
    return Math.PI * p * p;
}

// On completion, 'a' contains the area for a radius of 3 (pi * r^2), 28.2743346691132
```

Example 2: Second Syntax

```
onSelfEvent (load) {
    f = function (x) { return x*x; }
    y = f(10); // calls f with 10, returns 100
    trace(y);
}
```

Example 3: Functions in Dynamic Event Handlers (SWF6+)

```
onFrame (10) {
    myMovieClip.onRollOver = function () {
      MyTextField.text = "Roll Me Over!";
    };
}
onFrame (20) {
    myMovieClip.onRollOver = function () {
      MyTextField.text = "Roll Me Over Again";
    };
}
```

See Also
call function

4.6.8.11.139 getBounds()

Player Required
SWF5 or later

Syntax
myMovieClip.getBounds(*targetCoordinateSpace*)

Arguments

targetCoordinateSpace: the path of the target timeline with the coordinate system used for a reference point. This refers to the X and Y coordinates of the target timeline. X=0 on the _root is not necessarily the same as X=0 inside a Movie Clip / Sprite.

Returns
An object with the properties xMin, xMax, yMin, and yMax.

Description
Method; returns properites indicating the bounds of the specified Movie Clip / Sprite for the targetCoordinateSpace argument. You can use localToGlobal or globalToLocal to convert the coordinates of the Movie Clip / Sprite to Stage coordinates (localToGlobal) and vice versa with globalToLocal.

4.6.8.11.140 getBytesLoaded()

Player Required
SWF6 or later

Syntax
MovieClipName.getBytesLoaded()

Arguments
None.

Returns
The number of bytes currently loaded in the specified target.

Description
Read-only property returns the number of bytes currently loaded in the specified target.

Sample
```
onFrame (1) {
    bl = _root.getBytesLoaded();
}
// the variable 'bl' will have a value indicating the total number of bytes loaded at Frame 1.
```

4.6.8.11.141 getBytesTotal()

Player Required
SWF5 or later

Syntax
MovieClipName.getBytesTotal()

Arguments
None.

Returns
The total size in bytes of the specified target.

Description
Read-only property returns the total bytes in the specified target. When using this method for a root path, a level, or a movie loaded into a Movie Clip / Sprite, the value returned is the total bytes of the .SWF file.

Sample
```
onFrame (1) {
    bt = _root.getBytesTotal();
}
// the variable 'bt' will have a value indicating the total number of bytes contained within the root path.
```

4.6.8.11.142 getDepth()

Player Required
SWF5 or later

Syntax
myMovieClip.getDepth()

Arguments
None.

Returns
An integer.

Description
Method; returns the depth of the specified Movie Clip / Sprite.
Each stage object has a unique depth. Objects with a higher depth number appear above items with a lower depth number.
Items created at design time are assigned depths from -16383. Note that some depths may be used by static non scriptable objects.
Newer items are assigned the next available depth number.

Sample
```
A movie contains two movie clips, mc1 and mc2
onFrame (2)
{
    var mc1_depth:Number;
    var mc2_depth:Number;

    mc1_depth = mc1.getDepth();
    mc2_depth = mc2.getDepth();
    trace("mc1:" add mc1_depth add " mc2:" add mc2_depth);    // show the depth of mc1, mc2

    mc1.swapDepths(mc2_depth);    // give mc1 the mc2 depth. Note mc2 gets the mc1 depth.

    mc1_depth = mc1.getDepth();
    mc2_depth = mc2.getDepth();
    trace("mc1:" add mc1_depth add " mc2:" add mc2_depth);    // show the depth of mc1, mc2
}
```

The following is displayed in the debug window:

```
mc1:-16380 mc2:-16381
mc1:-16381 mc2:-16380
```

See Also
swapDepths()
getNextHighestDepth()

4.6.8.11.143 getNextHighestDepth()

Player Required
SWF7 or later

Syntax
myMovieClip.getNextHighestDepth()

Arguments
None.

Returns
An integer.

Description
Method; returns the next available (free) depth.

Note: This is relative to the timeline in the local Movie Clip / Sprite. If adding new elements the main movie, you should use _root.getNextHighestDepth().

See Also
swapDepths()
getDepth()

4.6.8.11.144 getPercentLoaded() +

Player Required
SWF4 or later

Syntax
MovieClipName.getPercentLoaded()

Arguments
None.

Returns
An integer indicating the current percentage of the total bytes loaded for the specified target.

Description
Read-only property returns the current percentage of the total bytes loaded for the specified target. This is the same as multiplying the result of getBytesLoaded by 100 then dividing by the result of getBytesTotal. This method is SWiSH Max specific and is not supported by Flash.

Sample

```
onFrame (1) {
    p = _root.getPercentLoaded();
}
// the variable 'p' will have a value indicating the total percentage of bytes loaded for the root path at Frame
```

4.6.8.11.145 getProperty(property)

Player Required
SWF4 or later

Syntax
[*instance*.]getProperty([instancename ,] property)

Arguments
instancename: The instance name of a Movie Clip for which the property is being retrieved. This can either be the Instance name (using slash notation) or the _target property. If omitted, the Instance is specified by *instance*. if *instance* is omitted, defaults to 'this'.

property: A property of a Movie Clip / Sprite.

Returns
The specified property value.

Description
Function: Returns the value of the specified property for the Movie Clip instancename.

Sample
The following example retrieves the horizontal axis coordinate (_x) for the Movie Clip myMovie and assigns it to the variable myMovieX:

```
myMovieX = getProperty(_root.myMovie._target,  _x);
myMovieX = getProperty("/myMovie", _x);
myMovieX = _root.myMovie.getProperty(_x);
```

See Also
getProperty() and setProperty().

4.6.8.11.146 getTimer()

Player Required
SWF4 or later

Syntax
getTimer()

Arguments
None.

Returns
As described below:

Description
Function: Returns the number of milliseconds that have elapsed since the Movie started playing.
Can be used to provide accurate timing between Events that are not dependent on the Frame Rate.

Sample
```
onFrame (20) {
    trace (getTimer()); // displays number around 1025 (frame rate was 20 fps)
}
```

4.6.8.11.147 getURL(url [, window])

Player Required
SWF4 or later

Syntax
getURL(*url {,window, method}*)

Arguments
url: Name of the URL to load (text / string).
window: Optional argument defining which Frame to load URL into. One of "_self", "_parent", "_top" or "_blank".
method: Optional argument consisting of a string defined as either "GET" or "POST" determining whether variables are retrieved using the GET or POST methods. The GET method will append the variables at the end of the URL (used for a small number of variables), and the POST method will send the variables in a separate HTTP header (used for longer strings of variables).

Returns
Nothing.

Description
Function: Loads the specified URL into the specified Frame. Note that Frame is an HTML Frame, not a Movie Frame in this situation.

If the URL specified is not an absolute reference, the Flash Player will assume a file relative to the currently playing .swf file.

Example
```
getURL("http://www.swishzone.com","_self");      // replace the current movie with the quoted URL.
```

Notes
If used in player, external browser window is opened irrespective of the window settings.
If the exported .swf is played on your local system, the flash player security settings may prevent it from opening the referenced URL. This restriction does not apply if the movie is uploaded to an external site. The problem can also be fixed by altering flash player security settings.

See Also
getURL() getURLGet() getURLPost()

4.6.8.11.148 getURLGet(url [, window])

Player Required
SWF4 or later

Syntax
getURLGet(url [,window])

Arguments
url: Name of the URL to load (text / string).
window: Optional parameter defining which Frame to load URL into. One of "_self", "_parent", "_top" or "_blank".

Returns
Nothing.

Description
Function: Loads the specified URL into the specified Frame. Note that Frame is an HTML Frame, not a Movie Frame in this situation.
Gets variables via the Get method.

If the URL specified is not an absolute reference, the Flash Player will assume a file relative to the currently playing .swf file.

Example

```
getURLGet("http://www.swishzone.com","_self");            // replace the current movie with the quoted URL.
```

Notes
If used in player, external browser window is opened irrespective of the window settings.
If the exported .swf is played on your local system, the flash player security settings may prevent it from opening the referenced URL. This restriction does not apply if the movie is uploaded to an external site. The problem can also be fixed by altering flash player security settings.

See Also
getURL() getURLGet() getURLPost()

4.6.8.11.149 getURLPost(url [, window])

Player Required
SWF4 or later

Syntax
getURLGet(url [,window])

Arguments
url: Name of the URL to load (text / string).
window: Optional parameter defining which Frame to load URL into. One of "_self", "_parent", "_top" or "_blank".

Returns
Nothing.

Description
Function: Loads the specified URL into the specified Frame. Note that Frame is an HTML Frame, not a Movie Frame in this situation.
Gets variables via the Post method.

If the URL specified is not an absolute reference, the Flash Player will assume a file relative to the currently playing .swf file.

See Also
getURL(), getURLGet() and getURLPost().

Example
getURLPost("http://www.swishzone.com","_self"); // replace the current movie with the quoted URL.

Notes
If used in player, external browser window is opened irrespective of the window settings.
If the exported .swf is played on your local system, the flash player security settings may prevent it from opening the referenced URL. This restriction does not apply if the movie is uploaded to an external site. The problem can also be fixed by altering flash player security settings.

See Also
getURL() getURLGet() getURLPost()

4.6.8.11.150 getVersion()

Identical to $version and can be used in place of $version.

4.6.8.11.151 globalToLocal()

Player Required
SWF5 or later

Syntax
myMovieClip.globalToLocal(*point*);

Arguments
point: The name or identifier of an object specifying coordinates as properties.

Returns
Nothing.

Description
Method; converts the point object from the main Stage coordinates (global) into the Movie Clip / Sprite's coordinates (local).

4.6.8.11.152 gotoAndPlay()

Player Required
SWF4 or later

Syntax
[*object.*]gotoAndPlay(*frame / label*)

Arguments
object: The name of the Movie Clip / Sprite to be played.
frame: The Frame number or label to which the playhead is sent.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead to the specified Frame in a Scene and plays from that Frame.

Sample
```
onSelfEvent (load) {
    gotoAndPlay(16); // Movie Clip / Sprite starts at Frame 16 when loaded.
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.153 gotoAndStop()

Player Required
SWF4 or later

Syntax
[*object*.]gotoAndStop(*frame / label*)

Arguments
object: The name of the Movie Clip / Sprite to be played.
frame: The Frame number or label to which the playhead is sent.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead to the specified Frame in a Scene and stops at that Frame.

Sample
```
onSelfEvent (load) {
    gotoAndStop(16); // Movie Clip / Sprite goes to Frame 16 and stops.
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.154 gotoSceneAndPlay(scene, frame)

Player Required
SWF4 or later

Syntax
gotoSceneAndPlay(*scene, frame*)

Arguments
scene: The Scene name to which the playhead is sent. This can also be a Scene name synonym.
frame: The Frame number or label to which the playhead is sent.

Returns
Nothing.

Description
Action: Sends the playhead to the specified Frame in a Scene and plays from that Frame. If no Scene is specified, the playhead goes to the specified Frame in the current Scene.

Sample
```
onSelfEvent (load) {
    gotoSceneAndPlay("Scene_2", 16);
}
```

See Also
Scene-based Movie Control
gotoSceneAndPlay(), gotoSceneAndStop(), nextSceneAndPlay(), nextSceneAndStop(), prevSceneAndPlay() and prevSceneAndStop().

Frame-based Movie Control
play(), stop(), gotoAndPlay(), gotoAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.155 gotoSceneAndStop(scene, frame)

Player Required
SWF4 or later

Syntax
gotoSceneAndStop(*scene, frame*)

Arguments
scene: The Scene name to which the playhead is sent. This can also be a Scene name synonym.
frame: The Frame number or label to which the playhead is sent.

Returns
Nothing.

Description
Action: Sends the playhead to the specified Frame in a Scene and stops at that Frame. If no Scene is specified, the playhead goes to the specified Frame in the current Scene.

Sample
```
onSelfEvent (load) {
    gotoSceneAndStop("Scene_2", 16);
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.156 hitArea

Player Required
SWF6 or later

Syntax
myMovieClip.hitArea

Arguments

None.

Returns
Nothing.

Description
Property; a reference to a Movie Clip that will be used as the hit area for a button Movie Clip. If this property is undefined the button Movie Clip itself will be used as the hit area. This property can be changed at any time.

4.6.8.11.157 hitTest()

Player Required
SWF5 or later

Syntax
myMovieClip.hitTest(*x, y {,shapeFlag}*)
myMovieClip.hitTest(*target*)

Arguments
x: The horizontal coordinate of the hit area relative to the Stage (global).

y: The vertical coordinate of the hit area relative to the Stage (global).

shapeflag: a Boolean value (true or false) telling the script whether to evaluate the shape of the specified Movie Clip (true), or just its bounding box (false). This argument can only be used if the x and y arguments are also specified. This is an optional argument and defaults to false if not specified.

target: The target path of the hit area that may come in contact with the specified Movie Clip.

Returns
true if the movie clip overlaps the specified hit area.

Description
Method; checks whether or not the specified Movie Clip comes in contact with the hit area of either the *target* or *x* and *y* coordinate.

Sample
The following samples apply to the mouse cursor touching the outer boundaries of a star shape named "star" inside a Movie Clip named "myMovieClip"

```
onEnterFrame() {
  if (myMovieClip.hitTest(_xmouse,_ymouse,true)) {
    trace("Hey, quit touching me!");
  }
}
/* will display "Hey, quit touching me!" in the debug window
whenever the mouse cursor is moved over the star shape inside 'myMovieClip' */

onEnterFrame() {
  if (myMovieClip.hitTest(_xmouse,_ymouse,false)) {
    trace("Hey, quit touching me!");
  }
}
/* will display "Hey, quit touching me!" in the debug window
whenever the mouse cursor is moved over the outer boundaries
of the Movie Clip itself (the rectangle container holding the star shape) */
```

4.6.8.11.158 if()

Player Required
SWF4 or later

Syntax
```
if(condition) {
   statement(s);
}
```

or

```
if (condition) {
   statements(s)
} else {
   statements(s)
}
```

or

```
if (condition) {
   statements(s)
} else if (condition) {
   statements(s)
} else {
   statements(s)
}
```

Arguments
condition: An expression that evaluates to true or false.
statement(s): The instructions to execute if or when the condition evaluates to true.

Returns
Nothing.

Description
Action: Evaluates a condition. If condition evaluates to true, then statements(s) within {} are executed.

Sample
```
onSelfEvent (load) {
    a = 2;
    b = 1;
    if (a > b) {
        trace("a > b");
    }
// "a > b" will be displayed in 'Debug' window. If a < b, nothing will be displayed.

onSelfEvent (load) {
    a = 5;
    b = 3;
    if (a == b) {
        trace(a add " and " add b add " are equal);
    } else {
        trace(a add " and " add b add " are not equal);
    }
}
// displays "5 and 3 are not equal" in the debug window

onSelfEvent (load) {
    a = 10;
    b = 5;
    c = 5;
    if ((a+b) == c) {
        trace("Values are equal");
    } else if ((a - b) == c) {
        trace(a add " minus " add b add " is equal to " add c);
    } else {
        trace(a add " + " add b add " is not equal to " add c);
    }
}
// displays "10 minus 5 is equal to 5" in the debug window
```

See Also
else and else if.

4.6.8.11.159 include

Player Required
Not player related

Syntax
include <*filename*>

Description
Allows inclusion of external script by reference to the name of the file containing the script.

When SWiSH Max is compiling the script, it reads script from the named file at point the point of the include.

To avoid the possibility of infinitely nested includes, a named file is only included once.
The file is assumed to exist in the current SWI file folder for files, unless the full path is quoted.

The include directive is invoked at compile time. Therefore, if you make any changes to an included file, you must save the file and recompile any SWF files that uses it.

See Also
#include

4.6.8.11.160 instanceof

Player Required
SWF5 or later

Syntax
object instanceof *class*

Arguments
object: A scripting object.
class: A reference to an Actionscript constructor class. Object, String, MovieClip, Date, Function, etc.

Returns
If the object is an instance of the specified class, instanceof will return true. If the object is not an instance of the specified class, instanceof will return false.

Description
Operator; Tests if an object is an instance of the specified class, returns true if it does and false if it does not.

Sample

```
onFrame (1) {
    today = new Date();
    if(today instanceof Date){
        trace("Yes it is");
    } else {
        trace("No it isn't");
    }
}
// The script above displays "Yes it is" in the debug window
```

4.6.8.11.161 int(value)

Player Required
SWF4 or later

Syntax
int(value)

Arguments
value: A number to be truncated to an integer.

Returns
As described below:

Description
Function: Converts a decimal number an integer value by discarding the fractional part of the number.

Note: This is an older function. *Math.round* is the preferred method to use.

Sample

```
int(3.8);      // return value is 3
int(-9.9);     // return value is -9
```

See Also
Math.floor

4.6.8.11.162 isNearTarget(target,dx,dy)

Player Required
SWF4 or later

Syntax
isNearTarget(targetname [,a] [,b])

Arguments
targetname: Name of the Movie Clip / Sprite whose proximity we are testing.
a: X near boundary or radius if parameter b omitted.
b: Y near boundary.

Returns
Boolean true if the two Objects are near each other.
Boolean false otherwise.

Description
Returns true if the current Sprite / Object / Movie Clip is 'near' the Sprite / Object / Movie Clip defined by targetname. This targetname must be defined using slash notation. The _target property of another Movie Clip / Sprite / Object is also an allowable parameter.

Calculation is defined with respect to the center points of the Movie Clip / Objects (not the anchor point).
The meaning of 'near' depends on the supplied Arguments.
If no Arguments are defined, the Objects are considered 'near' if their bounding rectangles overlap.
If only the 'a' parameter is defined, the Objects are considered 'near' if the distance between their centers is less than the radius defined by the 'a' parameter.
If 'a' and 'b' Arguments are defined, the Objects are considered 'near' if abs(x1 - x2) < a and abs(y1 - y2) < b
x1, y1 is the center point of one of the Objects, x2, y2 is the center point of the other Object.

Sample

Two shapes exist in the Movie, s1 and s2. The following script is included for s1:

```
onEnterFrame() {
    if (isNearTarget("/s2", 50, 50)) { // _root.s2._target can be used in place of "/s2"
      this._xscale = 150;     // s1 gets fat when it is near s2.
    } else {
      this._xscale = 100;
    }
}
```

See Also
isNearTarget(), isNearThis() and _target.

4.6.8.11.163 isNearThis(dx,dy)

Player Required
SWF4 or later

Syntax
MovieClipName.isNearThis([a][,b])

Arguments
MovieClipName: Name of the Movie Clip / Sprite whose proximity we are testing.
a: X near boundary or radius if parameter b omitted.
b: Y near boundary.

Returns
Boolean true if the two Objects are near each other.
Boolean false otherwise.

Description
Returns true if the current Sprite / Object / Movie Clip is 'near' the Sprite / Object / Movie Clip defined by MovieClipName.
Calculation is defined with respect to the center points of the Sprites / Objects (not the anchor point).
The meaning of 'near' depends on the supplied Arguments.
If no Arguments are defined, the Objects are considered 'near' if their bounding rectangles overlap.
If only the 'a' parameter is defined, the Objects are considered 'near' if the distance between their centers is less than the radius defined by the 'a' parameter.
If 'a′ and 'b' Arguments are defined, the Objects are considered 'near' if abs(x1 - x2) < a and abs(y1 - y2) < b
x1, y1 is the center point of one of the Objects, x2, y2 is the center point of the other Object.

Sample
Two shapes exist in the Movie, s1 and s2. The following script is included for s1:

```
onEnterFrame() {
    if (_root.s2.isNearThis(50,50)) {
      this._xscale = 150;     // s1 gets fat when it is near s2.
    } else {
      this._xscale = 100;
    }
}
```

See Also
isNearTarget(), isNearThis() and _target.

4.6.8.11.164 Key (Object)

This Object allows access to various Keyboard status functions and constants without a constructor.

Methods	**Description**
Key.getAscii()	Returns the ASCII value of the last key pressed.

Key.getCode()	Returns the virtual key code of the last key pressed.
Key.isDown()	Returns true if the key specified in the parameter is pressed.
Key.isToggled()	Returns true if the Num Lock or Caps Lock key is activated.
Key.addListener()	Adds an Event Listener object to the broadcast queue.
Key.removeListener()	Removes the named listener object from the queue.

Properties

Key.BACKSPACE	Code constant associated with the Backspace key (8).
Key.CAPSLOCK	Code constant associated with the Caps Lock key (20).
Key.CONTROL	Code constant associated with the Control key (17).
Key.DELETEKEY	Code constant associated with the Delete key (46).
Key.DOWN	Code constant associated with the Down Arrow key (40).
Key.END	Code constant associated with the End key (35).
Key.ENTER	Code constant associated with the Enter key (13).
Key.ESCAPE	Code constant associated with the Escape key (27).
Key.HOME	Code constant associated with the Home key (36).
Key.INSERT	Code constant associated with the Insert key (45).
Key.LEFT	Code constant associated with the Left Arrow key (37).
Key.PGDN	Code constant associated with the Page Down key (34).
Key.PGUP	Code constant associated with the Page Up key (33).
Key.RIGHT	Code constant associated with the Right Arrow key (39).
Key.SHIFT	Code constant associated with the Shift key (16).
Key.SPACE	Code constant associated with the Spacebar (32).
Key.TAB	Code constant associated with the Tab key (9).
Key.UP	Code constant associated with the Up Arrow key (38).

Additional key codes (PC keyboard only, may not be available or different on other platforms).
F1 returns code 112 *
F2 returns code 113
F3 returns code 114
F4 returns code 115
F5 returns code 116
F6 returns code 117
F7 returns code 118
F8 returns code 119
F9 returns code 120
F10 returns code 121 *
F11 returns code 122
F12 returns code 123
* Some Function Keys are not available as the operating system uses them to invoke special functions. eg on WinXP F1 is used to open help file. Some function keys may also have browser assigned functions and will hence not be available for use with specific browsers.
Num Lock 144
Pause / Break 19
Scroll Lock 145
Print Scrn 44

4.6.8.11.164.1 Key.getAscii()

Player Required
SWF5 or later

Syntax
Key.getAscii()

Arguments
none

Returns
The ascii code of the last pressed key.

Note: This cannot be used to check keys that do not correspond to an ascii character. eg. Shift or Caps lock. See Key.getCode() to monitor those keys.

Description
Method: Returns the ascii code of the last pressed key.

Sample
```
onEnterFrame() {
        // _root.ret.text is a dynamic text element.
        // this displays the ascii code of the last pressed key.
        _root.ret.text = Key.getAscii();
}
```

See Also
Key.getAscii(), Key.getCode()

4.6.8.11.164.2 Key.getCode()

Player Required
SWF5 or later

Syntax
Key.getCode()

Arguments
none

Returns
The keyboard code of the last pressed key.

Description
Method: Returns the keyboard code of the last pressed key.
This allows comparison / checking of keys that do not have an assigned ascii value via the other Key properties.
eg. Key.CAPSLOCK

Sample
```
onEnterFrame() {
        // _root.ret.text is a dynamic text element.
        // this displays the ascii code of the last pressed key.
        _root.ret.text = Key.getCode();
        if (Key.GetCode() == Key.CAPSLOCK) {
                // the caps lock key is pressed
        }
}
```

See Also
Key.getAscii(), Key.getCode()

4.6.8.11.164.3 Key.isDown()

Player Required
SWF5 or later

Syntax
Key.isDown(keycode)

Arguments
keycode (do not confuse with ascii code, although the two are the same for most ascii keys a..z etc.)
The keycode for non ascii keys can often be obtained via the Key properties. eg. Key.PGDN for the page down key.

Returns
The status of the nominated key.

Description
Method: Returns true if the nominated key is currently being pressed.

Sample
```
onEnterFrame() {
        // _root.ret.text is a dynamic text element.
        // this displays the status of the PGUP key
        _root.ret.text = Key.isDown(Key.PGUP);
}
```

4.6.8.11.164.4 Key.isToggled()

Player Required
SWF5 or later

Syntax
Key.isToggled(keycode)

Arguments
keycode (do not confuse with ascii code, although the two are the same for most ascii keys a..z etc.)
Either Key.CAPSLOCK (20) or 144 (Num Lock). Num Lock is not available on Macintosh.

Returns
The status of the toggling keys.

Description
Method: Returns true if the toggling key is on or selected.

Sample
```
onEnterFrame() {
        // _root.ret.text is a dynamic text element.
        // this displays the toggled status (rather than the key press status) of the CAPSLOCK key
        _root.ret.text = Key.isToggled(Key.CAPSLOCK);
}
```

4.6.8.11.165 length(string)

Player Required
SWF5 or later

Syntax
length(expression)
length(variable)
string.length

Arguments
expression: A string.
variable: The name of a variable containing a string.

Returns
length of the string defined by the expression or variable.

Description
String function: Returns the length of the specified string or variable name.

Sample

```
onSelfEvent (load) {
    s = "dude";
    trace(length(s));
}
// returns the value 4

or,

onSelfEvent (load) {
    s = "dude";
    trace(s.length);
}
// returns the value 4
```

4.6.8.11.166 lineStyle()

Player Required
SWF6 or later

Syntax
myMovieClip.lineStyle(*{thickness {,rgb {,alpha}}}*)

Arguments
thickness: integer defining the thickness of the line in points. Acceptable values are between 0 and 255. If no value is specified a line is not drawn. A value of 0 indicates a hairline thickness.
rbg: specifies the color of the fill in a hex color value (eg. Black is 0x000000, white is 0xFFFFFF, etc).
alpha: specifies the opacity of the fill color. This argument is optional and defaults to 100 (opaque) if not provided. Acceptable values are 0 through 100.

Returns
Nothing.

Description
Method; defines a line style for use with lineTo and curveTo methods.

Note: using the clear method will reset the lineStyle to an undefined state.

4.6.8.11.167 lineTo()

Player Required
SWF6 or later

Syntax
myMovieClip.lineTo(*x,y)*

Arguments
x: the horizontal position for the line to draw to
y: the vertical position for the line to draw to

Returns
Nothing.

Description
Method; draws a line from the current drawing position (set by a moveTo method) to the position defined by the x and y arguments. After the line is drawn the drawing position resets to the position set by those

arguments. Lines are drawn underneath any objects placed inside the Movie Clip / Sprite from SWiSH Max (such as images, shapes, buttons, etc.). If no drawing position is set by a moveTo method, then the drawing position defaults to 0,0. If any of the arguments are omitted, the function will fail.

See Also
beginFill()
beginGradientFill()
moveTo()
lineTo()
endFill()

Example
The following code will cause a orange square to be drawn.

```
onSelfEvent (load)
{
    this.beginFill(0xFF8800,100);
    this.moveTo(-50,-50);
    this.lineTo(-50,50);
    this.lineTo(50,50);
    this.lineTo(50,-50);
    this.endFill();
}
```

4.6.8.11.168 loadMovie(name[,variables])

Player Required
SWF4 or later

Syntax
MovieClipName.loadMovie("url"[, variables])

Arguments
MovieClipName: Name of the Movie Clip / Sprite that is loaded.

url: The absolute or relative URL of the .swf file or .jpeg file to be loaded. A relative path must be relative to the .swf file at level 0. The URL must be in the same subdomain as the URL where the Movie currently resides. For use in the Flash Player or for testing in test mode in the Flash authoring application, all .swf files must be stored in the same folder and the filenames cannot include folder or disk drive specifications.

variables: An optional parameter specifying an HTTP method for sending variables. The parameter must be the string GET or POST. If there are no variables to be sent, omit this parameter. The GET method appends the variables to the end of the URL, and is used for small numbers of variables. The POST method sends the variables in a separate HTTP header and is used for long strings of variables.

Returns
Nothing.

Description
Action and Movie Clip Method: Loads a .swf or .jpeg file into the Flash Player while the original Movie is playing. The loadMovie Action lets you display several Movies at once and switch between Movies without loading another HTML document.

Without the loadMovie Action, the Flash Player displays a single Movie (.swf file) and then closes. When you use the loadMovie Action, you must specify a target Movie Clip / Sprite, into which the Movie will load.

A Movie or image loaded into a target inherits the position, rotation, and scale properties of the targeted Movie Clip / Sprite. The upper-left corner of the loaded image or Movie aligns with the registration point of the targeted Movie Clip. Alternatively, if the target is the _root Timeline, the upper left corner of the image or Movie aligns with the upper-left corner of the stage.

Use the unloadMovie Action to remove Movies loaded with the loadMovie Action.

Note: An export setting of SWF6 version or higher is required to load a non-progressive JPEG image file.

Sample

The following loadMovie statement if added to the onSelfEvent (load) event of a Movie Clip, loads the external file into that Movie Clip when the Movie Clip is loaded.

```
onSelfEvent (load) {
    this.loadMovie("http://www.swishzone.com/script_samples/testswf.swf");
}
```

See also

loadMovie, loadMovieNum, unloadMovie, unloadMovieNum, loadVariables and loadVariablesNum.

4.6.8.11.169 loadMovieNum(name,level[,variables])

Player Required

SWF4 or later

Syntax

loadMovieNum("url",level[, variables])

Arguments

url: The absolute or relative URL of the .swf or .jpeg file to be loaded. A relative path must be relative to the .swf file at level 0. The URL must be in the same subdomain as the URL where the Movie currently resides. For use in the stand-alone Flash Player or for testing in test-movie mode in the Flash authoring application, all .swf files must be stored in the same folder; and the filenames cannot include folder or disk drive specifications.

level: An integer specifying the level in the Flash Player into which the Movie will be loaded.

variables:An optional parameter specifying an HTTP method for sending variables. The parameter must be the string GET or POST. If there are no variables to be sent, omit this parameter. The GET method appends the variables to the end of the URL and is used for small numbers of variables. The POST method sends the variables in a separate HTTP header and is used for long strings of variables.

Returns

Nothing.

Description

Action: Loads a .swf or .jpeg file into a level in the Flash Player while the originally loaded Movie is playing. The loadMovieNum Action lets you display several Movies at once and switch between Movies without loading another HTML document.

The Flash Player has a stacking order of levels starting with level 0. These levels are like layers of acetate; they are transparent except for the Objects on each level. When you use the loadMovieNum Action, you must specify a level in the Flash Player into which the Movie will load. Once a Movie is loaded into a level, you can use the syntax, _levelN, where N is the level number, to target the Movie.

When you load a Movie, you can specify any level number and you can load Movies into a level that already has a .swf file loaded into it. If you do, the new Movie will replace the existing .swf file. If you load a Movie into level 0, every level in the Flash Player is unloaded, and level 0 is replaced with the new file. The Movie in level 0 sets the Frame rate, background color and Frame size for all other loaded Movies.

The loadMovieNum Action also allows you to load .jpeg files into a Movie while it plays. For both images and .swf files, the upper-left corner of the image aligns with the upper-left corner of the stage when the file loads.

Also in both cases, the loaded file inherits rotation and scaling, and the original content is over written.

Use the unloadMovieNum Action to remove Movies or images loaded with the loadMovieNum Action.

Sample
This example loads the specified .swf into level 1 of the Player.
The URL "http://www.swishzone.com/script_samples/testswf.swf" is provided for testing purposes.

```
loadMovieNum("http://www.swishzone.com/script_samples/testswf.swf",1);
```

Note: The above script must be tested using the test in player or test in browser feature.
loadVariables / loadMovies functions cannot be debugged using standard debugging features

See also
loadMovie, loadMovieNum, unloadMovie, unloadMovieNum, loadVariables and loadVariablesNum.

4.6.8.11.170 LoadVars (Class)

The **LoadVars** class allows you to monitor download progress and verify successful completion.
This class is an alternative to the loadVariables() function.

The class has a number of methods and properties.

addRequestHeader(header:Object, headerValue:String)

Adds or changes HTTP request headers sent with POST actions.

Flash Player: 6+

Parameters

header:Object	String or array of strings that represents an HTTP request header name.
headerValue:String	String that represents the value associated with header.

contentType (Property)

Sets the MIME type that is sent to the server when LoadVars.send() or LoadVars.sendAndLoad() is called.

Flash Player: 6+

decode(queryString:String)

Flash Player: 7+

Parameters

queryString:String	URL query string containing name / value pairs.

getBytesLoaded() : Number

Returns number of bytes loaded. Returns undefined if no load operation is in progress.

Flash Player: 6+

Parameters None

---

getBytesTotal() : Number

Returns the total number of bytes to be loaded. Returns undefined if no load operation is in progress.

Flash Player: 6+

Parameters None

---

load(url:String)

Downloads variables based on the URL specified by the parameter url.

Flash Player: 6+

Parameters

url:String URL from which to download the variables.

Note: If the movie is running in a web browser, then the url must be in the same domain as the SWF file. SWF6, SWF7 and SWF8 impose different cross domain security restrictions.

---

loaded : Boolean (Property)

Indicates if load is complete. undefined if the load operation has not commenced.

Flash Player: 6+

LoadVars (constructor)

Creates the LoadVars object.

Flash Player: 6+

Example

```
var load:LoadVars = new LoadVars();
```

---

onData = function(src:String) (handler)

Referenced function is called when data is completely downloaded.

Flash Player: 6+

Parameters

src:String This is the raw data from the load operation. undefined if download data is incomplete or in error.

---

onHTTPStatus = function(status:Number) (handler)

Referenced function is called a HTTP status code is received from the server.

Flash Player: 8+

Parameters

status:Number	status code returned from the server. eg. 404 - page not found.

---

onLoad = function(success:Boolean) (handler)

Referenced function is called when the load operation has ended.

Flash Player: 6+

Parameters

success:Boolean	True if the load was successful, false otherwise.

---

send(url:String, target:String, [method:String]) : Boolean

Sends the variables in the LoadVars object using the HTTP POST method. The method can be modified to GET using the **method** parameter. The content type can be set using the **contentType** property.

Flash Player: 6+

Parameters

url:String	URL where the where the variables are uploaded.
target:String	target window name or: "_self" - current frame in current window "_blank" - opens a new window "_parent" - specifies parent window of the current frame "_top" - specifies top-level frame in the current window
method:String (optional)	Default is POST, can be used to change the method to GET by specifying "GET"
Returns	false if no parameters specified, true otherwise.

---

sendAndLoad(url:String, target:Object, [method:String]) : Boolean

Sends the variables in the LoadVars object using the HTTP POST method. The method can be modified to GET using the **method** parameter. The content type can be set using the **contentType** property. The server response is then downloaded parsed as variable data and placed into the target LoadVars or XML object.

Flash Player: 6+

Parameters

url:String	URL where the where the variables are uploaded.
target:Object	A LoadVars or XML object to receive the downloaded variables.
method:String (optional)	Default is POST, can be used to change the method to GET by specifying "GET"

Note: If the movie is running in a web browser, then the url must be in the same domain as the SWF file. SWF6, SWF7 and SWF8 impose different cross domain security restrictions.

---

toString() : String

Returns a string containing the variables in the LoadVars object.

Flash Player: 6+

Parameters None

Example

```
var lv:LoadVars = new LoadVars();
lv.name = "max";
lv.version = 2;
trace(lv.toString()); // output: version=2&name=max
```

General Example

```
onFrame (3) {
    colorVars = new LoadVars();
    colorVars.onLoad = function (success) {
        if (success) {
            gotoAndPlay("vars_loaded");
        }
    };
    colorVars.load("color_config.txt");
}
```

4.6.8.11.171 loadVariables(name[,variables])

Player Required

SWF4 or later

Syntax

MovieClipName.loadVariables ("*url*" [, *variables*])

Arguments

MovieClipName: Name of the Movie Clip / Sprite that the variables are loaded.

url: An absolute or relative URL where the variables are located. If you access the Movie using a Web browser, the host for the URL must be in the same subdomain as the Movie itself.

variables: An optional parameter specifying an HTTP method for sending variables. The parameter must be the string GET or POST. If there are no variables to be sent, omit this parameter. The GET method appends the variables to the end of the URL and is used for small numbers of variables. The POST method sends the variables in a separate HTTP header and is used for long strings of variables.

Returns

Nothing.

Description

Action and Movie Clip Method: Reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or PHP, or Perl script, and sets the values for variables in a Flash Player level or a target Movie Clip. This Action can also be used to update variables in the active Movie with new values.

The text at the specified URL must be in the standard MIME format application/x-www-form-urlencoded (a standard format used by CGI scripts). The Movie and the variables to be loaded must reside at the same subdomain.

Any number of variables can be specified. For example, the phrase below defines several variables:
name=dude&address=home&city=Sydney

When you use the loadVariables Action, you must specify a Movie Clip / Sprite target into which the variables will load.

Sample

The project contains a Movie Clip / Sprite named datetime. The datetime Movie Clip / Sprite contains two

Dynamic Text elements named date and time. The onSelfEvent (load) Event is for the datetime Movie Clip / Sprite.
The referenced URL returns a string
date=dd-mmmm-yyyy&time=hh:mm

The URL is available on our Web site for development / testing purposes.

```
onSelfEvent (load) {
    this.date = "date";
    this.time = "time";
    this.loadVariables("http://www.swishzone.com/script_samples/date.php",'GET');
}
```

The date.php Web page contains the following script:

```
<?php
        // example script that returns date and time.
        // demonstrates use of load variable function.
        echo "date=";
        print (date ("d-M-Y"));
        echo "&time=";
        print (date ("H:i"));
?>
```

See also
loadMovie LoadVars loadMovieNum ,unloadMovie ,unloadMovieNum ,loadVariables and loadVariablesNum
.

4.6.8.11.172 loadVariablesNum(name,level[,variables]

Player Required
SWF4 or later

Syntax
loadVariablesNum("url" ,level [, variables])

Arguments
url: An absolute or relative URL where the variables are located. If you access the Movie using a Web browser, the host for the URL must be in the same subdomain as the Movie itself.

level: An integer specifying the level in the Flash Player to receive the variables.

variables: An optional parameter specifying an HTTP method for sending variables. The parameter must be the string GET or POST. If there are no variables to be sent, omit this parameter. The GET method appends the variables to the end of the URL, and is used for small numbers of variables. The POST method sends the variables in a separate HTTP header and is used for long strings of variables.

Returns
Nothing.

Description
Action: Reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or PHP, or Perl script, and sets the values for variables in a Flash Player level. This Action can also be used to update variables in the active Movie with new values. When you load variables into a level, the Action in the 'Script' Panel in guided mode becomes loadVariablesNum; in expert mode, you must specify loadVariablesNum or choose it from the Add Script Menu.

The text at the specified URL must be in the standard MIME format application/x-www-form-urlencoded (a standard format used by CGI scripts). The Movie and the variables to be loaded must reside at the same subdomain.

Any number of variables can be specified. For example, the phrase below defines several variables:

name=dude&address=home&city=Sydney

When you use the loadVariablesNum Action, you must specify a Flash Player level into which the variables will load.

Sample

The project contains two Dynamic Text elements named date and time.
The onSelfEvent (load) Event is for the main page.
The referenced URL returns a string
date=dd-mmmm-yyyy&time=hh:mm

The URL is available on our Web site for development / testing purposes.

```
onSelfEvent (load) {
    date = "date";
    time = "time";
    loadVariablesNum("http://www.swishzone.com/script_samples/date.php",0,'GET');
}
```

Note: The above script must be tested using the test in player or test in browser feature. loadVariables / loadMovies functions cannot be debugged using standard debugging features

See also

loadMovie, LoadVars, loadMovieNum, unloadMovie, unloadMovieNum, loadVariables and loadVariablesNum.

4.6.8.11.173 localToGlobal()

Player Required

SWF5 or later

Syntax

myMovieClip.localToGlobal(*point*);

Arguments

point: The name or identifier of an object specifying coordinates as properties.

Returns

Nothing.

Description

Method; converts the point object from the main Movie Clip / Sprite coordinates (local) into the Stage coordinates (global).

Sample

This sample shows the local and global X/Y position of a shape named "shape1" inside the Movie Clip / Sprite named "thisMovieClip"

```
onEnterFrame() {
    location = new Object();
    location.x = thisMovieClip.shape1._X;
    location.Y = thisMovieClip.shape1._Y;
    thisMovieClip.localToGlobal(location);
    trace("Shape1's Local X location is: " add thisMovieClip.shape1._X);
    trace("Shape1's Local Y location is: " add thisMovieClip.shape1._Y);
    trace("Shape1's Global X location is: " add location.x);
    trace("Shape1's Global Y location is: " add location.y);
}
```

The local X/Y coordinates will never change - but if this Movie Clip / Sprite is being dragged by the mouse, the global X and Y coordinates will update as the position of the Movie Clip / Sprite changes in relation to the stage.

4.6.8.11.174 Math (Object)

This Object allows access to various mathematical functions and constants without a constructor.

When exporting SWF4 Movies, it is recommended that you select the Shared advanced math library option in the 'Export' Panel if you plan to use the Math Object. If this option is not selected, math functions still work, but the code is placed directly into your compiled script, which will cause your Movie to be much larger if you use math functions throughout your Movie.

The Math Object is emulated in SWiSH Max.

Function	**Requires Advanced Math Support?**	**Description**
Math.abs(number)	N	Absolute value of number
Math.acos(number)	Y	Arc cos of ratio in radians
Math.acosdeg(number)	Y	Arc cos of ratio in degrees
Math.approach(number,dest,factor)	N	Moves 'value' closer to 'dest' by a factor of 'factor'
Math.asin(number)	Y	Arc sin of ratio in radians
Math.asindeg(number)	Y	Arc sin of ratio in degrees
Math.atan(number)	Y	Arc tan of ratio in radians
Math.atandeg(number)	Y	Arc tan of ratio in degrees
Math.atan2(y,x)	Y	Arc tan (angle in radians) of specified coordinates
Math.atan2deg(y,x)	Y	Arc tan (angle in degrees) of specified coordinates
Math.ceil(number)	N	Returns closest integer >= specified number
Math.chance(percent)	N	Returns true (1) percent (on average) amount of the time

Math.clamp(number,lo,hi)	N	if (lo < number < hi) returns number
		if (number <= lo) returns lo
		if (number >= hi) returns hi

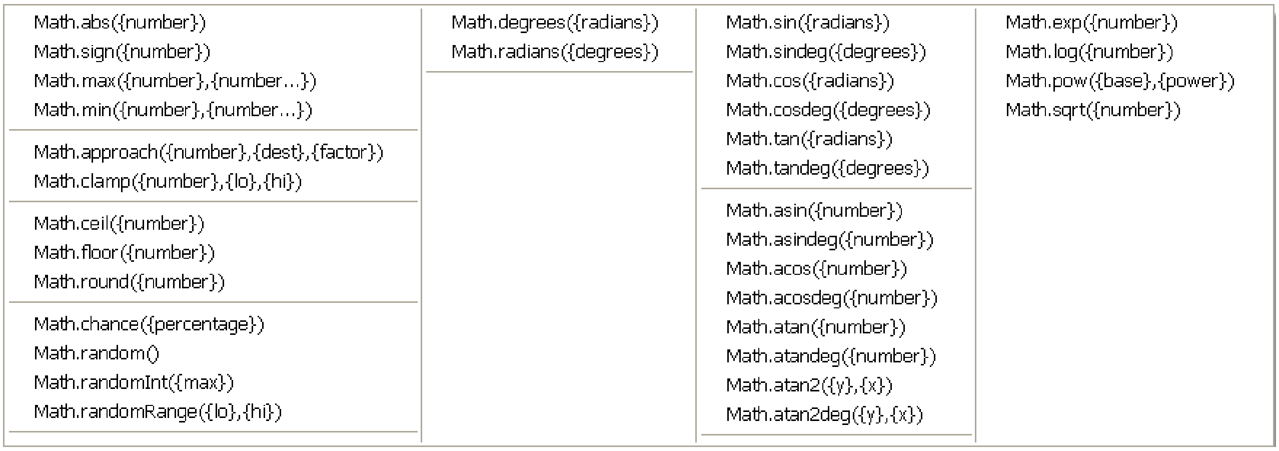
Math.cos(radians)	Y	Returns the cosine of the angle. Angle is in radians
Math.cosdeg(degrees)	Y	Returns the cosine of the angle. Angle is in degrees
Math.degrees(radians)	N	Converts an angle in radians to degrees
Math.exp(number)	Y	Raises e to the power of number
Math.floor(number)	N	Returns closest integer <= specified number
Math.log(number)	Y	Returns the natural log of number
Math.max(number,number...)	N	Returns the maximum of the supplied Arguments
Math.min(number,number...)	N	Returns the minimum of the supplied Arguments
Math.Pi	N	Constant, PI
Math.pow(base,power)	Y	Raises base to the power of power
Math.radians(degrees)	N	Converts an angle in degrees to radians
Math.random()	N	Returns a random number, 0 <= number <= 1
Math.randomInt(max)	N	Returns a random number in the range 0 <= number <= max
Math.randomRange(lo,hi)	N	Returns a random number in the range lo <= number <= hi
Math.round(number)	N	rounds the value of the parameter up or down to the nearest integer

Math.sign(number)	N	if (number < 0) returns -1
		if (number == 0) returns 0
		if (number > 0) returns +1

Math.sin(radians)	Y	Returns the sine of the angle. Angle is in radians

Math.sindeg(degrees)	Y	Returns the sine of the angle. Angle is in degrees
Math.sqrt(number)	Y	Returns the square root of number
Math.SQRT1_2	N	Constant, square root of 0.5
Math.SQRT2	N	Constant, square root of 2
Math.tan(radians)	Y	Returns the tangent of the angle. Angle is in radians
Math.tandeg(degrees)	Y	Returns the tangent of the angle. Angle is in degrees

Members of the Math Object are selectable from the following Menu.



4.6.8.11.174.1 Math.abs(number)

Player Required
SWF4 or later

Syntax
Math.abs(x)

Arguments
x: A number or expression.

Returns
A number, the absolute value of x.

Description
Method: Returns the absolute value of x.

Sample
```
Math.abs(3.4); // returns 3.4
Math.abs(-3.4);        // returns 3.4
```

4.6.8.11.174.2 Math.acos(number)

Player Required
SWF4 or later

Syntax
Math.acos(x)

Arguments
x: A number or expression. (-1 <= x <= 1).

Returns

A number, the angle (in radians) of arc cosine (x).

Description
Calculates the arc cosine (x).

Sample
```
Math.acos(0.5); // returns 1.0471975511966 (PI / 3 radians = 60 degrees)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.3 Math.acosdeg(number) +

Player Required
SWF4 or later

Syntax
Math.acosdeg(x)

Arguments
x: A number or expression. (-1 <= x <= 1).

Returns
A number, the angle (in degrees) of arc cosine (x).

Description
Calculates the arc cosine (x).

Sample
```
Math.acosdeg(0.5); // returns 60 degrees
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.4 Math.approach(number,dest,factor) +

Player Required
SWF4 or later

Syntax
Math.approach(number, dest, factor)

Arguments
All numeric numbers or expressions.
number:Current position (x or y coordinate)
dest: Desired position (x or y coordinate)
factor: Amount to approach dest.

Returns
A number, the new current position.

Description
Use to calculate 'easing' amount in scripted moves.
Calculated function is ((number - dest) * factor + dest)
If 0 < factor < 1, Object moves towards dest, slowing as it approaches.
if 1 < factor Object moves away from dest, accelerating as it leaves.

Sample
In subsequent Frames, move Object towards X = 10 by 5% each Frame

```
onEnterFrame() {
    this._X = Math.approach(this._X, 10, 0.95);
}
```

Flash MX Differences
This function is not implemented in Flash MX.

4.6.8.11.174.5 Math.asin(number)

Player Required
SWF4 or later

Syntax
Math.asin(x)

Arguments
x: A number or expression. (-1 <= x <= 1).

Returns
A number, the angle (in radians) of arc sine (x).

Description
Calculates the arc sine (x).

Sample
```
Math.asin(0.5);        // 0.523598775598299 ( PI / 6 radians = 30 degrees)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.6 Math.asindeg(number) +

Player Required
SWF4 or later

Syntax
Math.asindeg(x)

Arguments

x: A number or expression. (-1 <= x <= 1).

Returns

A number, the angle (in degrees) of arc sine (x).

Description

Calculates the arc sine (x).

Sample

```
Math.asindeg(0.5);     // returns 30 degrees
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences

This function is not implemented in Flash MX.

See Also

Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.7 Math.atan(number)

Player Required

SWF4 or later

Syntax

Math.atan(x)

Arguments

x: A number or expression.

Returns

A number, the angle (in radians) of arc tan (x).

Description

Calculates the arc tan (x).

Sample

```
Math.atan(1); // returns 0.785398163397448. (Pi/4 radians = 45 degrees)
```

Note: Requires Advanced Math Support. See here for more information

See Also

Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.8 Math.atandeg(number) +

Player Required

SWF4 or later

Syntax

Math.atandeg(x)

Arguments

x: A number or expression.

Returns
A number, the angle (in degrees) of arc tan (x).

Description
Calculates the arc tan (x).

Sample

```
Math.atandeg(1); // returns 45 degrees
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(),
Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(),
Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.9 Math.atan2(y,x)

Player Required
SWF4 or later

Syntax
Math.atan2(y,x)

Arguments
x, y: Numbers or expressions.

Returns
A number, the angle (in radians) of arc tan (y/x).

Description
Calculates the arc tan (y/x).
Handles the special cases where y = 0 or x = 0. Although x = y = 0 is undefined, function returns 0.

Sample

```
Math.atan2(1,1); // returns 0.785398163397448. (Pi/4 radians = 45 degrees)
Math.atan2(1,0); // returns 1.5707963267949 (Pi/2 radians = 90 degrees)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(),
Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(),
Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.10 Math.atan2deg(y,x) +

Player Required
SWF4 or later

Syntax

Math.atan2deg(y,x)

Arguments
x, y: Numbers or expressions.

Returns
A number, the angle (in degrees) of arc tan (y/x).

Description
Calculates the arc tan (y/x).
Handles the special cases where y = 0 or x = 0. Although x = y = 0 is undefined, function returns 0.

Sample

```
Math.atan2deg(1,1); // returns 45 degrees
Math.atan2deg(1,0); // returns 90 degrees
```

Note: Requires Advanced Math Support. See here for more information.

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.acosdeg(), Math.asin(), Math.asindeg(), Math.atan(), Math.atandeg(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan(), Math.tandeg(), Math.atan2(y,x) and Math.atan2deg(y,x).
Simple Trigonometry

4.6.8.11.174.11 Math.ceil(number)

Player Required
SWF4 or later

Syntax
Math.ceil(x)

Arguments
x: A number or expression.

Returns
A number, the closest integer that is greater than or equal to x.

Description
Returns the ceiling of the specified number or expression. The ceiling of a number is the closest integer that is greater than or equal to the number.

Sample

```
Math.ceil(4.2);        // returns 5
Math.ceil(-3.8);       // returns -3
```

See Also
Math.floor() and Math.round().

4.6.8.11.174.12 Math.chance(percent) +

Player Required
SWF4 or later

Syntax
Math.chance(percent)

Arguments
The percentage of time that the function should return 1 (or true).

Returns
0 or 1.

Description
Returns 1 or 0 depending on random chance.
On average, 1 is returned *per cent* of the time.

Sample

```
Math.randomInt(25);    // will return 1 on 25% of occasions. i.e. 1 in 4 chance
```

Flash MX Differences
This function is not implemented in Flash MX.

See also
Math.random(), Math.randomInt() and Max.randomRange().

4.6.8.11.174.13 Math.clamp(number,lo,hi) +

Player Required
SWF4 or later

Syntax
Math.clamp(number, lo, hi)

Arguments
All are numbers or expressions.
number: Value to be clamped.
lo: Low clamp value.
hi: Hi clamp value.

Returns
Clamped value.

Description
If number <= lo returns lo; If number >= hi returns hi; otherwise returns number.

This is a convenient way of limiting values to a set boundary.

Sample

```
Math.clamp(-5,2,15);   // returns 2
Math.clamp(4.6,2,15);  // returns 4.6
Math.clamp(103,2,15);  // returns 15
```

Flash MX Differences
This function is not implemented in Flash MX.

4.6.8.11.174.14 Math.cos(radians)

Player Required
SWF4 or later

Syntax
Math.cos(x)

Arguments
x: A number or expression representing the angle in radians.

Returns
A number, the cosine of (x).

Description
Calculates the cosine of x. This will be a value from -1.0 to 1.0.

Sample

```
Math.cos(Math.PI / 2); // returns 6.12303176911188e-17 (pretty close to 0, cos 90deg = 0)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().
Simple Trigonometry

4.6.8.11.174.15 Math.cosdeg(degrees) +

Player Required
SWF4 or later

Syntax
Math.cosdeg(x)

Arguments
x: A number or expression representing the angle in degrees.

Returns
A number, the cosine of (x).

Description
Calculates the cosine of x. This will be a value from -1.0 to 1.0.

Sample

```
Math.cosdeg(90); // returns 6.12303176911188e-17 (pretty close to 0, cos 90deg = 0)
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().
Simple Trigonometry

4.6.8.11.174.16 Math.degrees(radians) +

Player Required
SWF4 or later

Syntax
Math.degrees(x)

Arguments
x: A number or expression representing the angle in radians.
2 * PI radians = 360 degrees.

Returns

The corresponding number of degrees.

Description
Calculates the number of degrees based on the formula:
degrees = radians * 360 / (2 * pi).

Sample

```
Math.degrees(Math.PI); // returns 180 degrees
```

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().

4.6.8.11.174.17 Math.distance(x1,y1,x2,y2) +

Player Required
SWF4 or later

Syntax
Math.distance(x1,y1,x2,y2)

Arguments
x1, y1, x1, y2: Numbers or expressions.

Returns
A number, the distance between the two points (x1, y1) and (x2, y2).

Description
Calculates the distance between the two points (x1, y1) and (x2, y2) using Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)).

Sample

```
Math.distance(1,1,10,10); // returns 12.727
Math.distance(3,5,30,40); // returns 44.204
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.distanceSq()

4.6.8.11.174.18 Math.distanceSq(x1,y1,x2,y2) +

Player Required
SWF4 or later

Syntax
Math.distanceSq(x1,y1,x2,y2)

Arguments
x1, y1, x1, y2: Numbers or expressions.

Returns
A number, the square of the distance between the two points (x1, y1) and (x2, y2).

Description
Calculates the square of the distance between the two points (x1, y1) and (x2, y2) using

((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)).

Sample

```
Math.distanceSq(1,1,10,10); // returns 162
Math.distanceSq(3,5,30,40); // returns 1954
```

Note: Requires Advanced Math Support. See here for more information

See Also

Math.distance()

4.6.8.11.174.19 Math.exp(number)

Player Required

SWF4 or later

Syntax

Math.exp(x)

Arguments

x: The exponent - a number or expression.

Returns

A number.

Description

Returns the value e, to the power of the exponent specified in the parameter x.

e is a constant and is the value of the base of the natural logarithm.
e has an approximate value of 2.71828

Sample

```
Math.exp(2);   // returns e squared (or e * e), which is approximately 7.3890
```

Note: Requires Advanced Math Support. See here for more information

See Also

Math.log(), Math.log10(), Converting Log Base
Math.pow() and Math.sqrt().

4.6.8.11.174.20 Math.floor(number)

Player Required

SWF4 or later

Syntax

Math.floor(x)

Arguments

x: A number or expression.

Returns

A number, the closest integer that is less than or equal to x.

Description

Returns the floor of the specified number or expression. The floor of a number is the closest integer that is less than or equal to the number.

Sample

```
Math.floor(4.8);       // returns 4
Math.floor(-3.2);      // returns -4
```

See Also
Math.ceil() and Math.round().

4.6.8.11.174.21 Math.log(number)

Player Required
SWF4 or later

Syntax
Math.log(x)

Arguments
x: A number or expression > 0.

Returns
A number.

Description
Returns the natural logarithm of the parameter x.

Sample
```
Math.log(2);   // returns natural log of 2, approximately 0.6931
Math.log(100)/Math.log(10);   // returns 2, the log, base 10 of 100
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.exp(), Math.log10(), Converting Log Base
Math.pow() and Math.sqrt().

4.6.8.11.174.22 Math.log10(x) +

Player Required
SWF4 or later

Syntax
Math.log(x)

Arguments
x: A number or expression > 0.

Returns
A number.

Description
Returns the log base 10 of x.

Sample
```
Math.log10(2); // returns 1, the log base 10 of 10
Math.log10(100); // returns 2, the log base 10 of 100
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.exp(), Math.log(), Converting Log Base
Math.pow() and Math.sqrt().

4.6.8.11.174.23 Math.max(number,number...) +

Player Required
SWF4 or later

Syntax
Math.max(n1, n2[, n3 ....])

Arguments
n1: A number or expression.
n2: A number or expression.
n3: A list of numbers or expressions.

Returns
A number, the parameter with the highest value.

Description
Returns the value of the parameter with the highest value.

Sample

```
Math.max(1,4,-3);     // returns 4
```

Flash MX Differences
Under MX, the method is limited to two Arguments.

See Also
Math.max() and Math.min().

4.6.8.11.174.24 Math.min(number,number...) +

Player Required
SWF4 or later

Syntax
Math.min(n1, n2[, n3 ....])

Arguments
n1: A number or expression.
n2: A number or expression.
n3: A list of numbers or expressions.

Returns
A number, the parameter with the lowest value.

Description
Returns the value of the parameter with the lowest value.

Sample

```
Math.min(1,4,-3);     // returns -3
```

Flash MX Differences
Under MX, the method is limited to two Arguments.

See Also
Math.max() and Math.min().

4.6.8.11.174.25 Math.PI

Player Required
SWF4 or later

Syntax
Math.PI

Arguments
None.

Returns
The constant PI.

Description
Returns the mathematical constant PI.
PI is the ratio of the circumference of a circle to its diameter.
This is an irrational number and has the approximate value of 3.141592.

Sample

```
rad = deg * 2 * Math.PI / 360;        // converts deg to radians
```

4.6.8.11.174.26 Math.pow(base,power)

Player Required
SWF4 or later

Syntax
Math.pow(base,power)

Arguments
base, power: Numbers or expressions.
If base < 0, then power must be an integer value.

Returns
A number.

Description
Returns *base* to the power of *power.*

Samples

```
Math.pow(3,2); // returns 9 (3 * 3)
Math.pow(2,3); // returns 8 (2 * 2 * 2)
Math.pow(-2,3);        // returns -8
Math.pow(2,-1);        // returns 0.5 (1/2)
Math.pow(2,0.5);       // returns 1.414213... (square root of 2)
Math.pow(2,-0.5);      // reutrns 0.707... (1/(square root of 2)
Math.pow(-2,0.5);      // returns 0. (Error condition, base <0, non integer power)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.exp(), Math.log(), Math.log10(), Converting Log Base
Math.pow() and Math.sqrt().

4.6.8.11.174.27 Math.radians(degrees) +

Player Required
SWF4 or later

Syntax

Math.radians(x)

Arguments
x: A number or expression representing the angle in degrees.
2 * PI radians = 360 degrees.

Returns
The corresponding number of degrees.

Description
Calculates the number of radians based on the formula:
radians = degrees * 2 * pi / 360.

Sample

```
Math.radians(360);     // returns 2 * pi radians, (6.28318548202515)
```

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().

4.6.8.11.174.28 Math.random()

Player Required
SWF4 or later

Syntax
Math.random()

Arguments
None.

Returns
A random number.

Description
Returns a random number n, where 0 <= n < 1.
n has a maximum precision of 5 decimal places.

See also
Math.chance(), Math.random(), Math.randomInt() and Max.randomRange().

4.6.8.11.174.29 Math.randomInt(max) +

Player Required
SWF4 or later

Syntax
Math.randomInt(max)

Arguments
max: An integer number or expression > 0.

Returns
A random integer.

Description

Returns a random integer n, where 0 <= n < max.

Sample

```
Math.randomInt(5);     // will randomly return one of the values 0, 1, 2, 3 or 4
```

Flash MX Differences
This function is not implemented in Flash MX.

See also
Math.chance(), Math.random(), Math.randomInt() and Max.randomRange().

4.6.8.11.174.30 Math.randomRange(lo,hi) +

Player Required
SWF4 or later

Syntax
Math.randomRange(lo, hi)

Arguments
Both Arguments are numbers or expressions.
lo: The low value of the range.
hi: The high value of the range.

Note: lo < hi

Returns
A random number.

Description
Returns a random number n, where lo <= n < hi.
Maximum precision of 5 decimal places.

Sample

```
Math.randomRange(-10,10);     // will return a random value in the range -10 to 10
```

Flash MX Differences
This function is not implemented in Flash MX.

See also
Math.chance(), Math.random(), Math.randomInt() and Max.randomRange().

4.6.8.11.174.31 Math.round(number)

Player Required
SWF4 or later

Syntax
Math.round(x)

Arguments
x: A number or expression.

Returns
An integer.

Description
Rounds the value of the parameter x up or down to the nearest integer and returns the value.

Sample

```
Math.round(4.8);              // returns the value 5
Math.round(4.2);              // returns the value 4
Math.round(-4.8);      // returns the value -5
```

See Also

Math.ceil(), Math.floor() and Math.round().

4.6.8.11.174.32 Math.sign(number) +

Player Required

SWF4 or later

Syntax

Math.sign(number)

Arguments

All numeric numbers or expressions.
number: Value

Returns

The sign of the value.

Description

if (number < 0) returns -1.
if (number == 0) returns 0.
if (number > 0) returns +1.

Sample

```
Math.sign(3.8);        // returns +1
Math.sign(0);  // returns 0
Math.sign(-9); // returns -1
```

Flash MX Differences

This function is not implemented in Flash MX.

4.6.8.11.174.33 Math.sin(radians)

Player Required

SWF4 or later

Syntax

Math.sin(x)

Arguments

x: A number or expression representing the angle in radians.

Returns

A number, the sine of (x).

Description

Calculates the sine of x. This will be a value from -1.0 to 1.0.

Sample

```
Math.sin(Math.PI / 2); // returns 1 (sin 90deg = 1)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().
Simple Trigonometry

4.6.8.11.174.34 Math.sindeg(degrees) +

Player Required
SWF4 or later

Syntax
Math.sindeg(x)

Arguments
x: A number or expression representing the angle in degrees.

Returns
A number, the sine of (x).

Description
Calculates the sine of x. This will be a value from -1.0 to 1.0.

Sample
```
Math.sindeg(90);       // returns 1 (sin 90deg = 1)
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg(), Math.tan() and Math.tandeg().
Simple Trigonometry

4.6.8.11.174.35 Math.sqrt(number)

Player Required
SWF4 or later

Syntax
Math.sqrt(number)

Arguments
number: Number or expression.
number must be >= 0.

Returns
A number.

Description
Returns square root of the number.

Samples
```
Math.sqrt(3);  // returns 1.732050807...
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.exp(), Math.log(), Math.log10(), Converting Log Base

Math.pow() and Math.sqrt().

4.6.8.11.174.36 Math.SQRT1_2

Player Required
SWF4 or later

Syntax
Math.SQRT1_2

Arguments
None.

Returns
Square root of 1/2.

Description
Returns the square root of 1/2. This constant is often used in trig calculations.
It is an irrational number and has the approximate value of 0.707106.

Sample

```
Math.SQRT1_2;  // returns 0.707106
Math.sindeg(45) - Math.SQRT1_2;      // returns -1.11022302462515e-16 which is pretty close to 0
```

See Also
Math.SQRT2

4.6.8.11.174.37 Math.SQRT2

Player Required
SWF4 or later

Syntax
Math.SQRT2

Arguments
None.

Returns
Square root of 2.

Description
Returns the square root of 2. This constant is often used in trig calculations.
It is an irrational number and has the approximate value of 1.414213.

Sample

```
Math.SQRT2;    // returns 1.414213
Math.cosdeg(45) - 1/Math.SQRT2;      // returns 1.11022302462515e-16 which is pretty close to 0
```

See Also
Math.SQRT1_2

4.6.8.11.174.38 Math.tan(radians)

Player Required
SWF4 or later

Syntax
Math.tan(x)

Arguments

x: A number or expression representing the angle in radians.

Returns
A number, the tangent of (x).

Description
Calculates the tangent of x.

Sample
```
Math.tan(Math.PI / 4); // returns 1 (tan 45deg = 1)
```

Note: Requires Advanced Math Support. See here for more information

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg() and Math.tandeg().
Simple Trigonometry

4.6.8.11.174.39 Math.tandeg(degrees) +

Player Required
SWF4 or later

Syntax
Math.tandeg(x)

Arguments
x: A number or expression representing the angle in degrees.

Returns
A number, the tangent of (x).

Description
Calculates the tangent of x.

Sample
```
Math.tandeg(45);       // returns 1 (tan 45deg = 1)
```

Note: Requires Advanced Math Support. See here for more information

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Math.acos(), Math.asin(), Math.atan(), Math.cos(), Math.cosdeg(), Math.sin(), Math.sindeg() and Math.tan().
Simple Trigonometry

4.6.8.11.175 maxscroll

Player Required
SWF6 or later

Syntax
variable_name.maxscroll

Description
A read-only property associated with Script Text Fields.
Indicates the line number of the top-most visible line of text in a Text Field when the bottom-most line in the field is also visible.

The maxscroll property works with the scroll property to control the display of information in a Text Field.

Note: This property is not available with simple Text Field Objects

See also
scroll, Text Field Properties and Methods and TextField Access Matrix for more information about the properties that can be accessed.

4.6.8.11.176 Mouse Events

Event Handling procedures exist for Movie Clip / Sprites / Objects (onSelfEvent()) and for Button Objects (on()). The exact Event being handled is defined by the Event name supplied to the onSelfEvent() or on() functions.

In Windows-based computers that support left and right mouse buttons, only the left mouse button is monitored (the right mouse button is not available on Mac computers).

Event	Description
dragOut	Mouse button is pressed then mouse is moved away from the Object / Movie Clip / button
dragOver	Mouse button is pressed over the Object / Movie Clip / button. The mouse is then moved away and back while the button is pressed
keyPress(key)	The specified key has been pressed
press	Mouse button has been pressed over the Object / Movie Clip / button
release	Mouse button has been released over the Object / Movie Clip / button
releaseOutside	Mouse button has been released outside the Object / Movie Clip / button
rollOut	Mouse is moved away from the Object / Movie Clip / button
rollOver	Mouse is moved over the Object / Movie Clip / button

Event Listener routines can also be added / removed via the addListener() and removeListener() methods.

4.6.8.11.176.1 dragOut

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires:
1. The mouse button to be pressed while the mouse is positioned over the Object / Movie Clip / Sprite / button
2. The mouse to be moved outside of the Object / Movie Clip / Sprite / button while the button is held down.

The condition occurs as the mouse moves outside of the Object / Movie Clip / Sprite / button.

Sample
The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets smaller after a dragOut Event.

```
onSelfEvent (dragOut) {
    _xscale -= 10;
    _yscale -= 10;
}
```

See Also

dragOver, keyPress(key), press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.2 dragOver

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires:
1. The mouse button to be pressed while the mouse is positioned over the Object / Movie Clip / Sprite / button
2. The mouse to be moved outside the Object / Movie Clip / Sprite / button while the button is held down
3. The mouse to be moved over the Object / Movie Clip / Sprite / button while the button is still held down.

The condition occurs as the mouse returns over of the Object / Movie Clip / Sprite / button.

Sample
The following event script could be added to an Object / Movie Clip / Sprite or button so that it gets larger after a dragOver Event.

```
onSelfEvent (dragOver) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also
dragOut, keyPress(key), press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.3 keyPress(key)

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires the specified button to be pressed.

Apart from standard keyboard characters, non-printable characters can be selected via the pull-down Menu. The non-printable characters that are supported are:

Symbol	Entered as	Description
Space	"<Space>"	Space character
Left	"<Left>"	Left arrow
Right	"<Right>"	Right arrow
Up	"<Up>"	Up arrow
Down	"<Down>"	Down arrow
Home	"<Home>"	Home key
End	"<End>"	End key

PageUp	"<PageUp>"	Page Up key
PageDown	"<PageDown>"	Page Down key
Insert	"<Insert>"	Insert key
Delete	"<Delete>"	Del / Delete key
Backspace	"<Backspace>"	Backspace key
Tab	"<Tab>"	Tab key
Enter	"<Enter>"	Enter key
Escape	"<Escape>"	Esc key

Note: It is possible that the flash player will notify a different object about the event. See here for additional information.

Sample

The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets larger after the button 'a' is pressed.

```
onSelfEvent (keyPress("a")) {
    _xscale += 10;
    _yscale += 10;
}
```

If multiple characters are to be tested, it is necessary to use multiple Event Handling functions.

```
onSelfEvent (keyPress("a")) {
    _xscale += 10;
    _yscale += 10;
}
onSelfEvent (keyPress("<Tab>")) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also

dragOut, dragOver, press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.4 press

Player Required

Supported Internally

Syntax

A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description

The associated Event is executed when the Event condition is met.

The event condition requires the mouse button to be pressed while the mouse is positioned over the Object / Movie Clip / Sprite / button.

The condition occurs as the button is pressed.

Sample

The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets smaller after a press Event.

```
onSelfEvent (press) {
    _xscale -= 10;
    _yscale -= 10;
}
```

See Also
dragOut, dragOver, keyPress(key), release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.5 release

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires the mouse button to be released while the mouse is positioned over the Object / Movie Clip / Sprite / button.

The condition occurs as the button is released.

Sample
The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets larger after a release Event.

```
onSelfEvent (release) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also
dragOut, dragOver, keyPress(key), press, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.6 releaseOutside

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires:
1. The mouse button to be pressed while the mouse is positioned over the Object / Movie Clip / Sprite / button
2. The mouse button to be released while the mouse is positioned outside the Object / Movie Clip / Sprite / button.

The condition occurs as the button is released.

Sample
The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets larger after a releaseOutside Event.

```
onSelfEvent (releaseOutside) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also
dragOut, dragOver, keyPress(key), press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.176.7 rollOut

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event handling functions.

Description
The associated event is executed when the event condition is met.

The event condition requires:
1. The mouse is positioned over the object / Movie Clip / Sprite / button.
2. The mouse is positioned outside the object / Movie Clip / Sprite / button.

The condition occurs as the mouse is no longer over the object / Movie Clip / Sprite / button.

Sample
The following event script could be added to an object / Movie Clip / Sprite or button so that it gets larger after a rollOut Event.

```
onSelfEvent (rollOut) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also
dragOut, dragOver ,keyPress(key) ,press ,release ,releaseOutside ,rollOut ,rollOver ,on() and onSelfEvent()

4.6.8.11.176.8 rollOver

Player Required
Supported Internally

Syntax
A Mouse Event Parameter. Used as a parameter to on() and onSelfEvent() Event Handling functions.

Description
The associated Event is executed when the Event condition is met.

The Event condition requires:
1. The mouse is positioned outside the Object / Movie Clip / Sprite / button
2. The mouse is positioned over the Object / Movie Clip / Sprite / button.

The condition occurs as the mouse is moved over the Object / Movie Clip / Sprite / button.

Sample
The following Event script could be added to an Object / Movie Clip / Sprite or button so that it gets larger after a rollOver Event.

```
onSelfEvent (rollOver) {
    _xscale += 10;
    _yscale += 10;
}
```

See Also
dragOut, dragOver, keyPress(key), press, release, releaseOutside, rollOut, on() and onSelfEvent().

4.6.8.11.177 Mouse.hide()

Player Required
SWF5 or later

Syntax
Mouse.hide()

Returns
An integer, 0 or 1.
0 if the mouse pointer was already hidden
1 if the mouse pointer was not hidden

Description
Action: Hides the mouse pointer while inside the SWF file.
Note: Only supported in SWF5+

Sample
```
onFrame(5) {
    Mouse.hide(); // Hides the cursor when the playhead reaches Frame 5
}
```

See Also
Mouse.show

4.6.8.11.178 Mouse.show()

Player Required
SWF5 or later

Syntax
Mouse.show()

Returns
An integer, 0 or 1.
0 if the mouse pointer was already hidden
1 if the mouse pointer was not hidden

Description
Action: Displays the mouse pointer while inside the SWF file.
Note: Only supported in SWF5+

Sample
```
onSelfEvent(rollover) {
    Mouse.show(); // Displays the cursor when the mouse is moved on top of this object
}
```

See Also
Mouse.hide

4.6.8.11.179 moveTo()

Player Required
SWF6 or later

Syntax
myMovieClip.moveTo(*x,y)*

Arguments
x: the horizontal position that the drawing position is moved to.

y: the vertical position that the drawing position is moved to.

Returns

Nothing.

Description

Method; repositions the drawing position to the coordinates specified in the *x* and *y* arguments. If these arguments are omitted, the function will fail.

See Also

beginFill()
beginGradientFill()
moveTo()
lineTo()
endFill()

Example

The following code will cause a orange square to be drawn.

```
onSelfEvent (load)
{
    this.beginFill(0xFF8800,100);
    this.moveTo(-50,-50);
    this.lineTo(-50,50);
    this.lineTo(50,50);
    this.lineTo(50,-50);
    this.endFill();
}
```

4.6.8.11.180 Movie Clip / Sprite (Object)

Note: Sprite is a deprecated SWiSH term for Movie Clip.

Movie Clips support both properties and methods. Events associated with a Movie Clip are handled by the associated Event Handling or Event Method routines. A list of supported items can be found below.

The property can be accessed via *MovieClipName.property,* where *MovieClipName* is the name of the Movie Clip and *property* is the property being accessed.

The methods can be applied via *MovieClipName.method(),* where *MovieClipName* is the name of the Movie Clip and *method()* is the method being applied.

The synonym '*this*' can be used to mean the current Movie Clip / Object where scripting is used within the Movie Clip.
If the synonym '*this*' is omitted, it is assumed when valid property names or methods are specified.

e.g. this._X = 5; is equivalent to _X = 5; if the script is contained within the Movie Clip Object.

The list below defines the Sprite / MovieClip Methods and Properties and their corresponding player support.

Method	**SWF Version**
MovieClip._alpha	SWF4+
MovieClip.attachMovie	SWF5+
MovieClip.beginFill	SWF6+
MovieClip.beginGradientFill	SWF6+
MovieClip.clear	SWF6+
MovieClip.createEmptyMovieClip	SWF6+
MovieClip.createTextField	SWF6+

MovieClip._currentframe	SWF4+
MovieClip.curveTo	SWF6+
MovieClip._droptarget	SWF4+
MovieClip.duplicateMovieClip	SWF4+
MovieClip.enabled	SWF6+
MovieClip.endFill	SWF6+
MovieClip.focusEnabled	SWF6+
MovieClip._focusrect	SWF6+
MovieClip._framesloaded	SWF4+
MovieClip.getBounds	SWF5+
MovieClip.getBytesLoaded	SWF6+
MovieClip.getBytesTotal	SWF5+
MovieClip.getDepth	SWF6+
MovieClip.getURL	SWF4+
MovieClip.globalToLocal	SWF5+
MovieClip.gotoAndPlay	SWF4+
MovieClip.gotoAndStop	SWF4+
MovieClip._height	SWF4+
MovieClip._highquality	SWF4+
MovieClip.hitArea	SWF6+
MovieClip.hitTest	SWF5+
MovieClip.lineStyle	SWF6+
MovieClip.lineTo	SWF6+
MovieClip.loadMovie	SWF4+
MovieClip.loadVariables	SWF4+
MovieClip.localToGlobal	SWF5+
MovieClip.moveTo	SWF6+
MovieClip._name	SWF4+
MovieClip.nextFrame	SWF4+
MovieClip._parent	SWF4+
MovieClip.play	SWF4+
MovieClip.prevFrame	SWF4+
MovieClip._quality	SWF5+
MovieClip.removeMovieClip	SWF4+
MovieClip._rotation	SWF4+
MovieClip.setMask	SWF6+
MovieClip._soundbuftime	SWF4+
MovieClip.startDrag	SWF4+
MovieClip.stop	SWF4+
MovieClip.stopDrag	SWF4+
MovieClip.swapDepths	SWF5+
MovieClip.tabChildren	SWF6+
MovieClip.tabEnabled	SWF6+
MovieClip.tabIndex	SWF6+
MovieClip._target	SWF4+
MovieClip._totalframes	SWF4+
MovieClip.trackAsMenu	SWF6+
MovieClip.unloadMovie	SWF4+
MovieClip._url	SWF4+
MovieClip.useHandCursor	SWF6+
MovieClip._visible	SWF4+
MovieClip._width	SWF4+

MovieClip._x	SWF4+
MovieClip._xmouse	SWF6+
MovieClip._xscale	SWF4+
MovieClip._y	SWF4+
MovieClip._ymouse	SWF6+
MovieClip._yscale	SWF4+

4.6.8.11.180.1 Movie Clip Properties

Movie Clip / Sprites inherit the standard Scripting Object properties.

4.6.8.11.180.2 Movie Clip Methods

Method	Description
duplicateSprite duplicateMovieClip	Creates an Instance of a Movie Clip / Sprite while the Movie is playing
getURL	This Actions loads a document from the specified URL into a browser window
gotoAndPlay	Sends the playhead to the specified Frame in a Scene and plays from that Frame
gotoAndStop	Sends the playhead to the specified Frame in a Scene and stops at that Frame
loadMovie	Loads a .swf or .jpeg file into the Flash Player while the original Movie is playing
loadVariables	Reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or PHP, or Perl script and sets the values for variables in a Flash Player level or a target Movie Clip
nextFrameAndPlay	Sends the playhead to the next Frame in a Scene and plays from that Frame
nextFrameAndStop	Sends the playhead to the next Frame in a Scene and stops at that Frame
play	Moves the playhead forward in the Timeline
prevFrameAndPlay	Sends the playhead to the previous Frame in a Scene and plays from that Frame
prevFrameAndStop	Sends the playhead to the previous Frame in a Scene and stops at that Frame
removeMovieClip removeSprite	Removes an Instance of a Movie Clip / Sprite created with the duplicateMovieClip() or duplicateSprite() method
skipFrameAndPlay	Sends the playhead ahead n Frames and plays from that Frame
skipFrameAndStop	Sends the playhead ahead n Frames and stops at that Frame
startDragLocked	Makes the target Movie Clip / Sprite / Object draggable while the Movie is playing The Movie Clip / Sprite / Object is locked to the center of the mouse position
startDragUnlocked	Makes the target Movie Clip / Sprite / Object draggable while the Movie is playing. The Movie Clip / Sprite / Object is locked to mouse position where the user first pushed the mouse button
stop	Stops the playhead at the current position
unloadMovie	Removes a loaded Movie or a Movie Clip from the Flash Player

4.6.8.11.181 NaN

Player Required
SWF5 or later

Syntax
NaN

Arguments
None.

Description
Constant representing the IEEE-754 value for NaN "Not a Number". To determine if a number is NaN use isNan()

4.6.8.11.182 new

Player Required
SWF5 or later

Syntax
new *constructor*

Arguments
constructor; A constructor (an object type) - such as an Array object, a Date object, a Boolean object, an Object object, a Number object, etc. The constructor would be followed by any optional arguments for that specific constructor.

Returns
Nothing.

Description
Operator; used to create a new constructor.

4.6.8.11.183 newline

Player Required
SWF4 or later

Syntax
newline

Arguments
None.

Returns
Nothing.

Description
Constant: Inserts a carriage return character (\r) that inserts a blank line into the Actionscript code. Use newline to make space for information that is retrieved by a function or Action in your code.

4.6.8.11.184 nextFrameAndPlay()

Player Required
SWF4 or later

Syntax
[*object.*]nextFrameAndPlay()

Arguments
object: The name of the Movie Clip / Sprite to be played.

Returns
Nothing.

Description
Action and Movie Clip / Sprite Method: Sends the playhead to the next Frame in a Scene and plays from that Frame.

Sample
```
onSelfEvent (load) {
    nextFrameAndPlay();  // Movie Clip / Sprite starts playing from second Frame
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.185 nextFrameAndStop()

Player Required
SWF4 or later

Syntax
[*object.*]nextFrameAndStop()

Arguments
object: The name of the Movie Clip / Sprite to be played.

Returns
Nothing.

Description
Action and Movie Clip / Sprite Method: Sends the playhead to the next Frame in a Scene and stops at that Frame.

Sample
```
onSelfEvent (load) {
    nextFrameAndStop(); // Movie Clip / Sprite moves playhead to 2nd Frame and stops
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.186 nextSceneAndPlay() +

Player Required
SWF4 or later

Syntax
nextSceneAndPlay()

Arguments
None.

Returns
Nothing.

Description
Action: Sends the playhead to the next Scene and plays from the first Frame.

Sample
```
onFrame (10) {
    nextSceneAndPlay(); // identical to gotoSceneAndPlay("<next scene>", 1);
}
```

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Scene-based Movie Control
gotoSceneAndPlay(), gotoSceneAndStop(), nextSceneAndStop(), prevSceneAndPlay() and prevSceneAndStop().

Frame-based Movie Control
play(), stop(), gotoAndPlay(), gotoAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.187 nextSceneAndStop() +

Player Required
SWF4 or later

Syntax
nextSceneAndStop()

Arguments
None.

Returns
Nothing.

Description
Action: Sends the playhead to the next Scene and stops at the first Frame.

Sample
```
onFrame (10) {
    nextSceneAndStop(); // identical to gotoSceneAndStop("<next scene>", 1);
}
```

Flash MX Differences
This function is implemented in MX as prevScene().

See Also
Scene-based Movie Control
gotoSceneAndPlay(), gotoSceneAndStop(), nextSceneAndPlay(), prevSceneAndPlay() and prevSceneAndStop().

Frame-based Movie Control
play(), stop(), gotoAndPlay(), gotoAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.188 null

Player Required
SWF5 or later

Syntax
a = null; // shows that value for a is currently not defined.

Arguments
None.

Description
Constant that can be assigned to variables indicating that no data is currently provided. It can also be used as a return value from a function to indicate that no data was provided.

Example
```
var a = new Array();
a[0] = "zero";
a[1] = null;
a[2] = "two";

for (i=0;i<3;i++)
{
        if (null == a[i])
                trace("a[" add i add "] has a null value");
}
```

displays
a[1] has a null value

4.6.8.11.189 Number(expression)

Player Required
SWF4 or later

Syntax
Number(expression)

Arguments
expression: The expression to convert to a floating-point number.

Returns
Converted number.
On Error returns 0 for SWF4, NaN for SWF5+

Description
Function: Converts an expression to a floating-point number.
If expression is a string, behaves like parseFloat().

If expression is boolean, returns 1 if true, 0 if false.
If expression is numeric, returns the result of the expression.
Note that numbers with a leading 0 (zero) are assumed to be octal. Use parseFloat or parseInt if you need to force the input radix.

Sample

The following examples use the Number function to evaluate various expressions:

```
trace(Number("-4e3"));        // displays -4000
trace(Number(4 + 3));         // displays 7
trace(Number(4 > 3));         // displays 1 (boolean true)
trace(Number("garbage"));             // displays 0 in SWF4, NaN in SWF5+
trace(Number("016"));         // displays 14 (16 octal)
```

See Also

parseFloat() and parseInt().

4.6.8.11.190 Object

Player Required

SWF5 or later

Syntax

new Object({ value });

Arguments

value; Optional argument of a number, boolean, or string that will be converted into an object. If this argument is not specified, then a new object with no defined properties is created.

Returns

Nothing.

Description

Constructor; creates a new Object object

Similar to an Array object; however, the properites of an Object object are accessed by name using the dot operator, or by string

using the array access operator []

Sample

```
x = new Object;
x.greeting = "Hello";
y = x["greeting"];
trace(y); // displays "Hello" in the debug window
trace(x.greeting); // displays "Hello" in the debug window

x = new Object("Hello");
trace(x);

days = new Object;
days["M"] = "Monday";
days["Tu"] = "Tuesday";
days["W"] = "Wednesday";
days["Th"] = "Thursday";

abbrev = "Tu";

days[abbrev] contains "Tuesday"
```

4.6.8.11.191 on(mouseEvent)

Player Required

SWF4 or later

Syntax

on(mouseEvent) {
    statements;
}

Arguments

mouseEvent: One of the defined mouse / keyboard Events.
statements: A list of any valid scripting commands that are executed if the Event has occurred.

Description

An Event Handler. Code is executed when the Mouse Event condition applies to the named button.
Unlike onSelfEvent(), the statements executed are with respect to the containing Object.

Sample

Stop Movie playing if button is pressed.

```
on(press) {
    stop();// stops containing Movie Clip / Sprite. If this button is contained by the main movie (_root) then i
}
```

See Also

dragOut, dragOver, keyPress(key), press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.192 on(changed)

Player Required

SWF4 or later

Syntax

on (changed) {
    statements;
}

Arguments

statements: A list of any valid scripting commands that are executed if the Event has occurred.

Description

An Event Handler function called when the text changes.
Note that Event statements assume that "_parent." is the containing Object.

Sample

Make the Text Field y scale bigger after each character is typed. This assumes an Input Text Field Object named txtInput.

```
on (changed) {
    txtInput._yscale += 10;
}
```

See Also

Text Field Object, on(changed), onSelfEvent(changed), onChanged

Notes

If the text object contains an associated variable (**Var:** field in the Text Object Properties panel) for SWF4 and SWF5 the variable must NOT contain a path.
eg. myvar is valid but _root.myvar is not.
This restriction does not apply to SWF6+

4.6.8.11.193 onChanged

Player Required
SWF6 or later

Syntax
onChanged = function () {
statements;
}

Arguments
statements: A list of any valid scripting commands that are executed if the Event has occurred.

Description
An Event Method invokes any assigned callback function when the text changes.
Note that Event statements assume that "this." is the containing Object.

Sample
Make the Text Field y scale bigger after each character is typed. This assumes an Input Text Field Object named txtInput. This script resides in the main movie, not the text field object.

```
onSelfEvent(load) {
    // scene load handler.
    txtInput.onChanged = function() {
        txtInput_yscale += 10;
    }
}
```

See Also
Text Field Object, on(changed), onSelfEvent(changed), onChanged

4.6.8.11.194 onEnterFrame

Player Required
SWF6 or later

Syntax

mc.onEnterFrame = function () {
statement(s);
}

Arguments
none

Returns
Nothing.

Description
This Event Method invokes any assigned callback function at the start of each frame of display, whether the associated object is playing or not.
However, if the Movie Clip / Sprite is removed, then the onEnterFrame actions are no longer processed. The Actions associated with the Event are processed before any Frame Action events for the object.

This Event Method is an ideal place to install collision detection routines for games, etc.

Sample
s1, s2 are both Movie Clips. Below is the scripting for Movie Clip s1. Debug trace is generated on transition. The script tests for the proximity of the other Movie Clip on each new frame.

```
onSelfEvent (load) {
    blIsNearLast = false;      // initialise boundary condition
    this.onEnterFrame = function () {
        blIsNear = _parent.s2.isNearThis();
        if (blIsNear != blIsNearLast) {
            // we have moved to or from is Near condition
            trace("x=" + _X + " y=" + _Y + " isnear=" + blIsNear);
        }
        blIsNearLast = blIsNear;
    }
}

onSelfEvent (press) {
    this.startDragLocked();   // when object is clicked on, start mouse drag
}

onSelfEvent (release) {
    stopDrag();      // when mouse is released, stop mouse drag
}
```

See Also
onEnterFrame, onSelfEvent(enterFrame)

4.6.8.11.195 onFrame()

Player Required
SWF4 or later

Syntax

```
onFrame(frame) {
  statement(s);
}
onFrame(frame,afterPlacedObjectEvents) {
  statement(s);
}
onFrame(frame,beforePlacedObjectEvents) {
  statement(s);
}
```

Arguments
frame: The Frame number when the statements are to be executed.
afterPlacedObjectEvents: Optional keyword. See below.
beforePlacedObjectEvents: Optional keyword. See below.

Returns
Nothing.

Description
This Event Handler Function runs whenever the playhead enters the nominated frame. It will not run again unless the frame is exited and re-entered.

The order of execution of code during a frame is as follows

1) any frame events for existing child objects
2) onFrame(x,beforePlacedObjectEvents)
3) any frame events for newly placed child objects
4) onFrame(x,afterPlacedObjectEvents)

The default if you specify neither option is "beforePlacedObjectEvents" (this is the same behaviour as Flash uses for its frame actions).

NOTE: You can have separate events for the one frame, with different options

Sample

Stop the Movie on the 10th Frame of the current Scene.

```
onFrame (10) {
  stop();
}
```

4.6.8.11.196 onLoad

Player Required
SWF6 or later

Syntax
obj.onLoad = function () {
    statement(s);
}

Arguments
None.

Returns
Nothing.

Description
This Event Method invokes any assigned callback function when the Movie Clip / Sprite / Object is loaded.

Note: Commands that affect the play order of other Movie Clip / Sprites / Objects may yield unpredictable results, as the exact order of loading is not defined

See Also
onLoad, onSelfEvent(load)

4.6.8.11.197 onSelfEvent(changed)

Player Required
SWF4 or later

Syntax
onSelfEvent (changed) {
    statements;
}

Arguments
statements: A list of any valid scripting commands that are executed if the Event has occurred.

Description
An Event Handler function called when the text changes.
Note that Event statements assume that "this." is the containing Object.

Sample
Make the Text Field y scale bigger after each character is typed. This assumes an Input Text Field Object named txtInput.

```
onSelfEvent (changed) {
    _yscale += 10;
}
```

See Also
Text Field Object, on(changed), onSelfEvent(changed), onChanged

4.6.8.11.198 onSelfEvent(enterFrame)

Player Required
SWF4 or later

Syntax
onSelfEvent (enterFrame, [includingFirstFrame]) {
statement(s);
}

Arguments
includingFirstFrame: Optional keyword. see below

Returns
Nothing.

Description
This Event Handler Function is called at the start of each frame of display, whether the associated object is playing or not. However, if the Movie Clip / Sprite is removed, then the onEnterFrame actions are no longer processed. The Actions associated with the Event are processed before any Frame Action events for the object.

If "includingFirstFrame" is specified, then the actions will be processed on all frames of display starting from and including the very first frame that the object is placed. If "includingFirstFrame" is omitted, then it will not run for the first time until the frame AFTER the object is placed. Compare this with the onSelfEvent (load) Event Handler Function, which will run ONLY for the first frame of display.

This Event Handler is an ideal place to install collision detection routines for games, etc.

Sample
s1, s2 are both Movie Clips. Below is the scripting for Movie Clip s1. Debug trace is generated on transition. The script tests for the proximity of the other Movie Clip on each new frame.

```
onSelfEvent (enterFrame) {
    blIsNear = _parent.s2.isNearThis();
    if (blIsNear != blIsNearLast) {
        // we have moved to or from is Near condition
        trace("x=" + _X + " y=" + _Y + " isnear=" + blIsNear);
    }
    blIsNearLast = blIsNear;
}

onSelfEvent (load) {
    blIsNearLast = false;      // initialise boundary condition
}

onSelfEvent (press) {
    this.startDragLocked();   // when object is clicked on, start mouse drag
}

onSelfEvent (release) {
    stopDrag();      // when mouse is released, stop mouse drag
}
```

See Also
onEnterFrame, onSelfEvent(enterFrame)

4.6.8.11.199 onSelfEvent(load)

Player Required
SWF4 or later

Syntax

onSelfEvent (load) {
    statement(s);
}

Arguments
None.

Returns
Nothing.

Description
This Event Handler function is called when the Movie Clip / Object is loaded.
This function is useful for defining boundary (initial) conditions and initialising variables.

Note: Commands that affect the play order of other Movie Clip / Sprites / Objects may yield unpredictable results, as the exact order of loading is not defined

Sample
```
onSelfEvent (load) {
    mydebug = 0;        // define and initialise variable
    this._vx = 10;      // set initial velocity
}
```

See Also
onLoad, onSelfEvent(load)

4.6.8.11.200 onSelfEvent(mouseEvent) +

Player Required
SWF4 or later

Syntax
onSelfEvent(mouseEvent) {
    statements;
}

Arguments
mouseEvent: One of the defined Mouse / keyboard Events.
statements: A list of any valid scripting commands that are executed if the Event has occurred.

Description
An Event Handler. Code is executed when the Mouse Event condition applies to the Object.

Sample
Make this Object bigger when the mouse is rolled over it.

```
// Increase the scale by 10% when the mouse rolls over this button/Movie Clip / Sprite
onSelfEvent(rollOver) {
    _xscale += 10;
    _yscale += 10;
}

// return to normal size when mouse moves away
onSelfEvent(rollOut) {
    _xscale -= 10;
    _yscale -= 10;
}
```

Flash MX Differences
This function is not implemented in Flash MX.

See Also

dragOut, dragOver, keyPress(key), press, release, releaseOutside, rollOut, rollOver, on() and onSelfEvent().

4.6.8.11.201 ord()

Player Required
SWF4 or later

Syntax
ord(char)

Arguments
char: Character or string.

Returns
The ascii code of the 1st character.

Description
String function. Converts characters to ASCII code numbers.

Sample

```
myVar = ord("AB");     // myVar now contains 65, the ASCII code for 'A'
```

See Also
chr and String.charCodeAt().

4.6.8.11.202 parseFloat(string)

Player Required
SWF5 or later

Syntax
parseFloat(string)

Arguments
string: The string to read and convert to a floating-point number.

Returns
the numeric value of the item defined by the string.

Description
Function: Converts a string to a floating-point number. The function reads, or 'parses', and returns the numbers in a string until it reaches a character that is not a part of the initial number. If the string does not begin with a number that can be parsed, parseFloat returns NaN (not a number). White space preceding valid integers is ignored, as are trailing non-numeric characters.

Sample
The following examples use the parseFloat function to evaluate various types of numbers:

```
trace(parseFloat("-4.9"));            // displays -4.9
trace(parseFloat("-4e3"));            // displays -4000
trace(parseFloat("-4.9stuff"));       // displays -4.9
trace(parseFloat("garbage")); // displays 0
```

See Also
parseInt() and Number().

4.6.8.11.203 parseInt(string)

Player Required
SWF5 or later

Syntax
parseInt(*expression*, {*radix*})

Arguments
expression: A string to convert to a integer.
radix: An optional argument used to represent the radix (base) of the integer. Acceptable values range from 2-36.

Returns
An integer.

Description
Function: Converts a string to an integer. If the specified string in the Arguments cannot be converted to a number, the function returns 0. All strings are assumed to be base 10.
If the string represents a floating-point number, that number is truncated.

Sample
The following examples use the parseInt function to evaluate various types of numbers:

```
parseInt("3.5"); // returns 3

parseInt("-3.5"); // returns -3

parseInt("bar"); // returns 0

parseInt("4foo"); // returns 4

parseInt("3.5", 5); // returns 3
```

See Also
parseFloat()

4.6.8.11.204 PI

Alternate syntax for Math.PI.

4.6.8.11.205 play()

Player Required
SWF4 or later

Syntax
[object.]play()

Arguments
object: Movie Clip / Sprite / Object that the play command applies to.

Returns
Nothing.

Description
Action and Movie Clip / Sprite Method: Moves the playhead forward in the Timeline.

Sample
If new hi score, start the newhiscore Movie Clip / Sprite to play.

```
if (score > hiscore) {
    newhiscore.play();
}
```

See Also
stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.206 playSound()

Player Required
SWF4 or later

Syntax
playSound(soundname, dontplayifloaded [, volume, loop, fadein, fadeout, effect])

Arguments
soundname: Name of imported sound (string).
dontplayifloaded: Boolean. If true, don't play sound if it is already playing.
volume: Volume to play sound, 0 to 100% (default = 100).
loop:: After sound has played, loop this many times. default = 0 (loop audio file one time).
fadein: Boolean. If true, fade in over 1st loop. Default = false.
fadeout: Boolean. If true, fade out during last loop. Default = false.
effect: One of the supported Effect constants, 'None', 'Fade In', 'Fade Out', 'Pan left to right', 'Pan right to left', 'Left channel only' or 'Right channel only'. Default = 'None').

Note: Setting the loop value to "1" or "0" will play the sound through one time. Due to a limitation in the Flash player, there is no infinite loop setting. 65535 is the largest number you can have for the loop value and negative loop values will play the sound once.

Note: All of the parameters to the playSound/stopSound must be constant values (either quotes strings, numbers or true/false). You cannot use variables.

Note: The playSound() command cannot be placed inside a conditional section of code. To work around this, place the command in a movie clip or a different section of the time line and then conditionally play that movie clip or section of the movie.

Returns
Nothing.

Description
Defines how a sound should be played.

Sample
On Frame 2, play the sound "chatinvt.wav" 5 times at a max volume of 90%.
Fade in on the first loop, fade out on the last loop. Pan from left to right on each loop.

```
onFrame (2) {
    playSound("ChatInvt.wav",true,90,5,true,true,'Pan Left to Right');
}
```

Example of conditionally playing a sound. In this case, the sound is only played if the object is pressed and the object 'b' is not visible.

```
onSelfEvent (press)
{
    if (!b._visible)
    {
       b._visible = true;
      this.gotoAndPlay("pop");
    }
}
```

```
onSelfEvent (load)
{
    b._visible = false;
}

onFrame (1)
{
    stop();
}
onFrame (2)
{
    setLabel("pop");
    // play sound
    playSound("pop.wav");
    stop();
}
```

See Also
playSound(), stopAllSounds(), Sound Object

4.6.8.11.207 plus (addition)

Player Required
SWF4 or later

Syntax
expression1 plus *expression2*

Arguments
expression: A number or a variable that evaluates to a number.

Returns
Floating-point result of expression1 plus expression2.

Description
If both expressions are integers, the sum is an integer. If either or both expressions are floating-point numbers, the sum is a floating-point number

Sample
This statement below adds the integers 2 and 3 and displays the resulting integer, 5, in the 'Debug' window:

```
trace (2 plus 3);
```

This statement below adds the floating-point numbers 2.5 and 3.25 and displays the result, 5.75, a floating-point number, in the 'Debug' window:

```
trace (2.5 plus 3.25);
```

This statement will add the sum of both variables as numbers:

```
onSelfEvent (load) {
    num1 = "10";
    num2 = "9.55";
    trace(num1 plus num2);
}
// The output in the Debug window will be 19.55
```

See Also
+ (addition), add (concat strings)

4.6.8.11.208 prevFrameAndPlay()

Player Required
SWF4 or later

Syntax
[*object.*]prevFrameAndPlay()

Arguments
object: The name of the Movie Clip / Sprite to be played.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead to the previous Frame in a Scene and plays from that Frame.

Sample

```
onFrame (5) {
    prevFrameAndPlay();  // continues playing from frame 4
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.209 prevFrameAndStop()

Player Required
SWF4 or later

Syntax
[*object.*]prevFrameAndStop()

Arguments
object: The name of the Movie Clip / Sprite to be played.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead to the previous Frame in a Scene and stops at that Frame.

Sample

```
onFrame (5) {
    prevFrameAndStop();  // Movie Clip / Sprite moves playhead to 4th frame and stops.
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.210 prevSceneAndPlay() +

Player Required
SWF4 or later

Syntax
prevSceneAndPlay()

Arguments
None.

Returns
Nothing.

Description
Action: Sends the playhead to the previous Scene and plays from the first Frame.

Sample
```
onFrame (10) {
    prevSceneAndPlay(); // identical to gotoSceneAndPlay("<previous scene>", 1);
}
```

Flash MX Differences
This function is not implemented in Flash MX.

See Also
Scene-based Movie Control
gotoSceneAndPlay(), gotoSceneAndStop(), nextSceneAndPlay(), nextSceneAndStop() and prevSceneAndStop().

Frame-based Movie Control
play(), stop(), gotoAndPlay(), gotoAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.211 prevSceneAndStop() +

Player Required
SWF4 or later

Syntax
prevSceneAndStop()

Arguments
None.

Returns
Nothing.

Description
Action: Sends the playhead to the previous Scene and stops at the first Frame.

Sample
```
onFrame (10) {
    prevSceneAndStop(); // identical to gotoSceneAndStop("<previous scene>", 1);
}
```

Flash MX Differences
This function is implemented in MX as prevScene().

See Also
Scene -based Movie Control
gotoSceneAndPlay(), gotoSceneAndStop(), nextSceneAndPlay(), nextSceneAndStop() and prevSceneAndPlay().

Frame-based Movie Control
play(), stop(), gotoAndPlay(), gotoAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.212 print()

Player Required
SWF5 or later

Syntax
print(*{"Bounding box"}*);
printTarget(*target {, "Bounding box"}*);
printNum(*level {, "Bounding box"}*);
target.print(*{"Bounding box"}*);
_level.print(*{"Bounding box"}*);

Arguments
target; The Movie Clip that contains the frames you want to print. By default, all frames inside the Movie Clip will print. In order to print only specific frames, add a **#P** label to each frame you want to print.

level; The level of the SWF file that you want to print.

Bounding box; A modifier that sets the print area of the movie. This argument is optional.
You can choose one of the following:

- *bmovie;* uses the bounding box of a specific frame as the print area for all other printable frames. Use a #b frame label on a specific frame to use it as the print area.
- *bmax;* can be used when the printable frames may vary in size.
- *bframe;* uses the bounding box from each printable frame for the print area. The print area can change for each frame based on its size and will scale objects to fit within the print area. You can use bframe if each frame consists of different sized objects that you want to fill the printed area.

Returns
Nothing.

Description
Action; prints the specified *target* according to the boundaries set in the *bounding box* argument. If you only want to print specific frames, you will need to add a **#P** frame label to each frame you want to print. The print action will not allow you to print movies that have alpha transparency. The print action generally produces higher quality prints than that produced using the printAsBitmap action.

If no boundaries are specified in the *bounding box* argument, the dimensions of the loaded movie are used as the print area.
All of the data you wish to print must be loaded before the print action will be executed.

See Also
printAsBitmap

4.6.8.11.213 printAsBitmap()

Player Required
SWF5 or later

Syntax
printAsBitmap(*{"Bounding box"}*)
printAsBitmapTarget(*target {, "Bounding box"}*)
printAsBitmapNum(*level {, "Bounding box"}*)
target.printAsBitmap(*{"Bounding box"}*)
_*level*.printAsBitmap(*{"Bounding box"}*)

Arguments
target; The Movie Clip that contains the frames you want to print. By default, all frames inside the Movie Clip will print. In order to print only specific frames, add a **#P** label to each frame you want to print.

level; The level of the SWF file that you want to print.

Bounding box; A modifier that sets the print area of the movie. This argument is optional.
You can choose one of the following:

- *bmovie;* uses the bounding box of a specific frame as the print area for all other printable frames. Use a #b frame label on a specific frame to use it as the print area.
- *bmax;* can be used when the printable frames may vary in size.
- *bframe;* uses the bounding box from each printable frame for the print area. The print area can change for each frame based on its size and will scale objects to fit within the print area. You can use bframe if each frame consists of different sized objects that you want to fill the printed area.

Returns
Nothing.

Description
Action; prints the specified Movie Clip / Sprite as a bitmap image. You can use printAsBitmap if you have frames with objects using transparency. The printAsBitmap also prints using the printer's highest available resolution.

If no boundaries are specified in the bounding box argument, the dimensions of the loaded movie are used as the print area.
All of the data you wish to print must be loaded before the print action will be executed.

Note: If you do not need to print using alpha transparency, it is advised to use the print action for better results.

See Also
print

4.6.8.11.214 removeMovieClip()

Player Required
SWF4 or later

Syntax
MovieClipName.removeMovieClip()

Arguments
MovieClipName: The target path of the Movie Clip / Sprite to delete.

Returns
Nothing.

Description

Movie Clip Method: Removes an Instance of a Movie Clip created with the duplicateMovieClip() method.

Sample

This example duplicates the existing Movie Clip, mc1, and creates a new Movie Clip mc2 on the same level. The position of both Movie Clips is adjusted so that they can be seen. To remove the duplicated Movie Clip, use mc2.removeMovieClip(). The created Movie Clip, mc2 is removed after Frame 12.

```
onSelfEvent (load) {
    mc1.duplicateMovieClip("mc2",1);
    mc1._X += 50;
    mc1._Y += 50;
    mc2._X += 150;
    mc2._Y += 50;
}

onFrame (12) {
    mc2.removeMovieClip();  // remove the duplicated Movie Clip
}
```

See also

duplicateMovieClip

4.6.8.11.215 removeSprite() +*

NOTE: This is a deprecated command. Use removeMovieClip instead.

Player Required

SWF4 or later

Syntax

MovieClipName.removeSprite()

Arguments

MovieClipName: The target path of the Movie Clip / Sprite to delete.

Returns

Nothing.

Description

Movie Clip Method: Removes an Instance of a Sprite created with the duplicateSprite() method.

Sample

This example duplicates the existing Sprite, s1, and creates a new Sprite s2 on the same level. The position of both Sprites is adjusted so that they can be seen. To remove the duplicated Sprite, use s2.removeSprite(). The created Sprite, s2 is removed after Frame 12.

```
onSelfEvent (load) {
    s1.duplicateSprite("s2",1);
    s1._X += 50;
    s1._Y += 50;
    s2._X += 150;
    s2._Y += 50;
}

onFrame (12) {
    s2.removeSprite();  // remove the duplicated Sprite
}
```

See also

duplicateSprite

4.6.8.11.216 return

Player Required
SWF5 or later

Syntax
return;
return expression;

Arguments
expression: Optional parameter that returns string / variable value. If omitted, returns NULL.

Returns
Value defined by the expression.

Description
Allows a user-defined function to return a value.

Sample
```
function area(p) {
    return Math.PI * p * p;
}
```

See Also
function

Flash MX Differences
It is not possible to return arrays or Objects.

4.6.8.11.217 setMask()

Player Required
SWF6 or later

Syntax
myMovieClip.setMask(*maskMovieClip*)

Arguments
myMovieClip; The instance name of a Movie Clip that will be masked.
maskMovieClip; The instance name of a Movie Clip to be used as a mask.

Returns
Nothing.

Description
Method; makes the Movie Clip specified in the *maskMovieClip* argument a mask that reveals the Movie Clip specified in the *myMovieClip* argument.

Method; makes the movie clip in the parameter maskMovieClip into a mask that reveals the movie clip specified by the myMovieClip parameter. A Movie Clip cannot be used as its own mask, eg. *myMovieClip* .setMask(*myMovieClip*). The setMask method will allow Movie Clips with complex content to be used as a mask; however, device fonts will not be masked.

To disable a mask created with the setMask method, set the *maskMovieClip* argument to 'null', eg. *myMovieClip*.setMask(null)

4.6.8.11.218 scroll

Player Required
SWF4 or later

Syntax
textFieldVariableName.scroll = x

Description
A property that controls the display of information in a Text Field associated with a variable. The scroll property defines where the Text Field begins displaying content; after you set it, the Flash Player updates it as the user scrolls through the Text Field.

The scroll property is useful for directing users to a specific paragraph in a long passage, or creating scrolling Text Fields. This property can be retrieved and modified.

Sample
The following code is attached to an 'Up' button that scrolls the Text Field txt:

```
on (release) {
    txt.scroll += 1;  // scroll to the next line
}
```

Note:

- This property is not available with simple Text Field Objects
- Property must be an integer value. Consequently smooth scrolling is not possible
- If smooth scrolling is required, it is possible to simulate using Movie Clip / Sprites and Movie Clip / Sprite masks

See also
maxscroll, Text Field Properties and Methods and TextField Access Matrix for more information about the properties that can be accessed.

4.6.8.11.219 setProperty()

Player Required
SWF4 or later

Syntax
[*instance*.]setProperty([instancename,] property, value)

Arguments
instancename: The Instance name of a Movie Clip for which the property is being set. This can either be the Instance name (using slash notation) or the _target property. If omitted, the Instance specified by *instance* is used. If *instance* is omitted, defaults to 'this'.

property: A property of a Movie Clip / Sprite.

Returns
The specified property value.

Description
Function: Returns the value of the specified property for the Movie Clip instancename.

Sample
moves the object s1 to x = 300 then to x = 100

```
onFrame (10) {
    setProperty(_root.s1._target, _X, 300);
}
```

```
onFrame (20) {
    setProperty("/s1",_X, 100);
}
```

See Also
getProperty() and setProperty().

4.6.8.11.220 skipFrameAndPlay()

Player Required
SWF4 or later

Syntax
[*object.*]skipFrameAndPlay(*n*)

Arguments
object: The name of the Movie Clip / Sprite to be played.
n: The number of Frames to be skipped. If n is negative, moves to previous Frames.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead ahead n Frames and plays from that Frame.

Sample

```
onFrame (5) {
    skipFrameAndPlay(3);  // Movie Clip / Sprite starts playing from 8th frame
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop() and skipFrameAndStop().

4.6.8.11.221 skipFrameAndStop()

Player Required
SWF4 or later

Syntax
[*object.*]skipFrameAndStop(*n*)

Arguments
object: The name of the Movie Clip / Sprite to be moved.
n: The number of Frames to be skipped. If n is negative, moves to previous Frames.

Returns
Nothing.

Description
Action and Movie Clip Method: Sends the playhead ahead n Frames and stops at that Frame.

Sample

```
onFrame (5) {
    skipFrameAndStop(3);  // moves the playhead to the 8th frame.
}
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop() and skipFrameAndPlay().

4.6.8.11.222 Sound (Object)

The Sound object is used to create and control specific sounds in your movie or all the sounds in the movie.

4.6.8.11.222.1 new Sound()

Player Required
SWF5 or later

Syntax
new Sound(*{target}*);

Arguments
target: the Movie Clip / Sprite instance that contains the Sound. This is an optional argument.

Returns
Nothing.

Description
Constructor; creates a new Sound object for the *target* Movie Clip / Sprite. If the *target* argument is not specified then the new Sound object will control all sounds in the movie.

Sample

```
mySound = new Sound(myMovieClip);
mySound.setVolume(75);

// creates a sound object to control the sound playing within the Movie Clip / Sprite "myMovieClip" and sets its
```

4.6.8.11.222.2 getPan()

Player Required
SWF5 or later

Syntax
soundObject.getPan();

Arguments
None.

Returns
As described below:

Description
Method; retrieves the pan level (balance) of the Sound object. The result is an integer between -100 (full left pan) and 100 (full right pan). A pan level of zero indicates that the sound object is balanced equally between the left and right channels.

4.6.8.11.222.3 getVolume()

Player Required
SWF5 or later

Syntax
soundObject.getVolume();

Arguments
None.

Returns
As described below:

Description
Method; retrieves the volume level of the Sound object. The result is an integer between 0 (no volume) and 100 (full volume).

4.6.8.11.222.4 loadSound()

Player Required
SWF6 or later

Syntax
soundObject.loadSound("*url*", *isStreaming*);

Arguments
url: path to the sound file to be loaded
isStreaming: a setting of "true" or "false" to indicate whether the Sound is streaming or not.

Returns
Nothing.

Description
Method; loads an .MP3 file located at the specified *url* into an instance of the specified Sound object. The *isStreaming* argument determines whether or not the sound is streaming or event-based.

Sample
```
onFrame (1) {
    rockSong = new Sound(myMovieClip);
    rockSong.loadSound("http://www.yoursite.com/yoursong.mp3", true);
}
```

4.6.8.11.222.5 setPan()

Player Required
SWF5 or later

Syntax
soundObject.setPan(*amount*);

Arguments
amount: an integer between -100 (full left pan) and 100 (full right pan). A setting of zero sets even balance between the left and right channels.

Returns
Nothing.

Description
Method; sets the pan level for the specified Sound object. A setting of -100 plays the sound only in the left speaker, while a setting of 100 plays the sound only in the right speaker. A setting of zero plays the sound evenly in both speakers.

Sample
```
onSelfEvent (load) {
    introSound = new Sound(myMovieClip);
    introSound.setPan(-100);
}
// creates a new Sound object "introSound" for the target "myMovieClip" and sets the pan level to full left
```

4.6.8.11.222.6 setVolume()

Player Required
SWF5 or later

Syntax
soundObject.setVolume(*volume*);

Arguments
volume: an integer between 0 (off) and 100 (full volume).

Returns
Nothing.

Description
Method; sets the volume level for the specified Sound object. A setting of 0 turns the sound off, and a setting of 100 plays the sound at full volume (100%).

Sample

```
onSelfEvent (load) {
    introSound = new Sound(myMovieClip);
    introSound.setVolume(75);
}

// sets the volume level of the Sound object "introSound" to 75 percent.
```

4.6.8.11.222.7 start()

Player Required
SWF5 or later

Syntax
soundObject.start(*{secondOffset, loop}*);

Arguments
secondOffset: point in the Sound object to begin playing from (in seconds). This argument is optional.
loop: the number of times the Sound object should play through consecutively. This argument is optional.

Returns
Nothing.

Description
Method; starts playing the specified Sound object. Begins from the point specified by the *secondOffset* argument or the from the beginning if none is given.

Sample

```
onSelfEvent (load) {
    introSound = new Sound(myMovieClip);
    introSound.start(10, 100);
}
// begins playing the Sound object "introSound" at the 10-second mark and loops through it 10 times
```

4.6.8.11.223 SQRT1_2

Alternate syntax for Math.SQRT1_2.

4.6.8.11.224 SQRT2

Alternate syntax for Math.SQRT2.

4.6.8.11.225 Stage

Contains properties and methods used for gathering information about the movie itself (such as the physical dimensions of stage).

- Stage.align
- Stage.height
- Stage.scaleMode
- Stage.width

Event Listener routines can also be added / removed via the addListener() and removeListener() methods.

4.6.8.11.225.1 Stage.align

Player Required
SWF6 or later

Syntax
Stage.align

Arguments
None.

Returns
As described below:

Description
Property; gives the alignment of the movie in the Stage.

The following are the values of the align property (any value not listed will indicate centered).

"T" = Top-center
"TL" = Top-Left
"TR" = Top-Right
"L" = Left-Centered
"R" = Right-Centered
"BR" = Bottom-Right
"BL" = Bottom-Left
"B" = Bottom

Note: There appears to be a bug in the flash player that causes some of the values to be returned with the letters in a different order. It is possible that the bug may be fixed in future versions.

Value Set	Value Returned
TL	LT
TR	TR
BR	RB
BL	LB

Example

```
onSelfEvent (load)
{
    Stage.align="TL";
    trace(Stage.align);
    Stage.align="TR";
    trace(Stage.align);
    Stage.align="BL";
    trace(Stage.align);
    Stage.align="BR";
    trace(Stage.align);
}
```

```
Returns:
LT
TR
LB
RB
```

4.6.8.11.225.2 Stage.height

Player Required
SWF6 or later

Syntax
Stage.height

Arguments
None.

Returns
As described below:

Description
Property; a read-only property that supplies the current height of the Stage (_root).

4.6.8.11.225.3 Stage.scaleMode

Player Required
SWF6 or later

Syntax
Stage.scaleMode

Arguments
None.

Returns
Current property value as defined below:

Description
Property; returns or sets the current scale of the Flash movie on the Stage. The default value of this property is "showAll". Other acceptable values are "exactFit", "noScale", and "noBorder".

showAll: makes the entire Movie visible in the specified area. The original aspect ratio of the Movie is maintained, and no distortion occurs. Borders may appear on two sides of the movie.

noBorder: scales the Movie to fill the specified area. The original aspect ratio of the movie is maintained, and no distortion occurs. Portions of the movie may be cropped.

exactFit: makes the entire Movie visible in the specified area. However, no attempt is made to preserve the original aspect ratio, and distortion may occur.

noScale: behaves similar to showAll except that the movie is not resized if the player window is resized.

Note: the value of "*showAll*" is always returned in the Internal player - the exact property will be returned in the external player or browser.

4.6.8.11.225.4 Stage.width

Player Required
SWF6 or later

Syntax
Stage.width

Arguments
None.

Returns
As described below:

Description
Property; a read-only property that supplies the current width of the Stage (_root).

4.6.8.11.226 startDrag()

Player Required
SWF5 or later

Syntax
target.startDrag(lockCenter:Boolean[,left, top, right, bottom])

Arguments
target: The name of the Movie Clip / Sprite to be dragged. If omitted, the default is 'this'.
lockCenter: If true, the mouse pointer is locked to the center of the object.
The optional Arguments, left, top, right and bottom, if specified, define a bounding region for the Movie Clip. The coordinates are specified relative to the Movie Clip's parent. Note that the order of these parameters is different to the parameter ordering of startDragLocked() and startDragUnlocked().

Returns
Nothing.

Description
This Movie Clip Method makes the target Movie Clip / Object draggable while the Movie is playing. Only one Movie Clip / Object can be dragged simultaneously.

Dragging continues until a stopDrag Action is called or startDrag(), startDragLocked() or startDragUnlocked() is applied to a different Movie Clip / Sprite / Object.

Sample
Make Object draggable while mouse is pressed:

```
onSelfEvent (press)
{
    this.startDrag(false);
}
onSelfEvent(release)
{
    stopDrag();
}
```

See Also
startDragLocked(), startDragUnlocked() and stopDrag().

4.6.8.11.227 startDragLocked() +*

NOTE
This is a deprecated function. Use startDrag() instead.

Player Required
SWF4 or later

Syntax
target.startDragLocked([left, right, top, bottom])

Arguments
target: The name of the Object / Movie Clip to be dragged. If omitted, the default is 'this'.
The optional Arguments, left, right, top, and bottom, if specified, define a bounding region for the Movie Clip. The coordinates are specified relative to the Movie Clip's parent.

Returns
Nothing.

Description
This Movie Clip Method makes the target Movie Clip / Object draggable while the Movie is playing. Only one Movie Clip / Object can be dragged simultaneously.

The Movie Clip / Object is locked to the center of the mouse position.

Dragging continues until a stopDrag Action is called or startDragLocked or startDragUnlocked is applied to a different Movie Clip / Object.

Sample
Make Object draggable while mouse is pressed:

```
onSelfEvent (press) {
    this.startDragLocked();
}
onSelfEvent (release) {
    stopDrag();
}
```

See Also
startDrag(), startDragLocked(), strtDragUnlocked() and stopDrag(). startDrag()

4.6.8.11.228 startDragUnLocked() +*

NOTE
This is a deprecated function. Use startDrag() instead.

Player Required
SWF4 or later

Syntax
target.startDragUnlocked([left, right, top, bottom])

Arguments
target: The name of the Object / Movie Clip to be dragged. If omitted, the default is 'this'.
The optional Arguments, left, right, top, and bottom, if specified, define a bounding region for the Movie Clip. The coordinates are specified relative to the Movie Clip's parent.

Returns
Nothing.

Description
This Movie Clip Method makes the target Movie Clip / Object draggable while the Movie is playing. Only one Movie Clip / Object can be dragged simultaneously.

The Movie Clip / Object is locked to mouse position where the user first pushed the mouse button.

Dragging continues until a stopDrag Action is called or startDragLocked or startDragUnlocked is applied to a different Movie Clip / Object.

Sample
Make Object draggable while mouse is pressed:

```
onSelfEvent (press) {
    this.startDragUnlocked();
}
onSelfEvent (release) {
    stopDrag();
}
```

See Also
startDrag() startDragLocked(), startDragUnlocked() and stopDrag().

4.6.8.11.229 stop()

Player Required
SWF4 or later

Syntax
[object.]stop()

Arguments
object: Movie Clip / Object that the stop command applies to.

Returns
Nothing.

Description
Action or Movie Clip Method: Stops the playhead at the current position.

Sample
```
intro.stop();  // stops the Movie Clip / Sprite intro from playing
_root.stop();  // stop the main Timeline
```

See Also
play(), stop(), gotoAndPlay(), gotoAndStop(), gotoSceneAndPlay(), gotoSceneAndStop(), nextFrameAndPlay(), nextFrameAndStop(), prevFrameAndPlay(), prevFrameAndStop(), skipFrameAndPlay() and skipFrameAndStop().

4.6.8.11.230 stopAllSounds()

Player Required
SWF4 or later

Syntax
stopAllSounds()

Arguments
None.

Returns
Nothing.

Description

Action: Stops all currently playing sounds. Does not stop the Movie. Sounds set to stream will resume playing as the playhead moves over the Frames they are in.

Sample

```
stopAllSounds(); // stops all sounds in the Movie
```

See Also

playSound() and stopAllSounds().

4.6.8.11.231 stopDrag()

Player Required

SWF4 or later

Syntax

stopDrag()

Arguments

None.

Returns

Nothing.

Description

This Action stops the current Drag Action.
As only one simultaneous Drag Action is permitted, the target Movie Clip / Object is implied.

Sample

Make Object draggable while mouse is pressed, stop dragging when mouse is released.

```
onSelfEvent (press) {
    this.startDrag(true);
}

onSelfEvent (release) {
    stopDrag();
}
```

See Also

startDrag(), startDragLocked(), startDragUnlocked() and stopDrag().

4.6.8.11.232 String(expression)

Player Required

SWF4 or later

Syntax

String(expression)

Arguments

expression: An expression to convert to a string.

Returns

String representing the expression.

Description

This function returns a string representation of the specified parameter as follows:

- If expression is a boolean value, the return string is "1" or "0", ("1" == true).
- If expression is a number, the return string is a text representation of the number.
- If expression is a string, the return string is expression.

Samples

```
String(4 > 3);         // returns "1"
String(_root.s1._target);      // returns "/s1"
String(44.3);                  // returns "44.3"
```

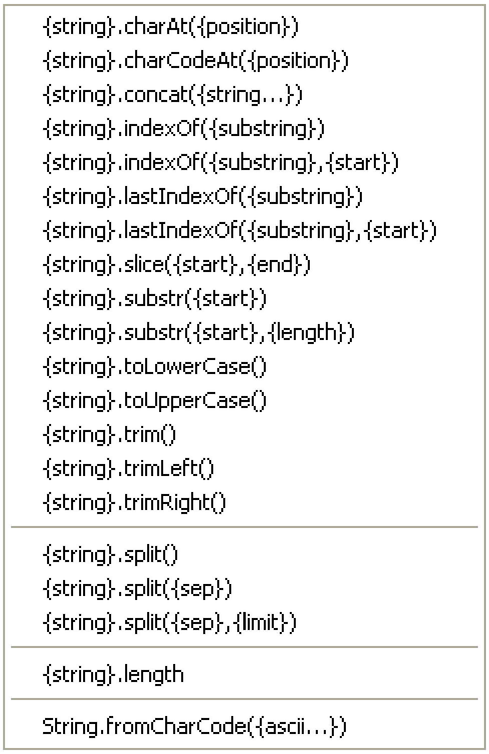
4.6.8.11.233 String (Object)

With all of the following methods, note that the 1st character in the string is at position '0'.

Some of the methods listed are of the form 'String.method()' whereas others are 'myString.method()'.

The methods listed as myString.method() will work with any string variable whereas methods listed as 'String.method()' are associated with the Class Object 'String' and will simply return a value (similar to Math.method()).

Methods available with the String Object can be selected from the list below.



{string}.charAt({position})
{string}.charCodeAt({position})
{string}.concat({string...})
{string}.indexOf({substring})
{string}.indexOf({substring},{start})
{string}.lastIndexOf({substring})
{string}.lastIndexOf({substring},{start})
{string}.slice({start},{end})
{string}.substr({start})
{string}.substr({start},{length})
{string}.toLowerCase()
{string}.toUpperCase()
{string}.trim()
{string}.trimLeft()
{string}.trimRight()

{string}.split()
{string}.split({sep})
{string}.split({sep},{limit})

{string}.length

String.fromCharCode({ascii...})

4.6.8.11.233.1 String.charAt(position)

Player Required
SWF5 or later

Syntax
myString.charAt(index)

Arguments
index: The number of the character in the string to be returned. The first character is at index position 0.

Returns

The character at the specified index.

Description

Method: Returns the character in the position specified by the parameter index. The index of the first character in a string is 0. If index is not a number from 0 to string.length - 1, an empty string is returned.

Sample

```
onSelfEvent (load) {
    i = 0;
    t1 = $version;
    trace(t1);
    do {
        trace(t1.charAt(i));
    } while (i++ < t1.length());
}
```

4.6.8.11.233.2 String.charCodeAt(position)

Player Required

SWF5 or later

Syntax

myString.charCodeAt(index)

Arguments

index: The number of the character in the string to be returned. The first character is at index position 0.

Returns

The character code at the specified index.

Description

Method: Returns the character code of the character in the position specified by the parameter index. The index of the first character in a string is 0. If index is not a number from 0 to string.length - 1, 0 is returned.

The character code is an integer from 0 to 65535.

Sample

```
onSelfEvent (load) {
    i = 0;
    t1 = $version;
    trace(t1);
    do {
        trace(t1.charCodeAt(i));
    } while (i++ < t1.length);
}
```

See Also

ord

4.6.8.11.233.3 String.concat(string)

Player Required

SWF5 or later

Syntax

myString.concat(value1,...valueN)

Arguments

value1,...valueN: Zero or more values to be concatenated.

Returns

Concatenated string. Original string is unchanged.

Description
Method: Combines the value of the String Object with the Arguments and returns the newly formed string; the original value, myString, is unchanged.

Sample
```
onSelfEvent (load) {
    str = "test2";
    trace(str.concat("hi","this", 4 > 3, "more"));  // returns "test2hithis1more"
    trace (str);  // returns "test2"
}
```

4.6.8.11.233.4 String.fromCharCode(ascii)

Player Required
SWF5 or later

Syntax
String.fromCharCode(c1,c2,...cN)

Arguments
c1,c2,...cN: Decimal integers that represent ASCII values.

Returns
String made up of characters assembled from the ASCII codes.

Description
Method: Returns a string made up of the characters represented by the ASCII values in the Arguments.

Sample
```
str = String.fromCharCode(60,61,62,63,64);
trace(str);    // returns the string "<=>?@"
```

Note: str.fromCharCode() is not valid

4.6.8.11.233.5 String.indexOf(substring [,start])

Player Required
SWF5 or later

Syntax
myString.indexOf(substring, [startIndex])

Arguments
substring: An integer or string specifying the substring to be searched for within myString.
startIndex: An integer specifying the starting point in myString to search for substring. This parameter is optional.

Returns
Returns the index of the first occurrence of the specified substring.

Description
Method: Searches the string and returns the position of the first occurrence of the specified substring. If the value is not found, the method returns -1.

Sample
```
onSelfEvent (load) {
    str = "abcdefgbc";
    trace (str.indexOf("bc"));        // returns 1, position of "bc"
    trace (str.indexOf("bd"));        // returns -1, not found
    trace (str.indexOf("bc",3));      // returns 7, second occurrence
    trace (str.indexOf("cd",4));      // returns -1 as "cd" exists before index 4
    trace (str);
}
```

4.6.8.11.233.6 String.lastIndexOf(substring [,start])

Player Required
SWF5 or later

Syntax
myString.lastIndexOf(substring, [startIndex])

Arguments
substring: An integer or string specifying the substring to be searched for within myString.
startIndex: An integer specifying the starting point in myString to search for substring. This parameter is optional. 0 represents the first character of the string.

Returns
Returns the index of the first occurrence of the specified substring.

Description
Method: Searches the string from right to left and returns the position of the first found (last in string) occurrence of the specified substring. If the value is not found, the method returns -1.

Sample

```
onSelfEvent (load) {
    str = "abcdefgbc";
    trace (str.lastIndexOf("bc"));           // returns 7, (last position of "bc")
    trace (str.lastIndexOf("bd"));           // returns -1, not found
    trace (str.lastIndexOf("bc",3));  // returns 7, (last position of "bc")
    trace (str.lastIndexOf("cd",3));  // returns 2, found because 'd' of "cd" is at 3. See note below.
    trace (str.lastIndexOf("cd",4));  // returns -1 as "cd" exists before index 4
    trace (str);
}
```

Note: As search is from right to left, returns position before startIndex as start of string (on right to left basis) started after startIndex

4.6.8.11.233.7 String.length

Player Required
SWF5 or later

Syntax
myString.length

Arguments
None. This read only property indicates the number of characters in the string.

Returns
This read only property indicates the number of characters in the string.

Description
Read only Property: Returns the number of characters in the string. As string indexes have zero based indexes. The last character is myString.length - 1

Sample

```
onSelfEvent (load) {
    str = "abcdefgbc";
    trace (str.length);       // returns 9
    trace (str.charAt(str.length - 1));      // returns the last character in the string, 'c'
}
```

4.6.8.11.233.8 String.slice(start,end)

Player Required
SWF5 or later

Syntax
myString.slice(start [, end])

Arguments
myString: The string variable.

start: A number specifying the index of the starting point for the slice. If start is a negative number, the starting point is determined from the end of the string, where -1 is the last character. 0 represents the first character.

end: A number specifying the index of the ending point for the slice. If end is not specified, the slice includes all characters from start to the end of the string. If end is a negative number, the ending point is determined from the end of the string, where -1 is the last character.

Returns
Substring specified by start, end Arguments.

Description
Method: Extracts a slice, or substring, of the specified String Object; then returns it as a new string without modifying the original String Object. The returned string includes the start character and all characters up to (but not including) the end character.

Sample

```
onSelfEvent (load) {
    str = "abcdefgbc";
    trace ("[" + str.slice(3,6) + "]");      // returns [def]
}
```

4.6.8.11.233.9 String.split(sep,{limit})

Player Required
SWF5 or later

Syntax
myString.split({sep},{limit})

Arguments
sep: The point at which the string is split. This can be either a single character, a string itself, or a variable that represents a string.
limit: An optional argument which determines the number of characters or substrings to place into the array.

Returns
The characters or substrings of the string placed in an array.

Description
Method: Splits any String into individual characters or substrings based on the location of the *sep* argument and the optional *limit* argument. If no arguments are specified, each individual character of the String is placed into the array separately.

Sample

This example uses no arguments:

```
onSelfEvent (load) {
    myString = "SWiSHmax";
    myString = myString.split();
    trace(myString); // displays "S,W,i,S,H,m,a,x" in the debug window
}
```

This example uses only the 'sep' argument:

```
onSelfEvent (load) {
    myString = "Dog and Cat";
    myString = myString.split(" and ");
    trace(myString); // displays "Dog,Cat" in the debug window
}
```

This example uses both the 'sep' and 'limit' arguments:

```
onSelfEvent (load) {
    myString = "Dog and Cat and Fish and Bird";
    myString = myString.split(" and ", 3);
    trace(myString); // displays "Dog,Cat,Fish" in the debug window
}
```

4.6.8.11.233.10 String.substr(start [,length])

Player Required
SWF5 or later

Syntax
myString.substr(*start* [, *length*])

Arguments
start: An integer. The position of 1st character in myString to form substring. If start is a negative number, the starting position is determined from the end of the string, where the -1 is the last character.
length: An optional integer. The number of characters in the substring being created. If length is not specified, the substring includes all of the characters from the start to the end of the string.

Returns
As described below:

Description
Method: Returns the substring formed from the characters in a string from the index specified in the start parameter through the number of characters specified in the length parameter. The substr method does not change the string specified by myString, it returns a new string.

Sample

```
onSelfEvent (load) {
    str = "abcdefgbc";
    trace (str.substr(3,3));  // returns "def"
    trace (str);              // returns "abcdefgbc"
}
```

4.6.8.11.233.11 String.substring(start,{end})

Player Required
SWF5 or later

Syntax
myString.substring(*start* {, *end*})

Arguments
start: An integer. The position of 1st character in myString to form substring. If start is a negative number, the starting position is determined from the end of the string, where the -1 is the last character.

end: An optional integer that is one index value higher than the last character of the string that you want to be extracted. Acceptable values are 1 through String.length. If this argument is not provided, the substring method will use string.length. Negative values for this argument default to a value of 0.

Returns
As described below:

Description
Method: Returns the substring formed from the characters in a string from the index specified in the *start* argument through the number of characters specified in the *end* argument. If the *end* argument is not provided then the end of this substring will be the last character in the string. If the value specified in the *start* argument is greater than the value specified in the *end* argument, then these values are swapped before the method is executed. If the value of the end argument is a negative number, then a value of 0 is used in its place.

Sample
```
onSelfEvent (load) {
    str = "SWiSH Max is Great!";
    trace(str.substring(0,5));
}
// displays "SWiSH" In the Debug window

onSelfEvent (load) {
    str = "SWiSH Max is Great";
    trace(str.substring(0,9));
}
// displays "SWiSH Max" In the Debug window
```

4.6.8.11.233.12 String.toLowerCase()

Player Required
SWF5 or later

Syntax
myString.toLowerCase()

Arguments
none

Returns
The specified string in all lower-case.

Description
Method: Returns the specified string with all upper-case characters converted into lower-case characters.

Sample
```
onSelfEvent (load) {
    str = "Hello World, how are YOU doing?";
    str = str.toLowerCase();
    trace(str);  // returns "hello world, how are you doing?"
}
```

4.6.8.11.233.13 String.toUpperCase()

Player Required
SWF5 or later

Syntax
myString.toUpperCase()

Arguments
none

Returns
The specified string in all upper-case.

Description
Method: Returns the specified string with all lower-case characters converted into upper-case characters.

Sample

```
onSelfEvent (load) {
    str = "Hello World, how are YOU doing?";
    str = str.toUpperCase();
    trace(str);  // returns "HELLO WORLD, HOW ARE YOU DOING?"
}
```

4.6.8.11.233.14 String.trim()

Player Required
SWF5 or later

Syntax
myString.trim()

Arguments
none

Returns
The specified string with any whitespace removed from the beginning and end of the string.

Description
Method: Returns the specified string with all whitespace (blank spaces) removed from the beginning and the end of the string.

Sample

```
onSelfEvent (load) {
    str = "   Hello World   ";
    trace("[" add str.trim() add "]"); // returns "[Hello World]"
}
```

4.6.8.11.233.15 String.trimLeft()

Player Required
SWF5 or later

Syntax
myString.trimLeft()

Arguments
none

Returns
The specified string with any whitespace removed from the beginning of the string.

Description
Method: Returns the specified string with all whitespace (blank spaces) removed from the beginning (left side) of the string.

Sample
```
onSelfEvent (load) {
    str = "   Hello World   ";
    trace(str.trimLeft()); // returns "Hello World   "
}
```

4.6.8.11.233.16 String.trimRight()

Player Required
SWF5 or later

Syntax
myString.trim()

Arguments
none

Returns
The specified string with any whitespace removed from the end of the string.

Description
Method: Returns the specified string with all whitespace (blank spaces) removed from the end (right side) of the string.

Sample
```
onSelfEvent (load) {
    str = "   Hello World   ";
    trace(str.trimRight()); // returns "   Hello World"
}
```

4.6.8.11.234 swapDepths()

Player Required
SWF5 or later

Syntax
myMovieClip.swapDepths(*depth*)
myMovieClip.swapDepths(*target*)

Arguments
depth: an integer used to determine the depth at which the specified Movie Clip will be moved to.
target: the target to be swapped with the specified Movie Clip (*myMovieClip*).

Returns
Nothing.

Description
Method; used to swap the depth level of the specified Movie Clip with either another movie (*target* argument) or a specified depth (*depth* argument). If another movie resides at the depth level specified, the depth level of both instances will be swapped.

Note: It is required that both instances have the same parent Movie Clip.

Sample
```
onFrame(10) {
    myMovieClip1.swapDepths(20); // moves myMovieClip1 to a depth level of 20
}

onFrame(10) {
    myMovieClip1.swapDepths(myMovieClip2); // swaps the depth level of myMovieClip1 with myMovieClip2
}
```

See Also

getDepth()

4.6.8.11.235 switch()

Player Required
SWF4 or later

Syntax

```
switch (expression1) {
case expression2 :
  statements
  break;
case expression3:
  statements
  break;
default :
  statements
}
```

Arguments
expression1; Any expression.
expression2; Any expression
statements; script to execute

Returns
Nothing.

Description
Action; used to create a branching structure for <%SWISHSCRIPT%> statements. There can be multiple 'case' labels, and optionally a single 'default' label. The switch action evaluates the first expression within the parentheses. It then compares it to each expression in the case labels. If it finds one that matches, then the statements of the switch statement are executed from there. If none of the case labels match, and there is a 'default' label, the statements are executed from there. Use a 'break' statement between cases if you don't want the statements of following cases to execute as well.

See Also
case, default, break

4.6.8.11.236 Synonyms

Various synonyms are used to refer to commonly used references. This use of synonyms can simplify scripting.

4.6.8.11.236.1 Object Reference

Various synonyms can be used when referring to an Object or Movie Clip / Sprite.

The _parent, _root and this synonyms are supported.

4.6.8.11.236.2 Scene Name

The following synonyms are recognized by the gotoSceneAndPlay() and gotoSceneAndStop() functions.

Synonym	**Meaning**
<current scene>	Use current Scene
<first scene>	Move to first Scene in Movie
<last scene>	Move to last Scene in Movie
<next scene>	Move to next Scene in Movie
<previous scene>	Move to previous Scene in Movie

4.6.8.11.237 TAB

Player Required
SWF5 or later

Syntax
TAB

Description
Property: Constant associated with the key code value for the Tab key (9).

4.6.8.11.238 tabChildren

Player Required
SWF6 or later

Syntax
myMovieClip.tabChildren

Arguments
None.

Returns
Current value of the property.

Description
Property; if this property is undefined or set to true, the children of a Movie Clip / Sprite are included in tab ordering. If the property is set to false, then they are not included in tab ordering.

4.6.8.11.239 tellTarget()

Player Required
SWF4 or later

Syntax

```
tellTarget(object) {
   statement(s);
}
```

Arguments
object: An Instance of an Object or Movie Clip.
statement(s): An Action or group of Actions enclosed in curly brackets {...}.

Returns
Nothing.

Description
Action: Allows you to specify an Object (such as a Movie Clip) with the Object parameter. The statements within the tellTarget curly brackets are applied to that Object. This prevents you from having to repeatedly write the Object's name or the path to the Object.

Sample
Consider a Movie Clip _root.s1. Movie Clip properties can be accessed from other Movie Clips or from the root level via the following code:

```
tellTarget (_root.s1) {
    _X += 10;  // same as _root.s1._X += 10
    _alpha = 40;       // same as _root.s1._alpha = 40
}
```

Note: In Flash the Object is a either a string containing a target OR a target reference. In SWiSH Max, it is always a target reference. If you want to use a string variable then enclose it in parentheses, as shown below:

```
who = "myMovieClip";
tellTarget((who)) {
    stop(); // same as myMovieClip.stop();
}
```

If you do not use the parentheses around the variable name, it will be treated as a target name instead. In the above example, if you did not put the extra parenthesis around the variable name you would get this:

```
who = "myMovieClip";
tellTarget(who) {
    stop(); // same as who.stop();
}
```

See Also
with

4.6.8.11.240 text

Player Required
SWF4 or later

Syntax
TextField.text

Description
Property: Indicates the current text in the Text Field. Lines are separated by the carriage return character ('\r', ASCII 13). This property contains the normal, unformatted text in the Text Field, without HTML tags, even if the text field is HTML.

This property allows access to the text of a Script Text Field Object.

Sample

```
txt.text = "this is new text";        // change the text in txt to be "this is the new text"
```

See Also
Text Field Properties and Methods and TextField Access Matrix for more information about the properties that can be accessed.

4.6.8.11.241 TextField (Object)

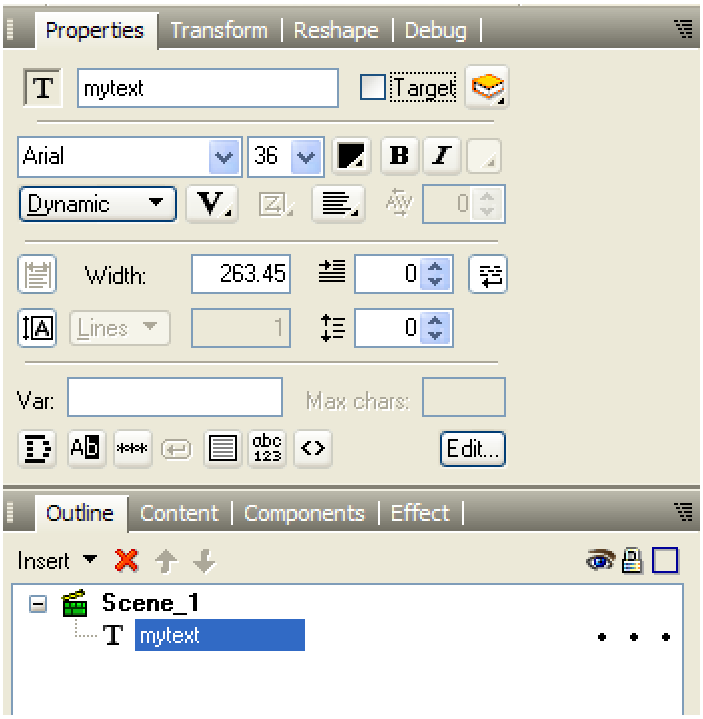
Dynamic Text Objects can either be a Simple Text Field Object or a Script Text Field Object.

See Also
createTextField()

4.6.8.11.241.1 Simple Text Field Object

These are created when Dynamic (or Input) Text is selected and the 'Target' checkbox is left unchecked.

When exporting to SWF4 or SWF5, an associated string variable, with the same name as the Text Object, is automatically created for these Objects to hold the text content of the Text Field. For SWF5 and later you can choose to specify the associated variable name using the Var: field in the Text Properties panel.



Text can be read and written to and from the Text Field via the associated variable. For SWF6 and later you can also access Text Field content directly without using an associated variable.

Sample (SWF4 or SWF5 export only)

```
trace(mytext);          // displays the text currently associated with the text field.
mytext = "this is some different text";      // assigns new text to the Text Object
```

Sample (SWF6 or later export)

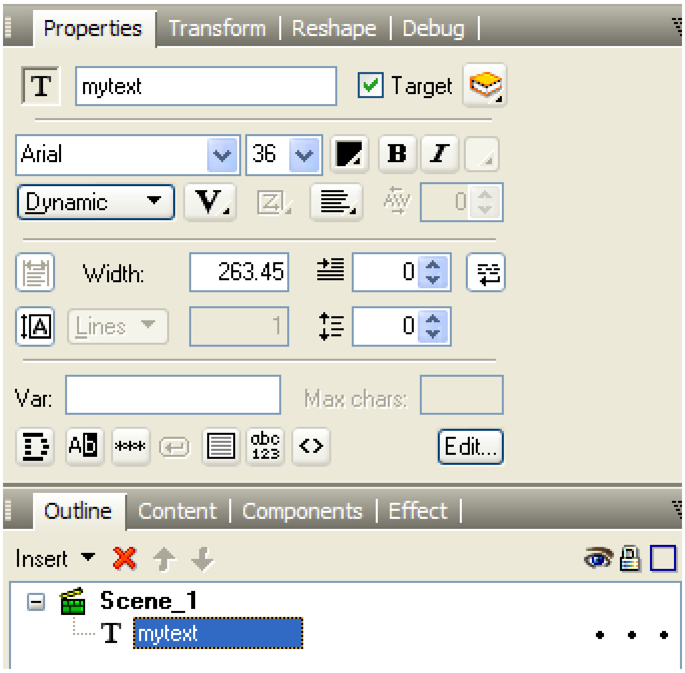
```
trace(mytext.text);     // displays the text currently associated with the text field.
mytext.text = "this is some different text"; // assigns new text to the Text Object
```

For SWF6 or later export, all of the SWF6 TextField Properties and Methods can be accessed directly without altering swf export options or using the _text property. For SWF5 export, only the 'scroll' and 'maxscroll' properties are available. For SWF4 export you cannot access these properties unless the 'Target' checkbox is set.

See the TextField Access Matrix for more information about the properties that can be accessed.

4.6.8.11.241.2 Script Text Field Object

These are created when Dynamic (or Input) Text is selected and the 'Target' checkbox is checked. This makes the TextField object a Scripting Object.


Properties | Transform | Reshape | Debug
Edit...
Outline | Content | Components | Effect
Insert

T mytext ☑ Target

Arial 36

Dynamic 0

Width: 263.45 0

Lines 1 0

Var: Max chars:

Scene_1

T mytext

Access to and from the text is via the .text property unless a value is quoted for the Var: field.
e.g. mytext.text returns the current text, mytext.text = "new text" sets the text.

Access to and from the text can also be via an associated variable defined in the Var: field. The associated variable must have a different name to that of the Object.

Access to maxscroll and scroll is via mytext._text.maxscroll and mytext._text.scroll. The TextField Access Matrix shows all of the permutations.
Access to the other TextField properties can be obtained via the _text property or the expose SWF6 properties checkbox in the Movie Export Options.

4.6.8.11.241.3 TextField Properties and Methods

The table below lists the properties and methods that are available with a TextField object.
Note that if the TextField object is a Script Object, then access to these properties and methods is via the _text property.

Property / Method	Flash Player	Notes
addListener method	6+	Registers a event listener object to receive event notifications.
_alpha property	6+	Read / Write property associated with the alpha transparency value of the text field.
antiAliasType property	8+	The type of anti-aliasing used by this text field.
autoSize property	6+	
background property	6+	
backgroundColor property	6+	
border property		
borderColor property	6+	
bottomScroll property	6+	
condenseWhite property	6+	

embedFonts property	6+	
filters property	8+	
getDepth method	6+	
getFontList method	6+	
getNewTextFormat method	6+	
getTextFormat method	6+	
getFitType property	8+	
_height property	6+	
_highquality property	7+	
hscroll property	6+	
html property	6+	
htmlText property	6+	
length property	6+	
maxChars property	6+	
maxhscroll property	6+	
maxscroll property	6+	(Read Only) Indicates the line number of the top-most visible line of text in a Text Field when the bottom-most line in the field is also visible
menu property	7+	
mouseWheelEnabled property	7+	
multiline property	6+	
_name property	6+	
onChanged event method	6+	
onKillFocus event method	6+	
onScroller event method	6+	
onSetFocus event method	6+	
_parent property	6+	
password property	6+	
_quality property	6+	
removeListener method	6+	
replaceSel method	6+	
replaceText method	7+	
restrict property	6+	
_rotation property	6+	
scroll property	6+	Controls the scrolling of information in a Text Field associated with a variable
selectable property	6+	
setNewTextFormat method	6+	
setTextFormat method	6+	
sharpness property	8+	
_soundbuftime property	6+	
styleSheet property	7+	
tabEnabled property	6+	
tabIndex property	6+	
_target property	6+	
text property	6+	Allows access to the text associated with the Object
textColor property	6+	
textHeight property	6+	
textWidth property	6+	
thickness property	6+	
type property	6+	
_url property	6+	
variable property	6+	
_visible property	6+	
_width property	6+	
wordWrap property	6+	
_x property	6+	
_xmouse property	6+	
_xscale property	6+	

_y property	6+
_ymouse property	6+
_yscale property	6+

Example

Assuming the text object is not a Script Object the following script can be used to change the font size. If the text object is a Script object then the property my_text._text must be used.

```
var my_format = new TextFormat();     // create text format object.
my_format = my_text.getTextFormat();  // get the existing text format.
my_format.size = 20;   // change font to 20 point.
my_text.setTextFormat(my_format);
```

See Also

TextFormat() Script Object

4.6.8.11.241.4 TextField Events

Apart from the standard Events associated with any Scripting Object, Text Field Objects can use the onChanged() Event as well as other on...() Event Methods defined in the Properties and Methods description.

See Also

Text Field Object, on(changed), onSelfEvent(changed), onChanged

4.6.8.11.241.5 TextField Access Matrix

Various properties associated with the Text Field object are accessed in different ways depending on the following:

- SWF Export version: 4, 5, or 6+
- If Expose SWF6 Properties is set
- If the Object is a Script Text Field Object (ie the Target checkbox is set).
- If the Object has an Associated Variable (Var: field is set).

The following matrix assumes that the text object is named myfield and the associated variable (if any) is named myvar.

SWF Export version	4	4	4	5	5	5	5	6+	6+	6+	6+	6+	6+	6+	6+
Expose SWF6 Properties	-	-	-	-	-	-	-	n	n	n	n	y	y	y	y
Target Checkbox set	n	y	y	n	n	y	y	n	n	y	y	n	n	y	y
Associated Variable defined (*myvar*)	n	n	y	n	y	n	y	n	y	n	y	n	y	n	y
Object *(1)*:															
myfield		**Xa**	**Xa**		**P**	**X**	**X**	**X2**	**X2**	**X**	**X**	**X2**	**X2**	**X**	**X**
Text field *(2)*:															
myfield								**X2**	**X2**			**X2**	**X2**		
myfield_txt				**P**				**Xd**				**Xd**			
myfield._text						**O**	**O**			**X**	**X**			**X**	**X**
Access to Text via *(3)*:															
myfield	**X**			**X**				**Xd**				**Xd**			
myfield.text		**X**	**X**			**X**	**X**	**X2**	**X2**	**X**	**X1**	**X2**	**X2**	**X**	**X**
myfield_txt.text								**Xe**				**Xe**			
myfield._text.text		**X2**	**X2**			**X2**	**X2**			**Xa**	**Xa**			**Xa**	**Xa**
myvar			**X**		**X2**		**X**		**X2**		**X**		**X2**		**X**
Scroll variables scroll / maxscroll *(4)*:															
myfield.scroll		**X**	**X**	**X2**		**X1**	**X1**	**Xa**	**X2**	**X1**	**X1**	**Xa**	**X2**	**X**	**X**
myfield_txt.scroll								**Xb**				**Xe**			
myfield._text.scroll		**X2**	**X2**							**Xc**	**Xc**			**X**	**X**
myfield.text.scroll						**X2**	**X2**			**X2**				**Xb**	**Xb**
myvar.scroll					**X2**				**X2**		**X2**		**X2**		**X2**

Other properties and methods *(5)*:															
myfield.var								X2	X2			X2	X2	Xa	Xa
myfield_txt.property								Xe				Xe			
myfield._text.var										Xa	Xa			Xa	Xa

X = supported in both Max1 and Max2
X1 = supported in Max1 only.
X2 = supported in Max2 only.
Xa = supported in both but Max1 via external flash player.
Xb = supported in Max1 only via external flash player.
Xc = supported in Max2 only. Will appear to play in Max1 via internal player but not when exported to .swf for external player.
Xd = supported in Max1, Max2 will convert .swi file so that "_txt" is appended to the name and the original max1 name is used as the associated variable.
Xe = supported in Max1only via external flash player, Max2 will convert .swi file so that "_txt" is appended to the name and the original max1 name is used as the associated variable.
P = Max2 only, accesses the parent of the object
O = Max2 only, access the object instead of the text field.

(1) this is the TextField object if the target is not set, or the wrapping object if the target checkbox is set.
(2) this is how to access the TextField object.
(3) this provides access to read and modify the dynamic text.
(4) although only scroll is shown, other properties such as maxscroll can also be accessed if supported by the SWF export version.
(5) these are the TextField properties and methods.

If creating .swi files in max1 to be compatible with both max1 and max2, then use options that are marked X or Xa or Xd.

4.6.8.11.242 TextFormat (Object)

Player Required
SWF6 or later

Syntax
var my_fmt = new TextFormat();

Arguments
None

Returns
New TextFormat Object.

Description
The new object can be loaded with the current format of existing text via the getTextFormat() method.
The object can then be modified (eg change font etc).
Then re saved to the text field via the setTextFormat() method.
See Properties for a description of the properties that can be modified.

See Also
TextFieldObject
Text Field Properties and Methods

NOTE:
If using this object to alter the text format of a static Script Text Field Object use the _text property.

Text field objects that are created dynamically using the movie clip.createTextField() function can be accessed directly without the use of the _text property.

For Example:
Use the following script to alter various format items of the a Text Field Object named mytext.

```
onSelfEvent (load)
{
    var my_fmt = new TextFormat();
    my_fmt = mytext._text.getTextFormat();

    my_fmt.align = "center";
    my_fmt.bold = true;
    my_fmt.color = 0xFF0000;
    my_fmt.font = "Comic Sans MS";
    my_fmt.size = 14;
    my_fmt.italic = true;

    mytext._text.setTextFormat(my_fmt);
}
```

4.6.8.11.242.1 Properties

The Text Format object has the following properties:

Property	**Type**	**Description**
align	"right", "center", "left", "full"	Sets the text alignment
bold	boolean	If true show text as bold
color	color	Sets the color of the text based on RGB number
font	string	string defining the font. eg. "Arial"
indent	number	Indentation from left margin. This only applies to 1st character in the paragraph.
italic	boolean	If true show text in italics
leading	integer	number representing the vertical space between the lines. From flash 8+ a negative number can be used to move the lines closer together.
leftMargin	number	Left margin of the paragraph in points.
rightMargin	number	Right margin of the paragraph in points.
size	number	Size of the text in points

4.6.8.11.243 this

Player Required
SWF4 or later

Syntax
this.property
this.property = value

Arguments
property: Any property of the current Object.
value: New value of the property.

Description
Synonym for the current Movie Clip / Sprite / Object that contains this script element.
An Event Handler for the Object can use this synonym to access any property associated with the Object that called the method.

This is extremely useful when Instances of a Movie Clip / Sprite are made. If the Movie Clip / Sprite makes reference to this (rather than via Object name) then each Instance will be self-contained and independent.

Sample

```
this.stop();   // stop the current Movie Clip / Sprite
```

See Also

_parent and _root.

4.6.8.11.244 Timer.ticks()

Player Required

SWF4 or later

Syntax

Timer.ticks

Arguments

None.

Returns

Milliseconds since the Movie started.

Description

Function: Returns the number of milliseconds that have elapsed since the Movie started playing. Can be used to provide accurate timing between Events that are not dependent on the Frame Rate.

Sample

```
onFrame (20) {
    trace (Timer.ticks());    // displays number around 1025 (frame rate was 20 fps)
}
```

4.6.8.11.245 toggleHighQuality()

Player Required

SWF4 or later

Syntax

toggleHighQuality()

Arguments

None.

Returns

Nothing.

Description

Action: Turns anti-aliasing on and off in Flash Player. Anti-aliasing smooths the edges of Objects and slows down the Movie playback. The toggleHighQuality Action affects all Movies in the Flash Player.

Sample

The following code could be applied to a button that, when clicked, would toggle anti-aliasing on and off:

```
on(release) {
    toggleHighQuality();
}
```

See also

_highquality

4.6.8.11.246 trace(expression)

Player Required
SWF4 or later

Syntax
trace(expression)

Arguments
expression: Can be numbers, strings or variables.

Returns
Nothing.

Description
The result of the expression is displayed in the 'Debug' window. Syntax errors in the script will disable all Debug display.
Arrays are displayed in the debug panel as each array element separated by a comma.

4.6.8.11.247 true

Player Required
SWF5 or later

Syntax
true

Description
A unique boolean value that represents the opposite of false.

4.6.8.11.248 typeof

Player Required
SWF5 or later

Syntax
typeof *expression*

Arguments
expression: Can be a Number, Boolean, String, Button, Movie Clip / Sprite, Object, or Function.

Returns
As described below:

Description
Operator; Tells the SWiSH Max debugger to evaluate the expression. The debug panel will display a string representing whether the expression is a Number, Boolean, Object, MovieClip (Sprite), Button, String, or Function.

Note: For SWF4, the typeof operator will always return "string" both internally and externally
Note: For SWF5+ internal player will get "object", "number", "string", "boolean" or "movieclip".

Sample

```
onFrame (1) {
```

```
myVar = "1";
trace(typeof myVar); // displays 'string' in the debug window
}

onFrame (1) {
myVar = 19;
trace(typeof myVar); // displays 'number' in the debug window
}
```

Assuming you have a Movie Clip named "myMovieClip" in your movie:

```
onFrame (1) {
    trace(typeof myMovieClip); // displays 'MovieClip' in the debug window
}
```

4.6.8.11.249 undefined

Player Required
SWF5 or later

Syntax
undefined

Description
A unique value that indicates an undefined value. This is normally associated with variables that have not been assigned a value.

For SWF6 or earlier String(undefined) returns "", Number(undefined) returns 0
For SWF7 and later String(undefined) returns "undefined", Number(undefined) returns NaN

Example

```
// if the variable x does not exist, "undef" will be displayed in the debug panel.
if (undefined == eval("x"))
{
        trace("undef");
}
else
{
        trace(eval("x"));
}
```

4.6.8.11.250 unloadMovie()

Player Required
SWF4 or later

Syntax
MovieClipName.unloadMovie()

Arguments
MovieClipName: Name of the Movie Clip / Sprite that is removed.

Returns
Nothing.

Description
Action or Movie Clip Method: Removes a loaded Movie or a Movie Clip from the Flash Player.

Sample
In the following unloadMovie statement to the onSelfEvent (load) Event of a Movie Clip / Sprite, the Movie previously in mc1 is unloaded.

```
onSelfEvent (load) {
    mc1.unloadMovie(); // unload mc1
}
```

See also
loadMovie, loadMovieNum, unloadMovie, unloadMovieNum, loadVariables and loadVariablesNum.

4.6.8.11.251 unloadMovieNum()

Player Required
SWF4 or later

Syntax
unloadMovieNum(level)

Arguments
level: The level of the Movie to be unloaded.

Returns
Nothing.

Description
Action: Removes a loaded Movie or a Movie Clip from the Flash Player.

Sample

```
unloadMovie(5);        // unload Movie that was resident in level 5.
```

See also
loadMovie, loadMovieNum, unloadMovie, loadVariables and loadVariablesNum.

4.6.8.11.252 value.toString()

Player Required
SWF5 or later

Syntax
toString(*value*);
value.toString();
toString(*value*,*base*);
value.toString([*base*]);

Arguments
value; a variable representing a numerical value.

base; the numeric base used for the conversion from number to string. The *base* is a number between 2 and 36 and defaults to a value of 10 if this argument is not specified.

Returns
A String representing the value.

Description
Method; returns a string representing the specified value.

Sample

```
onSelfEvent (load) {
    a = 7 + 6;
    trace(a.toString());
}
// displays "13" in the debug window

onSelfEvent (load) {
    a = 7 + 6;
    trace(a.toString(2));
}
// displays "1101" in the debug window
```

4.6.8.11.253 var

Player Required
SWF5 or later

Syntax
var variableName [: typeName1] [= value1] [...,variableNameN [: typeNameN] [= valueN]]

Arguments
variableName An identifier.

typeName A type keyword, one of String, Number of Boolean

value The value assigned to the variable.

Returns
Nothing.

Description
Action: used to declare local or Timeline variables.

If you declare variables inside a function, the variables are local. They are defined for the function and expire at the end of the function call. Otherwise the variables are interpreted as Timeline variables. However, you don't have to use var to declare Timeline variables.

The var statement supports optional types. These types may be used by the <%SWISHSCRIPT%> compiler to test whether the values assigned to variables are of the expected type, and that the variable is not used where a different type of value is required. Types can include Number, Boolean or String. The type checking may not occur in some cases.

Sample:

```
function f ( x ) {
    var y = x +1;
    return y*y;
}

// the variable y is local to the function
```

See Also
const

4.6.8.11.254 variable(string)

Player Required
SWF5 or later

Syntax
1. variable(string); or
2. variable(string) = expression;

Arguments
string: A string containing the name of a variable.
expression: Any valid expression.

Returns
Value of the variable defined by the string (mode 1).

Description

1. Function: Accesses variables, by name. If the expression is a variable or a property, the value of the variable or property is returned. In this mode it is equivalent to eval().

2. Function: Allows the variable specified within the string to be assigned the value defined by the expression.

Sample

The following example uses the eval function to determine the value of the expression "piece" + x. Because the result is a variable name, piece3, the eval function returns the value of the variable and assigns it to y:

```
Item5 = "test";
x = 5;
y = variable("Item" add x);
trace(y); // Output: test this demonstrates function 1
variable(Item5) = "hi"
trace(test);   // output: "hi", new variable test is created. This demonstrates function 2
```

See Also

eval()

4.6.8.11.255 void

Player Required

SWF5 or later

Syntax

void

Description

Evaluates the expression, but ignores the result. It is included only for compatibility with Flash Actionscript.

4.6.8.11.256 while()

Player Required

SWF4 or later

Syntax

```
while (condition) {
        statements;
}
```

Arguments

statements: The statements that are to be executed while the condition is true.
condition: An expression that evaluates to a boolean value.

Returns

Nothing.

Description

This Action executes and repeats while the condition remains true. If the condition is initially false, no statements are executed.

Sample

```
onSelfEvent (load) {
    a = 1;
    while (a++ < 5) {
        trace(a);
    }
}
```

The above example writes the numbers 2 3 4 5 to the 'Debug' window.

See Also
do...while and for.

4.6.8.11.257 with()

'with' in Flash is identical to tellTarget in SWiSH Max.

Note: In Flash, a 'with' statement does not apply statements to the target. Instead, whenever you refer to a variable, method, or Clip by name, the nominated Object is searched first. If no corresponding name is found, then the search continues the same as outside the 'with'. For example:

```
with (xxx) {
        with (yyy) {
            zzz = 10;
        }
}
```

This will search the yyy Object for a variable called zzz. If it is found, it will set if to 10, if not, then it will search the xxx Object for variable 'zzz', etc.

See Also
tellTarget

4.6.8.12 Useful Stuff

This section includes some standard mathematical formulas that may be useful. These formulas can often be used to expand the power of the Math Object.

4.6.8.12.1 Converting Log Base

If the logarithm of a different base is required, use the following formula:

log*N(x)* = ln(x)/ln(*N*)

e.g. To calculate the log10(5) use Math.log(5) / Math.log(10) (note Math.log() is ln()).

To calculate x^y (here ^ represents power operator).

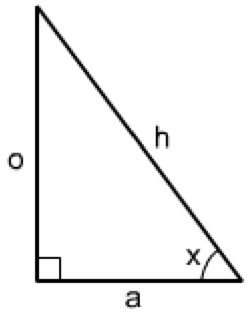
x^y = e ^ (y * ln(x)) where e is the base of the natural logarithm.

e.g. To calculate 10 ^ 2 use Math.exp(2 * Math.log(10)) .
Note Math.exp() is e ^ n.

To calculate the square root of a number, use x^y where y = 1/2 or 0.5.
e.g. Math.exp(0.5 * Math.log(3)) returns the square root of 3.

4.6.8.12.2 Simple Trigonometry

Sin, Cos and Tan are functions that return the ratio of the sides of a right angled triangle if the angle is specified.



For the triangle above, sin(x) = o/h, cos(x) = a/h and tan(x) = o/a

The sides are named **'o'** meaning the side opposite the angle, **'a'** meaning the side adjacent to the angle and **'h'** meaning the hypotenuse.

The asin(), acos() and atan() are the inverse functions of the ones above.
ie. asin(o/h) = x, acos(a/h) = x and atan(o/a) = x

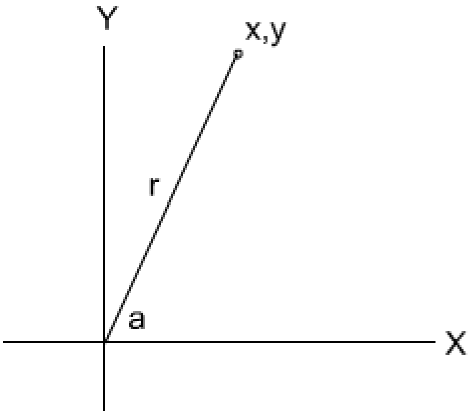
sin(), cos(), tan() asin(), acos() and atan() all work where the angle (x) is specified in radians.
the functions with deg in the name, eg. sindeg() work where the angle (x) is specified in degrees.

It is possible to convert from radians to degrees as follows:
360deg = 2*pi radians.

so:

d = 2*pi*radians/360 and
radians = d*360/(2*pi)

It is sometimes necessary to convert between **Polar** and **Rectangular** co-ordinate systems.



In a Rectangular co-ordinate system a point is described by its x,y co-ordinates. This system provides easier calculation of point translation and vector addition / subtraction.

In a Polar co-ordinate system the same point is described by its radius (r) and angle (a) from the origin (0,0). This system has the advantage that rotation calculations are easier. ie r stays the same and the angle is altered as necessary.

To convert from Rectangular to Polar co-ordinates:
r = sqrt(**x**^2 +**y**^2) Pythagorean Theorem
a = atan(**y**/**x**) will return the angle in radians or atandeg(**y**/**x**) if you want the angle expressed in degrees.
Note if **x** is 0, **a** is pi/2 radians (90 degrees) if **y** is positive or -pi/2 radians (-90 or 270 degrees) if **y** is negative.
If both **x** and **y** are 0 then **r** is 0 and the angle **a** is undefined. 0 can be used as a default setting in this instance.

To convert from Polar to Rectangular co-ordinates:
x = **r** * cos(**a**) if **a** is specified in radians or **r** * cosdeg(**a**) if **a** is specified in degrees.
y = **r** * sin(**a**) if **a** is specified in radians or **r** * sindeg(**a**) if **a** is specified in degrees.

4.6.8.12.3 Transformation Matrix

Flash gradient fills often make use of a matrix that defines the boundary conditions of the gradient.

2x3 gradient matrix a, b, c, d, tx, ty expressed as a 3x3 matrix. This matrix is used in the Fill Object. Once expressed as a 3x3 matrix it can be multiplied with the following Transformation matrix.	$\begin{pmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1 \end{pmatrix}$
3x3 gradient matrix used by beginGradientFill()	$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$
Multiplication by this matrix will translate the fill by ***tx, ty***	$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$
Multiplication by this matrix will scale the fill by ***Sx, Sy***	$\begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Multiplication by this matrix will skew the matrix by ***kx, ky***	$\begin{pmatrix} 0 & ky & 0 \\ kx & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Multiplication by this matrix will rotate the fill by the angle ***a***	$\begin{pmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

4.6.8.12.4 Matrix Multiplication

In order to multiply two 3x3 matrix consider the diagram below:

$$\begin{pmatrix} Aa & Ab & Ac \\ Ad & Ae & Af \\ Ag & Ah & Ai \end{pmatrix} * \begin{pmatrix} & Ba & Bb & Bc \\ \Bigg\downarrow & Bd & Be & Bf \\ & Bg & Bh & Bi \end{pmatrix} = \begin{pmatrix} Ca & Cb & Cc \\ Cd & Ce & Cf \\ Cg & Ch & Ci \end{pmatrix}$$

The result of the multiplication has the following values:
Ca = Aa*Ba + Ab*Bd + Ac*Bg
Cb = Aa*Bb + Ab*Be + Ac*Bh
Cc = Aa*Bc + Ab*Bf + Ac*Bi
Cd = Ad*Ba + Ae*Bd + Af*Bg
Ce = Ad*Bb + Ae*Be + Af*Bh
Cf = Ad*Bc + Ae*Bf + Af*Bi
Cg = Ag*Ba + Ah*Bd + Ai*Bg
Ch = Ag*Bb + Ah*Be + Ai*Bh
Ci = Ag*Bc + Ah*Bf + Ai*Bi

4.6.9 Error Messages

When a movie is played in the internal player, error message may be displayed in the Debug panel. The more common error messages and a description of how to solve the error is given below.

"ERROR: Cannot find '<name>'"
The variable has not been declared or initialized. Although this is acceptable under SWF4 this will cause errors in later flash player versions. It is recommended that variables be defined via a var statement before use or be initialized with a value during the onSelfEvent (load) event.

"<name> can't be shown because it is rotated by N° and skewed by N°"
Device fonts cannot be displayed if rotated or skewed. Either choose a non device font, or reset rotation and skew angle to 0. Note that if the font is used within a movie clip then rotation / skewing of the movie clip can cause this problem.

"This movie has script that is too long or complex in '<name>'"
AS1/2 SWF format has a limit to the size of code that can exist within a code section. A code section is generally the code within a pair of {}.
The error means that a single section of code is too long. To fix this error simplify and / or break your code into smaller sections. Code can be broken into smaller sections by placing sections of code within a user defined functions and replacing repetitive operations by loops. Good coding practice suggests that any single function should not exceed one page in length.

Note: Comments do not affect the code size and should be left in place. ie removing comments will not fix this error.

4.6.10 Code Section Limit

Flash uses a 16bit counter to represent code offsets. Consequently the largest compiled code section that can be represented within conditional code such as branches and loops is 32767 bytes. This script size limitation occurs for functions, conditional statements, loops, with and try statements, basically any piece of code that starts with { and ends with }. See also Debug Panel. This error can be avoided through the use of appropriate Code Structure.

NOTE: Comments are not compiled and will therefore not affect the code section sizes.

Basic Working Procedures

Chapter 5

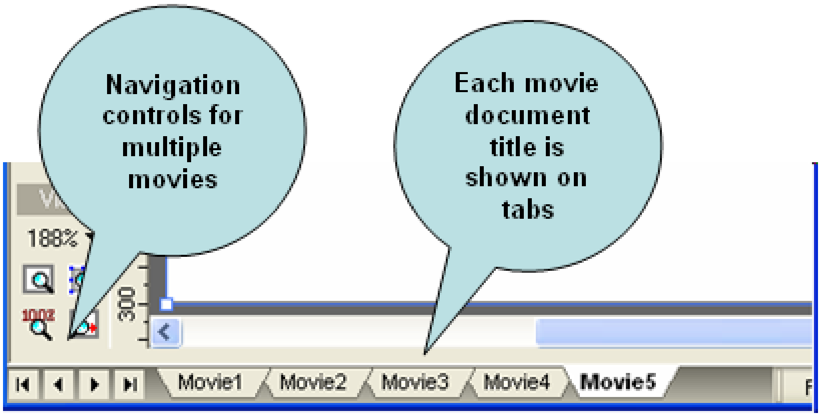
5 Basic Working Procedures

This section describes the most common basic working tasks in SWiSH Max. This section gives a brief overview of how to open, save and publish from SWiSH Max. This section is intentionally brief. Also review the Tutorials to become familiar with the operation of SWiSH Max.

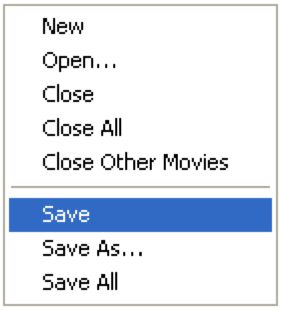
- Creating and managing Movies
- Managing Windows and Panels
- Exporting and Publishing
- Working with the Library
- Templates

5.1 Creating and managing Movies

A SWiSH Max document is called a 'Movie'. The document file has a .swi extension (pronounced "SW-eye"). Select File | New to create a new movie file (.swi), or open an exisitng file. Multiple movies can be opened and will show on the tabs. Click on a tab to select the active movie.

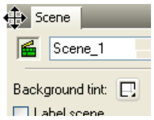

Navigation controls for multiple movies
Each movie document title is shown on tabs
188%
300
Movie1
Movie2
Movie3
Movie4
Movie5

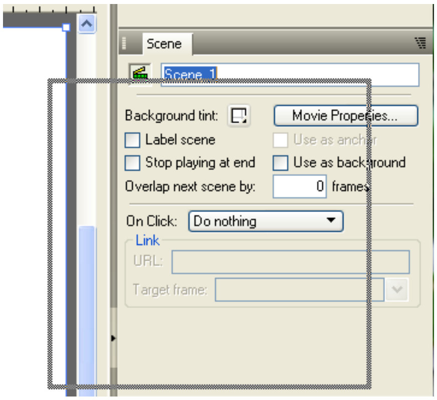
Right click on any tab to get a properties dialog to manage the open movies.



5.2 Managing Windows and Panels

Click on the Panel Grip icon to grip and drag each panel from it's docked position. When dragged from it's docked position (whilst the left mouse button is held down) the new position is represented by an border. Drag the pointer position to the edge of existing panels and the panel border will snap to possible dock positions.





Scene_1

Background tint:

□ Label scene

□ Stop playing at end

Overlap next scene by: 0 frames

□ Use as anchor

□ Use as background

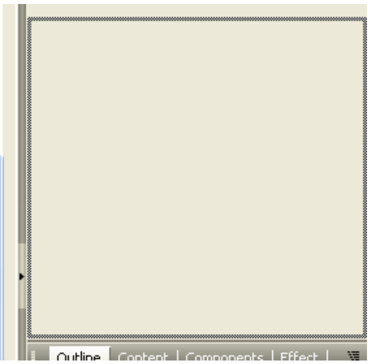
On Click: Do nothing

Link

URL:

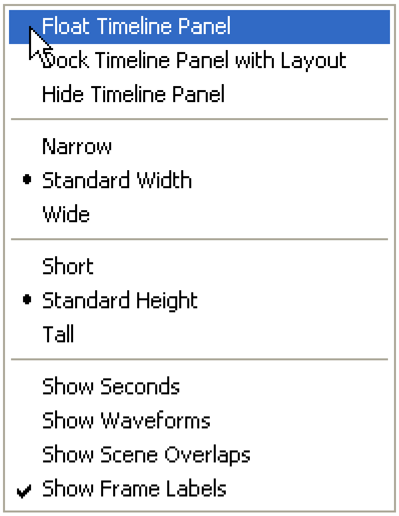
Target frame:

A dragged panel

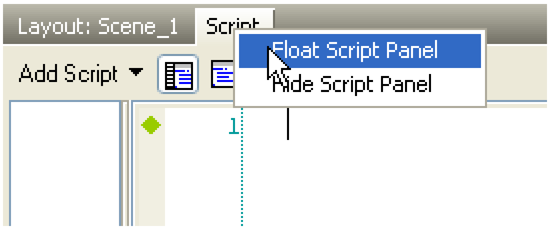


A floating panel snapping to a dock position

Alternatively right-click on the title bar of the panel to see a Panel Menu.



Timeline Panel Menu



Script Panel Menu

All panels can float, including the script panel. Floating panels onto a second monitor and saving this layout is a good way to expand the stage work area

The following hot keys will allow the user to quickly hide all the panels and maximize the space available for the stage. The hot-keys are the same as those used in other industry standard applications (e.g. Adobe® products):

- **Tab** hides/shows all panels and toolbars except the main view
- **Shift-tab** hides/shows all panels except timeline, toolbars and main view

5.3 Exporting and Publishing

To publish from SWiSH Max it is necessary to export to another file format (in contrast to saving to the .swi document format).

SWiSH Max is able to export four file formats:
Flash file format (.swf): The file the Flash player reads. You can also choose to export the HTML page the browser uses to display the SWF movie

- **Video file format (.avi):** A video format which can be played by (for exampl) the Windows Media Player™ .
- **Flash Player Projector (.exe):** A stand-alone executable files .
- **Animated GIF (.gif):** A GIF format file which contains a set of images in the specific order of the animation. This will only playback an animation, since GIF does not support interaction.
- PNG **images (.png):** Exports a sequence of PNG image files with a filename prefix indicating the position in the sequence. This option can be useful to get an animation into an application which imports image sequences.

To display a Movie on the web, it is necessary to upload the .swf to server. Although a .swf can be played over the web without being placed within a web page, it is typically embedded within one, so we advise inexperienced users to upload the HTML page as well.

Helpful hints when creating Movies

Flash has quickly become the technology of choice for creating rich, interactive, multimedia content for the web. Three factors that have greatly contributed to the technology's popularity are:

1. the .swf format can compress both images and sounds
2. the format allows for the inclusion/use of vector formats, scalable graphics with very small file sizes
3. the Flash player's streaming capabilities, which allow preloading images and sounds into the visitor's browser cache, and animation to stream while playing, rather than making the visitor wait for it to fully download.

To ensure visitors view the Movies correctly and without pauses in the flow of the animation, ask them to wait while a portion of the Movie has a chance to preload. The larger the size of the .swf file, the longer the viewer has to wait.

There are a number of things which can be done to keep file sizes smaller and so to minimize the time a visitor has to wait for the Movie to begin. Images, music, sound effects and animation effects all add to the file size of the .swf and can quickly increase the file size. For this reason, it is wise to carefully consider each of the files added the Movies and keep the original sizes to a minimum. SWiSH Max will also compress both images and sounds.

Don't forget that there is a lot of support available through the growing community of Support Forums for SWiSH Max.

5.4 Working with the Library

The Library is used to accumulate and share common resources throughout the Movie. The Library can contain Symbols and Resources. Resources can be shared between a number of different Objects. The contents of the Library are called 'Library Items'.

Some Definitions

The **Library** is a collection of Symbols and Resources.
A **Library Item** is general description of any Symbol or Resources in the library.
A **Symbol** is an Object which is stored in the library (eg. Shape Object).
A **Resource** is any information stored in the library which is ***not*** a Symbol (eg. an image, color, gradient).
An **Instance** is a shared use of a library Symbol.
A **Linked Resource** is a shared use of a library resource.

The following topics discuss the Library and how to manage Library items:
Adding to the Library
Using Library Items
Instances and Symbols
Editing Symbols

5.5 Working with Templates

SWiSH Max uses two types of Templates:

- SWI Templates and
- HTML Templates

SWI Templates are SWiSH Max Movies that can be used as a starting point to create new Movies. A SWI Template can be saved with features that are commonly required when starting a new Movie. Templates can also be built by others and shared.
A template can be used to define the movie properties such as Background Color, Width, Height, Frame Rate and general Export Settings. In addition the template also defines the default scene properties such as Background tint, Label scene, Stop playing at end, Use and anchor, use as background, Overlap next scene by... and On Click:.

Upon starting SWiSH Max the user is presented with a Start Up Menu. One of the options in this menu is to create a new Movie 'New from Template'. It is also possible to open a Template by selecting File | New From Template. There are a number of Templates supplied with SWiSH Max.

Any SWiSH Max Movie (.swi) can be used as a SWI Template. New templates are created by selecting 'File | Save as Template', and are saved in the 'SWITemplates' folder in the application installation folder (e.g. C:\Program Files\SWiSH Max2\SWITemplates).

When a movie is exported with the File | Export | HTML + SWF option, a **HTML Template** is used as a basis for the HTML file that is created. The exact file that is used is defined via the File | Export Settings | HTML option.

The template folders and default templates can be altered using Tools | Preferences | Templates.

User Interface

Chapter 6

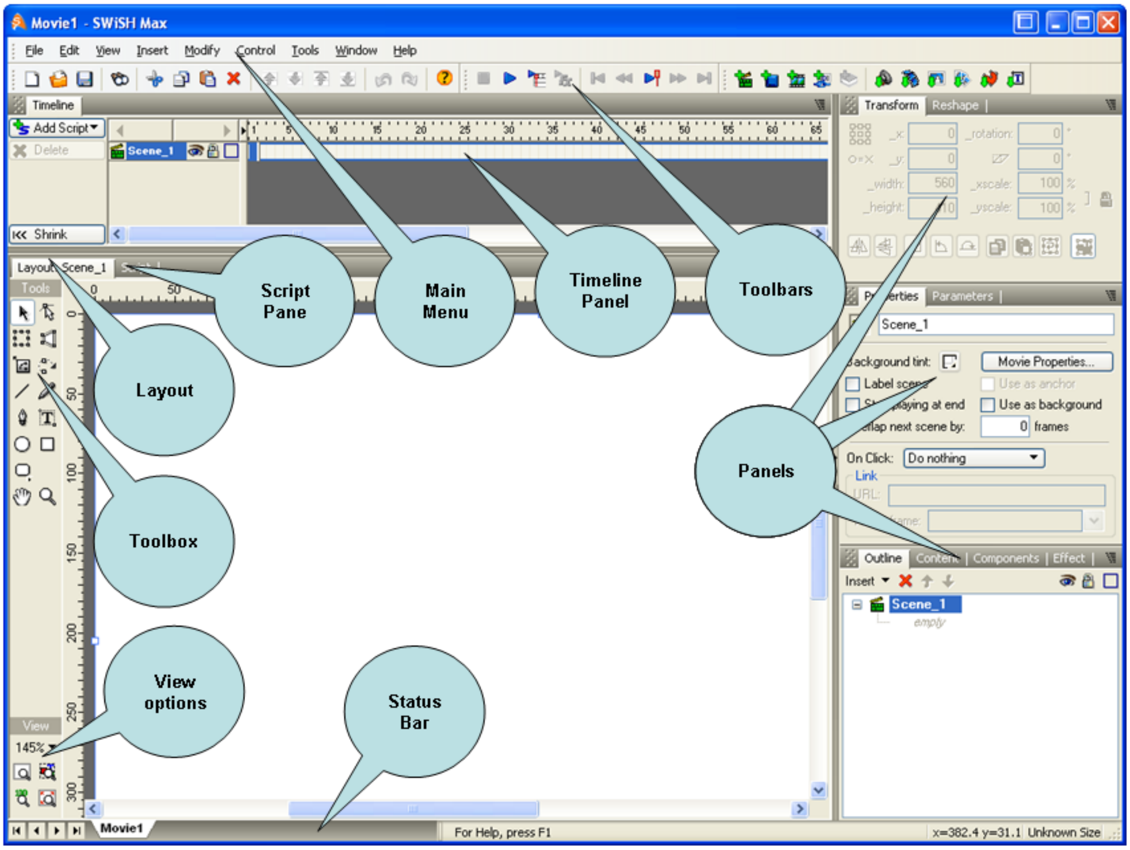
6 User Interface

When SWiSH Max is first started, it displays a Start Up Menu to allow selection of common tasks. After the Start Up Menu has been used or closed, the following interface is visible.

The SWiSH Max User Interface is made up of the following panes or panels:

- Main Menu
- Toolbox
- Layout and Script pane
- Timeline
- Toolbars including the Standard, Insert and Control Toolbars
- Panels including the Outline, Transform and Scene Panels among many others
- Status Bar
- View options.

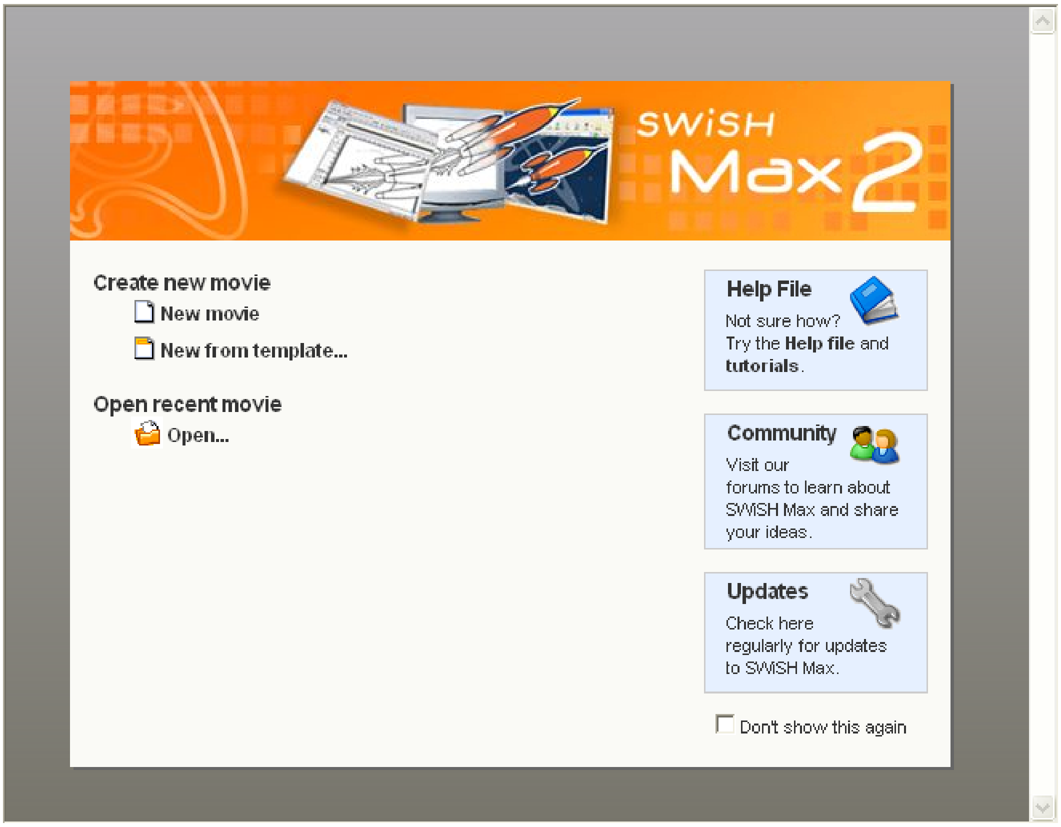
Each of these features are indicated in the illustration below.



6.1 Start Up Menu

This Menu is displayed when SWiSH Max is started. It allows quick selection of common SWiSH Max tasks.

Disable this Menu from the 'Don't show this again' checkbox, as shown below.



SWiSH Max 2
Create new movie
New movie
New from template...
Open recent movie
Open...
Help File
Not sure how? Try the Help file and tutorials.
Community
Visit our forums to learn about SWiSH Max and share your ideas.
Updates
Check here regularly for updates to SWiSH Max.
□ Don't show this again

Also disable or re-enable this Menu from the Tools | Preferences | Appearance 'Show startup menu' option.

6.2 Menus

The Main Menu is located at the top of the SWiSH Max application window. The Menus are:

- File
- Edit
- View
- Insert
- Modify
- Control
- Tools
- Window
- Help

Many of the Main Menu commands and options are also available from the Toolbars.

Note: In this manual, to describe pathways through the Menu system, the '|' character is used to indicate a selection on the next level of the Menu system. For example, File | Open... means the command can be accessed by selecting File from the Main Menu and then Open from the Submenu.

Shortcut keys shown as (key sequence) allow direct access to the command. For example, File | Open... (Ctrl + O) means the command can also be accessed by pressing the Control and O keys simultaneously.

If the command is available from a Toolbar, the associated icon is shown.

6.2.1 File Menu

The File Menu enables you to work with .swi files and Movies as a whole.

The available options are as follows:

Manage .swi files

- New
- Open...
- Save
- Save As...
- Save All
- Revert

Templates

- New from Template
- Save as Template
- Save as Default

Components

- Save as Component...

External files and testing

- Samples
- Import to Stage
- Import to Library
- Convert Video to FLV
- Export
- Export Settings
- Test

Recent files

- Recent Files

Closing windows and exiting

- Close
- Close All
- Exit.

6.2.1.1 New

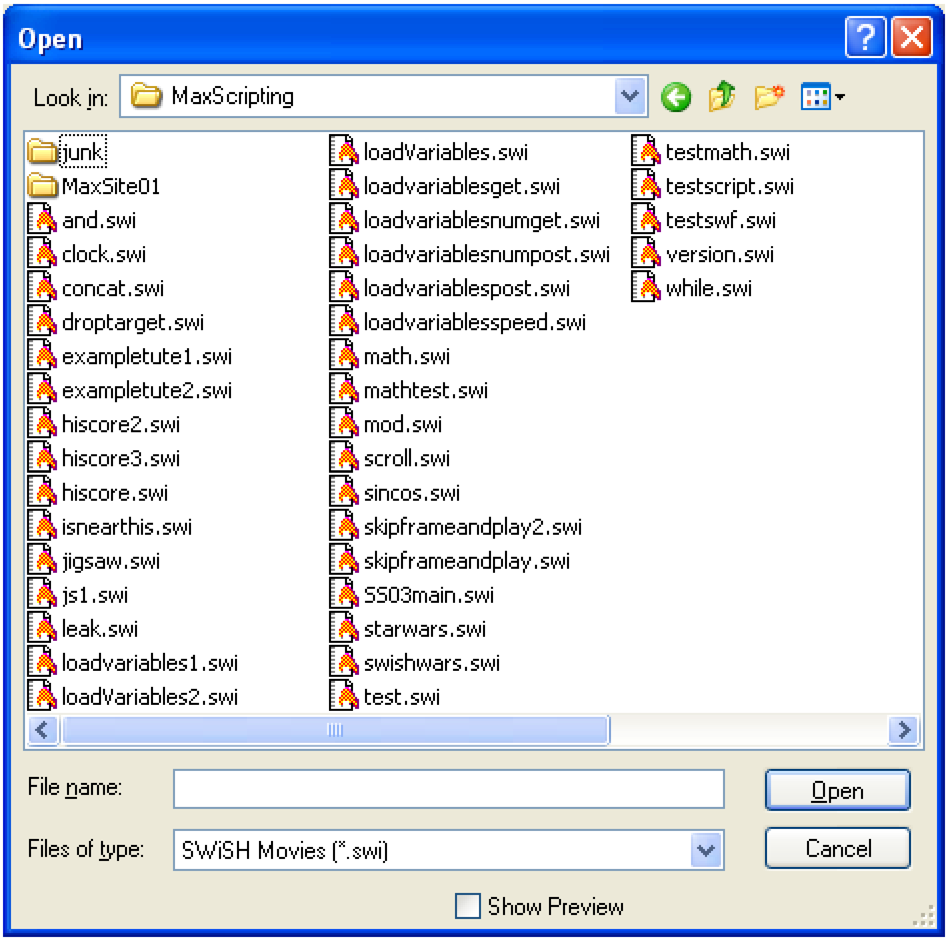
File | New (Ctrl+N)

Creates a new SWiSH Max Movie called "Movie*n*". A new Movie document tab will be created.

6.2.1.2 Open

File | Open... (Ctrl+O)
Opens a .swi (SWiSH Max Movie) file.

You can also use File | Open to create a new SWiSH Max Movie containing the imported file. To do this, choose a file with a type and extension that is supported by File | Import.



Show Preview
A preview of the currently selected .swi, .swf or image file is provided to allow you to see the Movie or image before opening it.

Note:

- SWiSH Max cannot open .fla files because it is not a public format
- If you want to open an existing .swf file and do not have the corresponding .swi file, use File | Import to import it into the current Movie

6.2.1.3 Save

File | Save (Ctrl+S)
Saves the current SWiSH Max Movie. If it is a new Movie, SWiSH Max will ask you for a name. There is no need to type in the .swi extension.

If the Auto Export option is set, every time you save your .swi file you will be prompted to save a corresponding .swf file. This way, the Save command also automatically saves a new .swf file for the Movie.

6.2.1.4 Save As

File | Save As... (Ctrl+Shift+S)

Saves the current SWiSH Max Movie under a new name. There is no need to type in the .swi extension.

6.2.1.5 Save All

File | Save All (Ctrl+Alt+S)

Saves all modified Movie documents. Unmodified Movie documents are not saved.

6.2.1.6 Close

File | Close (Ctrl+F4)

Closes the current SWiSH Max Movie. SWiSH Max will prompt the user to save the Movie before closing it.

6.2.1.7 Close All

File | Close All (Ctrl+Alt+F4)

Closes all current Movie documents. SWiSH Max will prompt the user to save any modified Movie before closing it.

6.2.1.8 Revert

File | Revert

Revert will discard any changes to your Movie and reload the previously saved file from disk.

6.2.1.9 New from Template

File | New from Template...

Creates a new SWiSH Max Movie based on a Template Movie.

See Working with Templates for more information on using Templates in your SWiSH Max Movies.

6.2.1.10 Save as Template

File | Save as Template...

Saves the current .swi file to the template area as a template .swi.

See Working with Templates for more information on using Templates in your SWiSH Max Movies.

6.2.1.11 Save As Default Template

File | Save as Default Template

Saves the current .swi file to the template area as the startup default .swi Movie.

See Working with Templates for more information on using Templates in your SWiSH Max Movies.

6.2.1.12 Save As Component...

File | Save as Component...

Saves the current .swi file to the component folder.

See Components for more information on using Components in your SWiSH Max Movies.

6.2.1.13 Samples

File | Samples

The Samples Menu contains a list of sample .swi Movies to illustrate some of the potential of SWiSH Max. Use the submenu to choose from either Beginner level files, Intermediate level files, Advanced level files, or the files used in the SWiSH Max Tutorials and Scripting Tutorials.

See an overview of sample files here.

6.2.1.14 Import

File | Import ...
Imports an external file into the current Scene or to the content Library. Multiple files may be selected for importing by holding the Shift or Ctrl key while selecting.

'Import to Library...' adds the content to the Library as a Resource or Symbol. If subsequently added to the stage, an appropriate Object is created which is either an Instance that links to the imported Symbol, or an Object that uses a link to the imported Resource.

'Import to Stage...' creates an appropriate Object for the content and places it on the Stage (Layout Panel).

The following sections describe the acceptable content file formats and what is created in each case (to stage or to library).

/ **Sound (also see Import Sound)**

- **to Stage**: Adds a Soundtrack to the current Scene/Movie Clip. To play a Sound as an event see more details at **Import Sound).**
- **to Library**: Adds a sound resource to the Library

Supported file formats:

- Wave files (*.wav)
- MP3 files (*.mp3)
- Windows Media Audio (*.wma)
- Other audio files (*.au;*.aif;*.aiff;*.snd)

Note:

- SWiSH Max supports DirectShow so if a filter (codec) is installed for a particular audio file SWiSH Max will be able to import that file type. As a general check if Microsoft® Windows Media Player® can play the file, SWiSH Max will be able to import it.
- When importing a sound, SWiSH Max will add the sound as a Soundtrack.
- MP3 sounds are imported without SWiSH Max processing it, so external programs can be used for optimum compression or quality

/ **Video (also see Import Video)**

- **to Stage**: Adds an Embedded Video Object including the inserted video frames, or a Video Component linked to the converted Flash Video, to the current Scene/Movie Clip
- **to Library**: Adds a video resource to the Library

Supported file formats:

- Adobe Flash Video (*.flv)
- AVI files (*.avi)
- Quicktime Video files (*.mov, *.qt)
- MPEG video files (*.mpg)
- Windows Media Video files (*.wmv, *.asf)

Note: Manipulation of the video file within SWiSH Max is processor (CPU) and memory intensive and you will notice degraded performance.

/ **Images (also see Import Image)**

- **to Stage**: creates a Shape Object on the current Scene/Movie Clip with the inserted image as a fill
- **to Library**: adds an image resource to the Library

Note: Raster images, for vector images see 'Vector Graphics' below.
Supported file formats:

- Windows bitmap (*.bmp; *.dib)
- GIF image (*.gif)
- JPEG image (*.jpg; *.if, *.jpeg)
- PNG image (*.png)

/ **Animation (also see Import Animation)**

- **to Stage**: creates a Movie Clip, Shape or Group Object with the animation frames on the current Scene/Movie Clip
- **to Library**: creates a Movie Clip, Shape or Group Object with the animation frames in the Library as a Symbol

Supported file formats:

- Animated GIF (*.gif) (imports as a frame sequence)
- Flash Animation (*.swf both compressed and uncompressed)
- Flash Projector (*.exe)
- SWiSH Max Movie (*.swi)

/ **Vector Graphics (also see Import Vector)**

- **to Stage**: creates a Shape or Group Object with the vector graphic on the current Scene/Movie Clip
- **to Library**: creates a Shape or Group Object with the vector graphic in the Library as a Symbol

Supported file formats:

- Windows Metafile (*.wmf)
- Enhanced Metafile (*.emf)
- Flash Graphics (*.swf both compressed and uncompressed)

/ **Text (also see Import Text)**

- **to Stage**: creates a Text Object in the current Scene/Movie Clip with the imported text
- **to Library**: creates a Text Symbol in the Library with the imported text

Supported file formats:

- Plain text (*.txt)

6.2.1.15 Test

File | Test | SWF in Flash Player (Ctrl+T)
Creates a temporary .swf file, launches the Flash Player and plays the current SWiSH Max Movie in the Flash Player.

Note: If the stand-alone Flash Player is not detected on your system, a projector executable installed with SWiSH Max is used.

File | Test | HTML + SWF in Browser (Ctrl+Shift+N)
Creates temporary .htm and .swf files, launches the default browser and displays the HTML page containing the Movie in the browser.

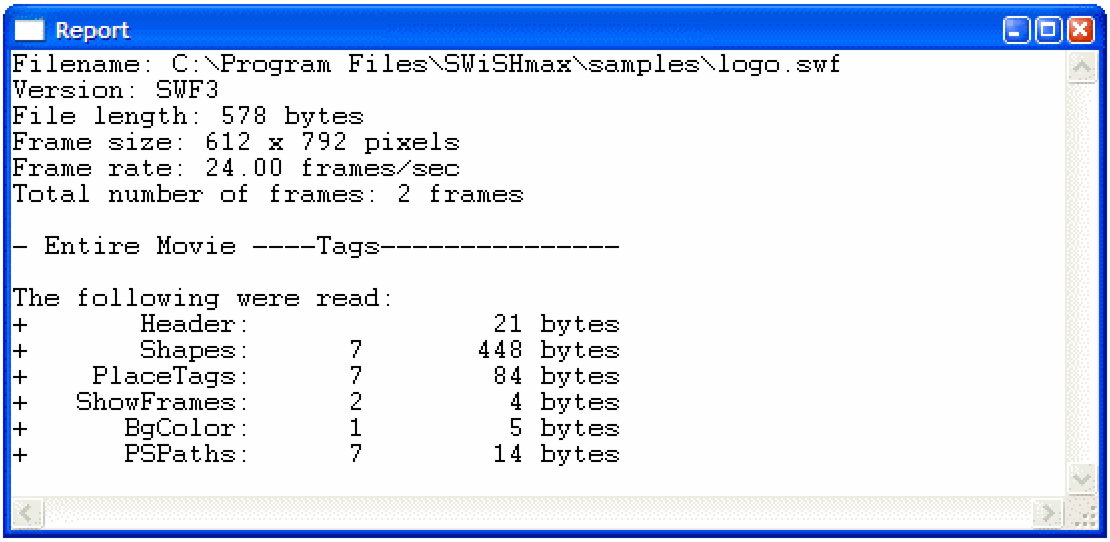
Note: You must have at least one browser associated with .htm files

Note: The Flash Player version depends on the SWF version chosen for export. e.g. if SWF8 then Flash

Player projector version 8 is used. See the section Export Settings for SWF for export options.

File | Test | Report

Displays the details of the Movie to be exported.



```
Filename: C:\Program Files\SWiSHmax\samples\logo.swf
Version: SWF3
File length: 578 bytes
Frame size: 612 x 792 pixels
Frame rate: 24.00 frames/sec
Total number of frames: 2 frames

- Entire Movie ----Tags---------------

The following were read:
+       Header:                21 bytes
+       Shapes:        7      448 bytes
+    PlaceTags:        7       84 bytes
+   ShowFrames:        2        4 bytes
+      BgColor:        1        5 bytes
+      PSPaths:        7       14 bytes
```

6.2.1.16 Export

File | Export | SWF... (Ctrl+E)

Exports the current SWiSH Max Movie to a .swf file. You can import the .swf file into Flash by choosing File | Import and selecting Files of Type: Flash Player (*.swf, *.spl) on the 'Flash Import' dialog box. See the section Export Settings for SWF for export options.

File | Export | HTML + SWF... (Ctrl+P)

Creates an .htm and .swf file for uploading to your web site. Both the .htm and .swf files will have the name you enter on the 'Publish SWiSH Max Movie' dialog box. See the section Export Settings for HTML for export options.

Note: Once these files have been created, simply upload to your web server (using FTP or similar). That's it. No other files are necessary to play your SWiSH Max Movie over the internet.

FTP Upload

Note that files exported for web use will be queued for upload by the integrated FTP upload tool, SWiSH FTP. See Tools | FTP Connection... for more details.

File | Export | EXE (projector)... (Ctrl+Shift+P)

Exports the current SWiSH Max Movie to a Flash Player projector executable. This allows you to show your Movie without any external program or browser.   See the section Export Settings for SWF for export options.

Note: The Flash Player version depends on the SWF version chosen for export. e.g. if SWF8 then Flash Player projector version 8 is used.

File | Export | AVI Movie... (Ctrl+M)

Exports SWiSH Max Movies in the .avi video format. Actions (except sounds), buttons and other interactive components are excluded. See the section Export Settings for AVI for export options.

Note: During the export process, the 'SWiSH Max main application' window will be minimized. The export process is done in two stages: capturing audio and recording data. During the capturing of audio:

- make sure your computer is not running a heavy process, as this will affect the quality of the exported audio
- **Do not** press the 'Cancel' button and then 'Continue' otherwise the exported audio might be out of sync.

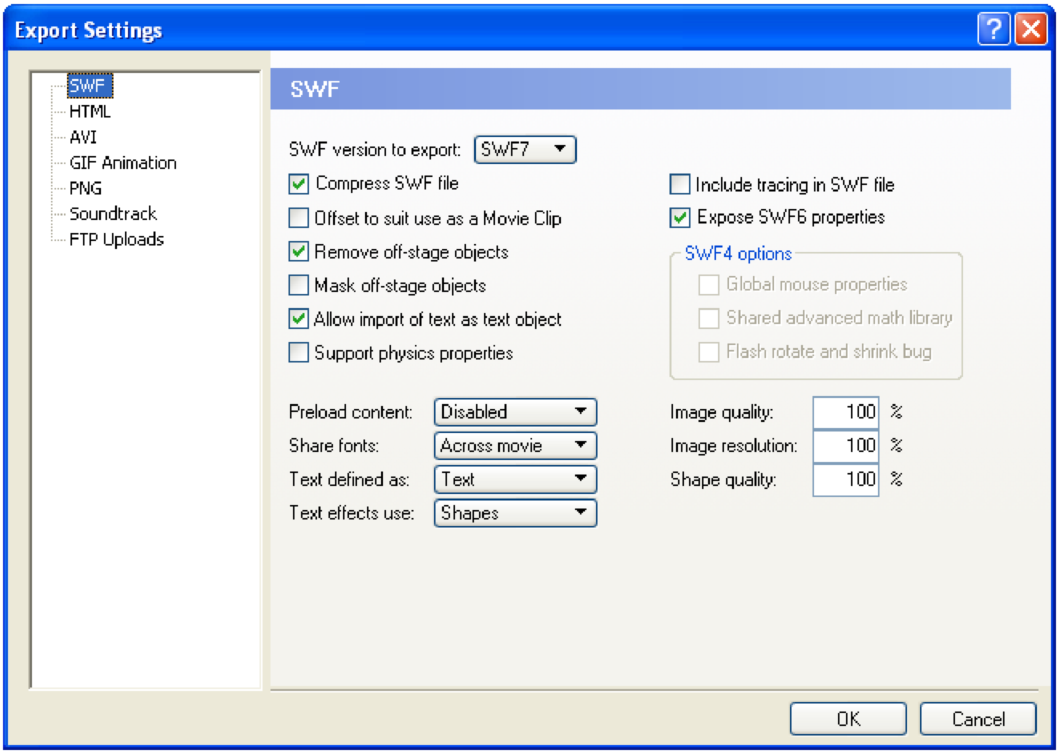
File | Export | GIF Animation... (Ctrl+Shift+F)
Exports SWiSH Max Movies as a GIF animation. See the section Export Settings for GIF Animation for export options.

File | Export | PNG Images... (Ctrl+Shift+N)
Exports SWiSH Max Movies as a sequence of PNG images. See the section Export Settings for PNG for export options.

6.2.1.17 Export Settings

File | Export Settings... (Ctrl+Shift+F12)

Opens the 'Export Settings' dialog box for settings the various export options.



SWF

SWF version to export: SWF7

☑ Compress SWF file
□ Offset to suit use as a Movie Clip
☑ Remove off-stage objects
□ Mask off-stage objects
☑ Allow import of text as text object
□ Support physics properties

□ Include tracing in SWF file
☑ Expose SWF6 properties

SWF4 options
□ Global mouse properties
□ Shared advanced math library
□ Flash rotate and shrink bug

Preload content: Disabled
Share fonts: Across movie
Text defined as: Text
Text effects use: Shapes

Image quality: 100 %
Image resolution: 100 %
Shape quality: 100 %

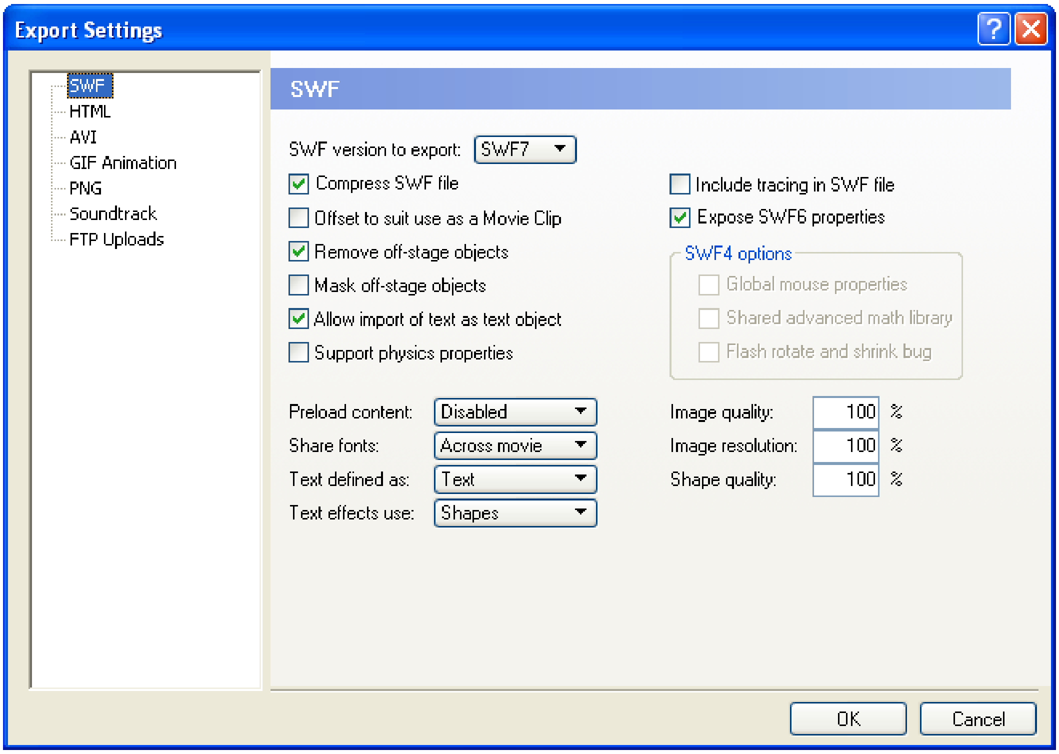
The export options are separated into the following tabs:

- SWF (Flash) export options
- HTML export options
- AVI (Video) export options

- GIF Animation export options
- PNG image export options
- Soundtrack export options
- FTP Upload Export Options

6.2.1.17.1 SWF (Flash) Export Options

The SWF (Flash) option of the 'Export Settings' dialog box offers options to control the generation of the .swf file for the File | Test | SWF in Flash Player, File | Export | SWF... and File | Export | HTML+SWF... commands.



SWF

SWF version to export: SWF7

☑ Compress SWF file
☐ Offset to suit use as a Movie Clip
☑ Remove off-stage objects
☐ Mask off-stage objects
☑ Allow import of text as text object
☐ Support physics properties

☐ Include tracing in SWF file
☑ Expose SWF6 properties

SWF4 options
☐ Global mouse properties
☐ Shared advanced math library
☐ Flash rotate and shrink bug

Preload content: Disabled
Share fonts: Across movie
Text defined as: Text
Text effects use: Shapes

Image quality: 100 %
Image resolution: 100 %
Shape quality: 100 %

SWF version to export
Specifies the version of SWF file to export. Higher versions create more efficient and accurate SWF files due to additional low-level scripting support and options such as compressed SWF files. Lower versions are sometimes required for some platforms (like PocketPC's), or to reach a wider audience (eg more people have players that can play SWF6 files than those that can play SWF8 files).

Compress SWF Movie
This option is only available if 'SWF version to export' is set to versions higher than 'SWF6'. Check this option to generate a compressed Flash Movie to reduce the file size and download time.

Offset to suit use as a Movie Clip
Check this option to export the SWF to a Flash Movie clip, otherwise the animation will be shifted down and to the right when imported into in a Movie clip. The default setting is to have this option turned off.

Remove Off-Stage Objects
Check this option to not animate objects when they are completely outside the defined Movie area (or stage). Checking this option may result in a smaller .swf file size for some Effects because the SWF file will not contain any instructions for animating objects that are outside the stage area.

When importing the .swf file into Flash (or back into SWiSH Max) or if areas outside the stage will be visible, uncheck this option. Otherwise letters will appear to 'stick' when they reach the edge of the 'Movie' window. However, it is possible to use the "Mask Off-Stage Rectangle" option to ensure that off-stage objects are not visible anyway, in which case leave the "Remove Off-Stage Objects" option checked. The default setting is to have this option turned on.

Mask Off-Stage Rectangle
Check this option to mask objects if they are completely outside the defined Movie area (or Stage). Checking this option will add a few bytes to the file size of the SWF and may result in a slight decrease in performance. However, if areas outside the stage are going to be visible, then this will neatly clip objects that move offstage so unexpected objects are not seen. Consider combining this option with the "Remove Off-Stage Objects" option. The default setting is to have this option turned on.

Note: There is a bug in the import option in Flash that will result in errors in SWF publishing from Flash when importing a SWF file from SWiSH Max with the "Mask Off-Stage Rectangle" option turned on. When importing a SWF into Flash leave this option **off**. This does **not** apply when using Load Movie to load a SWF into another SWF.

Allow import of text as text object
Check this option when importing the resultant .swf file into Flash (or back into SWiSH Max) and when it is necessary to edit any text. If this option is unchecked, SWiSH Max will remove the information required to edit text. Unchecking this option may result in smaller and more secure .swf files. The default setting is to have this option turned on.

Note: There is a bug in Flash where it will crash when trying to import a SWF files with this information removed. This acts as some protection for the SWF file.

Support Physics Properties
Adds the shared script necessary for physics properties for objects in the Movie. Each Object using physics properties must ALSO have the 'Uses physics properties' option turned on in the selected Object's 'Export' Panel. If physics properties are not used, for any objects in the movie, then do not check this option.

Preload content
This option determines where object definitions are written to the .swf file. Grouping definitions can avoid delays or jerkiness while the Movie is playing. The Preload content options are as follows:

- **Disabled**: definitions are written out at the Frame where the object is first placed. If the definition is large because these is a large object or block of text, or if the connection speed is slow, this could introduce a short delay or jerk in the playback
- **Before Scene**: definitions are grouped together at the start of the Scene. This may mean a short delay before the Scene starts playing, but such delays are generally less noticeable than those that occur while the Scene is playing
- **Before Movie**: definitions are grouped together before the first Frame of the Movie. This can mean a delay before the Movie begins to play
- **At Preload frame**: Add a Preload Content Action as a Scene Event, this acts as a marker for where the definitions appear. Any objects before the first Preload Content option are treated as though Preload content was disabled.

Share Fonts
Font definitions in a .swf file define the shapes for characters. To save space, only the actual characters used appear in font definitions. This setting specifies where font definitions are written out and what they contain. Unless it is particularly necessary to override the default setting for the Scene, leave the setting as 'Scene Default'. The possible values for this setting are:

- **Disabled**: a separate font definition is written out for each Text Object for the characters used in that text object only
- **Across Scene**: font definitions are written out at the start of the Scene that include all the characters

used in the Scene. This saves space by combining multiple fonts

- **Across Movie**: font definitions are written out at the start of the first Scene found that has any Text Objects with Share fonts set to Across Movie. These definitions include all the characters used in the Movie.

Note: In all cases, if a font has already been defined that includes all the necessary characters, the existing font definition is used. In general the 'Across Movie' setting will result in the smallest .swf size. However, it may mean that there is a delay at the start of the Movie when fonts are defined. This can be a problem when using a preloader. In this case, it is common to use Across Scene as the default for the preloader Scene, and Across Movie as the default for subsequent Scenes.

Text Defined As

This option sets the export of characters of the Text Object as individual shapes or fonts and text. This should make no difference visually. However, if the .swf file is imported and text is defined as 'shape' the Text Object can no longer be edited. Set this option if text does not appear in the correct font when importing it into another application, such as Flash. The possible values for this setting are:

- **Text**: text is not exported as shapes, but rather is exported as font definitions and text records. If Complex Effects are used, then SWiSH Max will automatically define the text as individual objects for each character, so the characters can be animated individually
- **Shape**: text is converted into shapes. SWiSH Max generates separate shape definitions for each combination of font size or color.

Note: If it is necessary to define text as shape, the most efficient setting will depend on the variety of letters, fonts, font sizes and colors in use. Experiment with the settings for each individual Movie to get the smallest possible .swf size.

Text Effects Use

When Complex Effects are applied to Text Objects, SWiSH Max needs to break the text into individual objects for each character within the .swf file itself. This option specifies what sort of objects SWiSH Max will break the text into (i.e. **Letters** or **Shapes**). Changing this setting can affect the overall .swf file size.

Include tracing in SWF file

Include tracing in exported SWF files to aid debugging when loaded into another movie.

Expose SWF6 Properties

When text or button objects have the target option checked, they are wrapped in Movie Clips. This prevents normal access to the native text and button properties and methods. Setting this option allows use of SWF6 properties within Text or Button Objects when the object has the 'target' option checked.

Notes:

- Using this option adds an overhead to your movie as additional scripting is inserted to access the native object properties.
- SWF6 properties can be accessed directly if the object target checkbox is not set.
- If the target checkbox is set, SWF6 properties can still be accessed without this option via the _text and _button properties.

SWF4 options group:

These options are only available if exporting to SWF4

Global mouse properties

Check this option to track the current x and y coordinates of the cursor in pixels through the _xmouseglobal and _ymouseglobal properties in script.

Shared advanced math library
Only available when exporting SWF4 movies. If selected SWiSH Max generates an extra Movie Clip with all the math methods in it. The extra Movie Clip adds approximately 9K to the SWF file produced, but means that calls to math methods are shorter. If a lot of math methods are used this will produce a smaller SWF file. If loading this SWF movie into a Movie Clip in another SWF Movie ensure the other SWF Movie ALSO has this option checked. If the SWF is loaded into a Movie Clip in a non-SWiSH Max Movie, then the math functions will not work.

Flash rotate and shrink bug
Only available when exporting SWF4 movies. There is a bug in all versions of Flash player when playing SWF4 movies, where every time the _rotation property is changed using script, the objects will shrink slightly. Selecting this option will make SWiSH Max generate additional code that works around this Flash Player bug. If script does not rotate objects, then do not check this option.

Image Quality
Only applicable to JPEG compressed images that specify image quality. Specifies image quality up to 100%. 0% will reduce the quality of the affected images to 0% resulting in a small file size but very poor quality. 100% will not change the quality for the images, and will use the settings for each individual image. This sets the default value for the Image Quality setting for the Movie. However, individual Scenes or Objects can override these defaults.

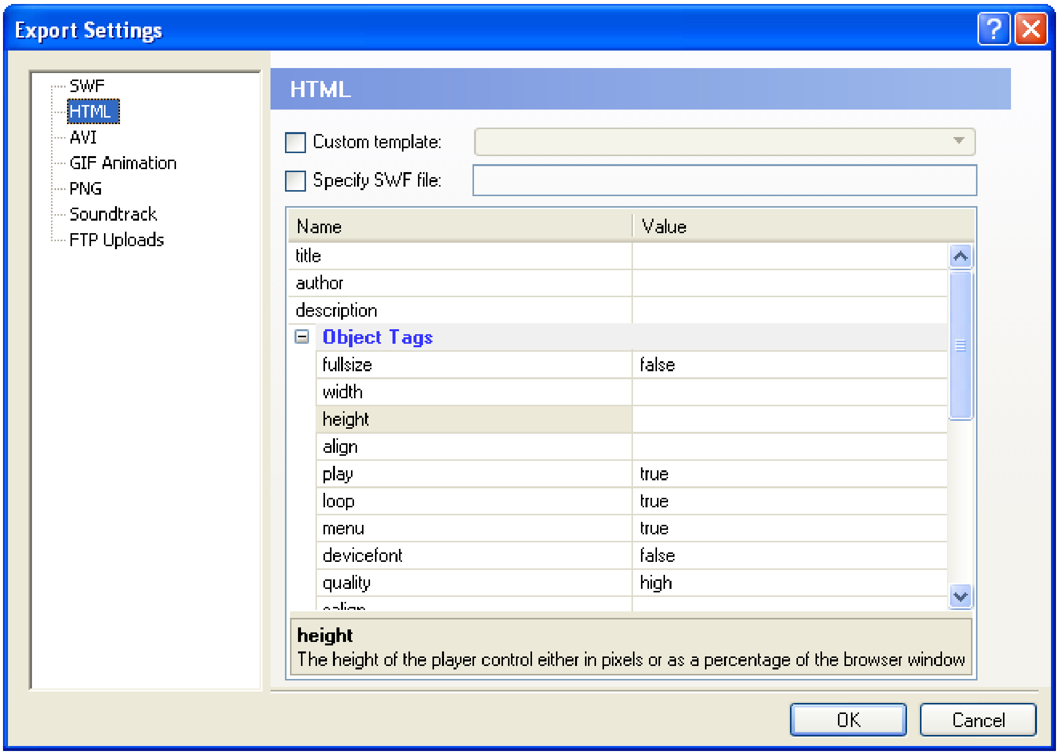
Note: Each image can be set to ignore these image quality settings, see the 'Ignore export quality overrides' option in the Image Properties dialog for more information.

Image Resolution
Specifies an images resolution up to 100%. Reducing an images resolution will reduces the images file size but add a mosaic effect to it. For an 100x100 pixel image, resetting the resolution to 50% will make it export as a 50x50 pixel image scaled by 200% to maintain the original image size. This sets the default value for the Image Resolution setting for the Movie. However, individual Scenes or Objects can override these defaults.

Shape Quality
Specifies the shape quality up to 100%. Reducing a shapes quality will shift the vertices and control points in a shape to allow for smaller file size. Reducing quality will reduce the detail in the shape and change curvature of curved edges. Setting a shape quality of 0% will reduce the shape quality of all affected shapes to 0%, so they will become a single simple shape. Setting a shape quality of 100% would not change the quality for the shapes, and would use the settings for each individual shape. This sets the default value for the Shape Quality setting for the Movie. However, individual Scenes or Objects can override these defaults.

6.2.1.17.2 HTML Tags Export Options

The HTML option of the 'Export Settings' dialog box defines the .htm file exported in File | Export | HTML+SWF... and File | Test | HTML + SWF in Browser.



☐ Custom template:

☐ Specify SWF file:

Name	Value
title	
author	
description	
⊟ Object Tags	
fullsize	false
width	
height	
align	
play	true
loop	true
menu	true
devicefont	false
quality	high
salign	

height
The height of the player control either in pixels or as a percentage of the browser window

Note: See Tools | Preferences | Templates for the default templates used for File | Export | HTML + SWF ... and File | Test | HTML + SWF in Browser.

Custom template

When the 'Custom template' checkbox is ***unchecked*** (the default), then SWiSH Max uses the template defined in Tools | Preferences | Templates for the default templates used for File | Export | HTML + SWF ... and File | Test | HTML + SWF in Browser.

When the 'Custom template' checkbox is ***checked*** the HTML is defined by the custom template selected from the list.

Currently the available options are:

- **default** - uses the <embed> html tag to access the flash player.
- **no-embed-tag** - uses the <object> html tag to access the flash player.
- **xhtml** - uses xhtml to access the flash player object.

The template html file can have one or more tags, or variables, embedded inside it. The tag names can be user defined however SWiSH Max has a number of reserved tags which insert values from the Movie setup directly and these are not shown in the Variable list.

Specify SWF file

Normally when exporting to HTML the SWF file is named the same as the HTML file. If 'Specify SWF file' is ***checked***, the SWF file name can be given a unique filename. For example 'index.html' can be exported with a different SWF filename. In all cases the SWF is saved to the same location target location as the HTML.

Name / Value

SWiSH Max uses template files to tell it how to produce the HTML files exported. These template files are basically HTML files with some additional information. Because they are not just plain HTML files, SWiSH Max uses the file extension ".swhtml" for these file.

During the export process when the output HTML file is created, SWiSH Max reads and parses the template file, looks for the tag name and replaces the tag with the matching values in the export template. A number of reserved tags also replace information from the Movie setup, see below for a list of reserved tags.

Object tags

Object tags define attributes of the HTML and playing back the SWF movie.

- **fullsize**: Boolean - Play SWF Movie full size (100%)
- **width**: String - the width of the Movie, or 100% for full size
- **height**: String - the height of the Movie, or 100% for full size
- **align**: String [default,l,r,t,b] - aligns html in playback
- **play**: Boolean - true pauses Movie at start
- **loop**: Boolean - true loops Movie
- **menu**: Boolean - true displays full menu instead of reduced menu
- **devicefont**: Boolean - true renders static text as device font
- **quality**:[low, autolow, autohigh, high, best] - sets Playback quality
- **salign**: String [default,l,r,t,b,tl,tr,bl,br - Determines how the Movie is placed within the 'Movie' window.
- **scale**: String [showall, no border, exactfit] - sets scale options
- **wmode**: String [window, opaque, transparent] - sets window mode options
- **swftitle**: String - sets title of SWF Movie
- **standby**: String - message to display while loading SWF Movie
- **base**: String - base directory or URL to resolve relative file paths in the SWF Movie
- **flashvars** (FP6+): String - Variables to pass to the SWF Movie
- **seamlesstabbing** (FP7+): Boolean - Allows tabbing between movie and web page
- **tabindex**: Integer - Position of the SWF movie in the browser tab order
- **allowscriptaccess**: String [always, never, samedomain] - controls outbound scripting from within the SWF movie
- **allownetworking** (FP9+): String [all, internal, none] - restricts browser communication

Custom Tags

Additional information can be placed in special template tags. These tags always start with the pair of characters "[[" and end with the pair of characters "]]".

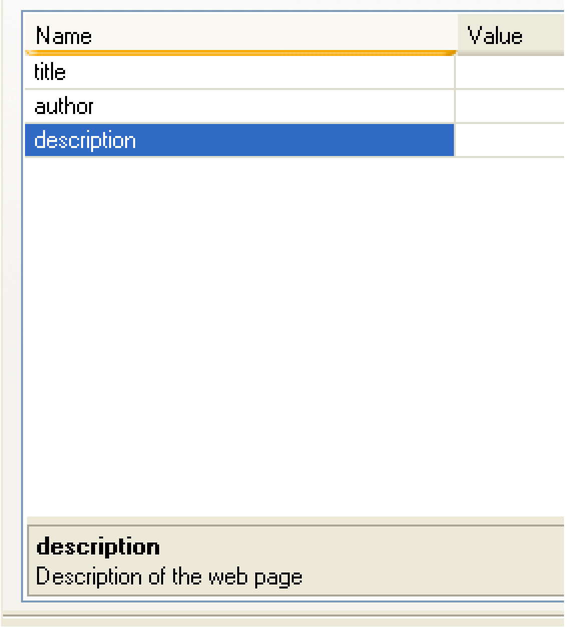
There are two template tags types:

- **parameter**: The simplest template tag is an HTML parameter name within the [[]]. eg. "[[myparameter]]". When exported, the value of the named parameter is converted to a string and replaces the whole tag (including the [[]] delimiters).
- **expression**: An expression can also be used in the template tag by writing it between parentheses. eg. "[[(firstname+' '+lastname)]]". No spaces are allowed between the '[[' and the opening '('. When exported, the expression is evaluated and converted to a string, and the result replaces the whole tag (including the [[]] delimiters).

Special template tags

- **[[#deleteblanklines]]**: Usually blank lines (that is, empty lines, or lines with only spaces or tabs) should be removed for an html template. The format for this tag is: "[[#deleteblanklines]]". If this is present in the file, then blank lines are removed.
- **[[<parameters>...</parameters>]]** : This determines how they appear in the 'Export Settings | HTML Template' dialog. The format for this tag is: "[[<parameters>...</parameters>]]". Where "..." is where the parameter definitions are placed. e.g. from default.swhtml

```
...
[[<parameters>
    <parameter name="title" prompt="Web page title" />1
    <parameter name="author" prompt="Author of the web page" />
    <parameter name="description" prompt="Description of the web page" />
</parameters>]]
...
```



Name	Value
title	
author	
description	

description
Description of the web page

Reserved Tags
The following lists the reserved variables used:

- **testinbrowser**: Boolean - true if the html is for testing in browser, false for publishing.
- **generator**: String - a string identifying SWiSH Max as the program that created the HTML
- **hasanchor**: Boolean - true if the movie has any named anchors, otherwise false
- **keywords**: String - a list of words used in the Movie
- **keywordscommenttags**: String - a set of comment variables for the list of words used in the Movie
- **urlscommenttags**: String - a set of comment variables for the list of URLs used in the Movie
- **defauthor**: String - user name
- **fscommandtag**: String - skeleton FSCommand handling script tag, FSCommands are used.
- **anchortags**: String - tags required for handling named anchors
- **adtag**: String - a comment variable saying that the html file was produced by SWiSH Max
- **bgcolor**: String - background color in #RRGGBB format
- **version**: String - flash player version string, eg 9.0.28.0
- **id**: String - the name of the SWF Movie
- **movie**: String - the file name of the SWF Movie

6.2.1.17.2.1 fullsize

fullsize (size=100%)
Scales the Movie scale to occupy the entire html page. It sets the width and height values of the Object and Embed tags to 100%. When turned off (the default setting), the width and height are set from the Movie size (or stage).

6.2.1.17.2.2 width

width
The width of the player control either in pixels or as a percentage of the browser window.

6.2.1.17.2.3 height

height
The height of the player control either in pixels or as a percentage of the browser window.

6.2.1.17.2.4 align

align (HTML alignment)
Determines how the Flash Player 'Movie' window is positioned within the 'Browser' window. It sets the Align value of the Object and Embed tags. The 'align' options are as follows:

- **default**: centers the Movie along within the 'Browser' window and crops the top, bottom, left, and right sides as necessary
- **l (left)**: aligns the Movie along the left edge of the 'Browser' window and crops the top, bottom, and right sides as necessary
- **r (right)**: aligns the Movie along the right edge of the 'Browser' window and crops the top, bottom, and left sides as necessary
- **t (top)**: aligns the Movie along the top edge of the 'Browser' window and crops the bottom, left, and right sides as necessary
- **b (bottom)**: aligns the Movie along the bottom edge of the 'Browser' window and crops the top, left, and right sides as necessary.

6.2.1.17.2.5 play

play (paused at start)
If checked, the Movie is paused at the start.

6.2.1.17.2.6 loop

Loop
Repeats the Movie from the beginning when it reaches the last Frame. It sets the Loop value of the Object and Embed tags to True. De-selecting this option stops the Movie when it reaches the last Frame. Loop is checked by default.

6.2.1.17.2.7 menu

Display Menu
This option makes a shortcut Menu available to users when they right-click (Windows) or Command-click (Macintosh) on the Movie. It sets the Menu value of the Object and Embed tags to True. Deselect the Show Menu option and the About Flash is the only option available in the shortcut Menu. The Display Menu option is enabled by default.

6.2.1.17.2.8 devicefont

Device Font
Automatically use Device Fonts for all Static Text if the required fonts are available. Renders static text using device fonts if possible, even if device fonts option is not selected.

6.2.1.17.2.9 quality

Quality
Sets the level of anti-aliasing performed, and hence the smoothness of objects when playing a Movie. It sets the Quality value of the Object and Embed tags. Because anti-aliasing requires a faster processor to smooth each Frame of the Movie before it is rendered on the viewer's screen, the Quality parameter assigns priorities to appearance and playback speed. The priority options are as follows:

- **Low**: prioritizes playback speed over appearance. With this setting, anti-aliasing is never used
- **Auto low**: emphasizes speed at first, but improves appearance whenever possible. Playback begins with anti-aliasing turned off. If the Flash player detects that the processor can handle it, anti-aliasing is turned on
- **Auto high**: emphasizes playback speed and appearance equally at first, but sacrifices appearance for the sake of playback speed if necessary. Playback begins with anti-aliasing turned on. If the actual Frame Rate drops below the specified Frame Rate, anti-aliasing is turned off to improve playback speed. Use this setting to emulate the 'View > Anti-alias' setting in Flash
- **High**: prioritizes appearance over playback speed. With this setting, anti-aliasing is always used. If the Movie does not contain animation, bitmaps are smoothed; if there is animation, bitmaps are not smoothed. High is the default setting
- **Best**: provides the best display quality and does not consider playback speed. All output is anti-aliased and all bitmaps are smoothed.

6.2.1.17.2.10 salign

s[wf]align (flash alignment)
Determines how the Movie is placed within the 'Movie' window and, if it must be cropped to fit that window, how it is cropped. It sets the Salign value of the Object and Embed tags.

The salign options are as follows:

- **" " (default)**: displays the Movie centered in the 'Movie' window. All edges might be cropped if the 'Movie' window is smaller than the Movie
- **l (left)**: aligns the Movie along the left edge of the 'Movie' window and crops the top, bottom, and right sides as necessary
- **r (right)**: aligns the Movie along the right edge of the 'Movie' window and crops the top, bottom, and left sides as necessary
- **t (top)**: aligns the Movie along the top edge of the 'Movie' window and crops the bottom, left, and right sides as necessary
- **b (bottom)**: aligns the Movie along the bottom edge of the 'Movie' window and crops the top, left, and right sides as necessary
- **tl (top left)**: aligns the Movie along the top and left edges of the 'Movie' window and crops the bottom and right sides as necessary
- **tr (top Right**: aligns the Movie along the top and right edges of the 'Movie' window and crops the top and left sides as necessary
- **bl (bottom left)**: aligns the Movie along the bottom and left edges of the 'Movie' window and crops the top and right sides as necessary
- **br (bottom right)**: aligns the Movie along the bottom and right edges of the 'Movie' window and crops the top and left sides as necessary.

6.2.1.17.2.11 scale

Scale
Defines how the Movie is placed within the boundaries of the 'Browser' window. It sets the Scale value of the Object and Embed tags.
These settings are used if there is a width and height that are different from the Movie's original size (for example, if the Size 100% option is selected). The available settings are as follows:

- **showall**: makes the entire Movie visible in the specified area. The original aspect ratio of the Movie is maintained, and no distortion occurs. Borders may appear on two sides of the movie. This is the default setting
- **noborder**: scales the Movie to fill the specified area. The original aspect ratio of the Movie is maintained, and no distortion occurs. Portions of the Movie may be cropped
- **exactfit**: makes the entire Movie visible in the specified area. However, no attempt is made to preserve the original aspect ratio, and distortion may occur.

6.2.1.17.2.12 wmode

wmode (Window Mode)
Sets the transparent Movie, absolute positioning, and layering capabilities available in Internet Explorer 4.0 (or later). This option works only in the Windows version of Internet Explorer with the Flash Active X control. It sets the WMode value of the Object and Embed tags. The settings are as follows:

- **window**: plays the Movie in its own rectangular window on a web page. This setting normally provides the fastest animation performance and is the default setting
- **opaque**: use this setting to move elements behind Flash movies (for example, with dynamic HTML) and not have them show through
- **transparent**: use this setting to make the background of the HTML page on which the Movie is embedded show through all the transparent portions of the Movie. Animation performance might be slower when this setting is used.

6.2.1.17.2.13 swftitle

swftitle
Sets the title of the SWF movie.

6.2.1.17.2.14 standby

standby
Sets the message to display while loading the SWF movie.

6.2.1.17.2.15 base

Base
Specifies the Base directory or URL to resolve any relative path names in the Movie. It sets the Base value of the Object and Embed tags.

6.2.1.17.2.16 flashvars

flashvars (FP6+)
Variables to pass to the SWF movie.

6.2.1.17.2.17 seamlesstabbing

seamlesstabbing (FP7+)
Allows tabbing between movie and web pages.

6.2.1.17.2.18 tabindex

tabindex
Sets the position of the SWF movie in the browser tab order.

6.2.1.17.2.19 allowscriptaccess

allowscriptaccess (FP6+)
This option sets the 'allowScriptAccess' parameter in the HTML code. This option controls the ability for ' FSCommand' and 'getURL' actions to perform scripting from within a .SWF file. This parameter will only be accepted for browsers using Flash Player v.6.0.40.0 or later and will be ignored by earlier versions.
The 'allowScriptAccess' parameter has three options:

- **never**: This will disallow any outbound scripting from the .SWF File
- **always**: This will allow all outbound scripting from the .SWF file.
- **samedomain**: This will allow all outbound scripting from the .SWF file when located on the same domain.

6.2.1.17.2.20 allownetworking

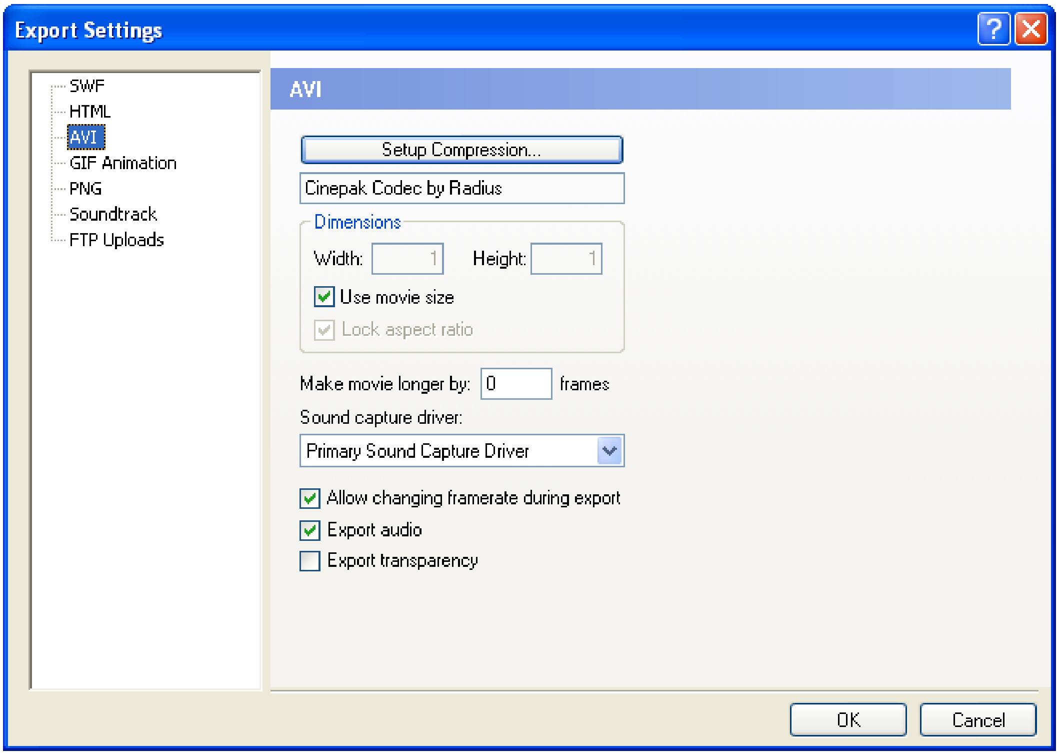
allownetworking (FP9+)
This HTML parameter governs a number of ActionScript APIs. It has the following possible values:

- **all**: the default. No networking restrictions; player behaves normally.
- **internal**: SWF files may not call browser navigation browser interaction APIs, but may call any other APIs.
- **none**: SWF files may not call any networking any SWF-to-SWF communication APIs.

6.2.1.17.3 AVI (Video) Export Options

The AVI option of the 'Export Settings' dialog box allows you to control the generation of the .avi file for the File | Export | AVI movie ... command.

Note: The AVI export automatically rounds up the width to the closest number divisible by four. This is required for most types of compression. All action script in the scene will be disabled.



Export Settings

- SWF
- HTML
- AVI
- GIF Animation
- PNG
- Soundtrack
- FTP Uploads

AVI

Setup Compression...

Cinepak Codec by Radius

Dimensions

Width: 1 Height: 1

☑ Use movie size

☑ Lock aspect ratio

Make movie longer by: 0 frames

Sound capture driver: Primary Sound Capture Driver

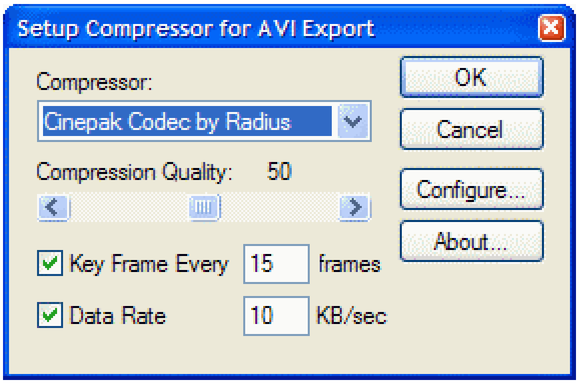
☑ Allow changing framerate during export

☑ Export audio

□ Export transparency

Setup Compression

Click this button to open up the 'Setup Compressor for AVI Export' dialog box to choose the available compression types and their attributes.



Setup Compressor for AVI Export

Compressor: Cinepak Codec by Radius

Compression Quality: 50

☑ Key Frame Every 15 frames

☑ Data Rate 10 KB/sec

Compressor

Choose your desired compression type here. Different compression types will produce different .avi file quality and size. When choosing compression type, make sure the people who are going to view the .avi file also have that compression type available. The most common one is Cinepak Codec by Radius, which is selected by default. If this is not available, the default changes to Uncompressed.

Some compression types are not available for a certain color depth. If the desired compression type can not be found, try changing the screen color depth before clicking on this button.

Compression Quality

Set this value to adjust the image quality of your .avi file. The better the quality, the bigger the file size. The default setting is 50%. Some compression types do not let you adjust Compression Quality.

Key Frame
This sets how many frames apart the AVI exporter needs to set Keyframes. Smaller gaps between Keyframes will produce better, but bigger .avi files. Some compression types do not let you adjust Key Frame. The default setting is to have this option turned on with a Keyframe inserted every 15 Frames. Uncheck this to use a compression-type-specific default value for Data Rate.

Data Rate
This sets the Data Rate of your .avi file. Smaller values will produce smaller .avi files, but worse quality than higher values. Some compression types do not let you adjust Data Rate. The default setting is to have this option turned on with a Data Rate figure of 10 KB/sec. Uncheck this option to use a compression-type-specific default value for Data Rate.

Configure
Click this button to adjust a compression-type-specific setting. Different compression types will have different settings.

Dimensions:

Width
Sets the width of the exported .avi file. This value will be ignored if 'use movie size' is chosen.

Height
Sets the height of the exported .avi file. This value will be ignored if 'use movie size' is chosen. Height will automatically be calculated from Width if 'Lock Aspect Ratio' is chosen.

Lock Aspect Ratio
Automatically calculates the width to maintain the aspect ratio of the exported .avi file. The default setting is to have this option turned on.

Use Movie size
Check this option to use the Movie size for the exported .avi file. The default setting is to have this option turned on.

Make movie longer by [___] frames
Enter number of frames to add at the end of the movie. The default setting is 0. If this number is 0, the exported AVI will be as long as the total frames within the main timeline of all scenes.

Sound capture driver
Set the capture source for the sound.

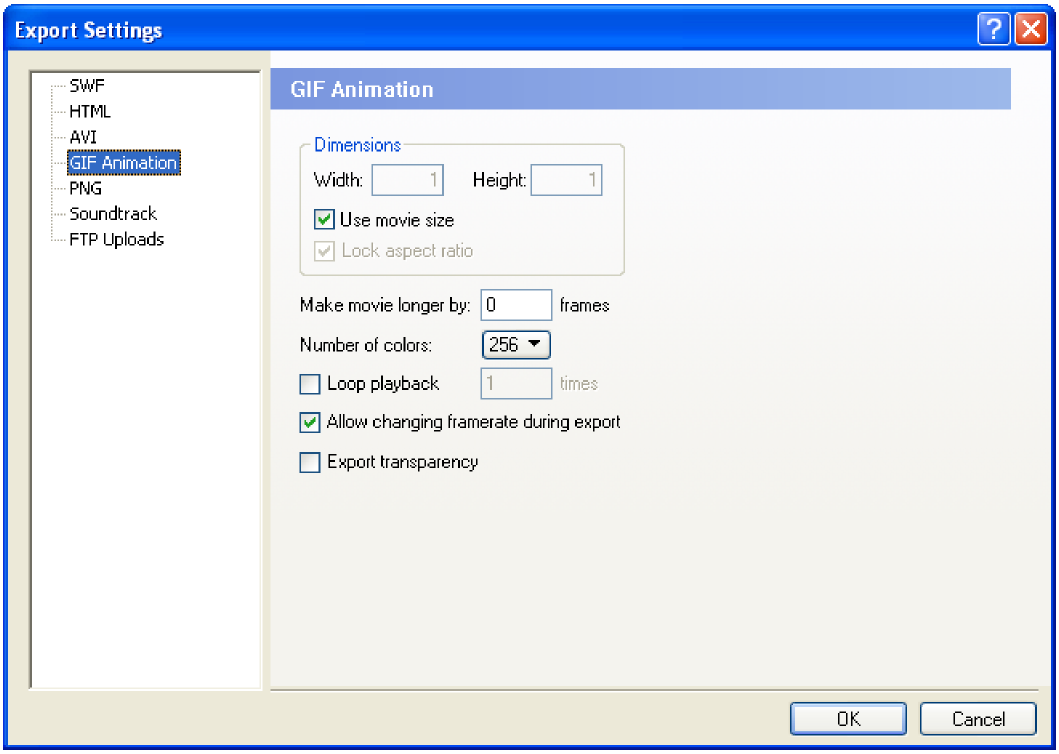
Allow changing frame rate during export
The movie Frames per second will be increased to as high as the Personal Computer can cope to speed up the export process.

Export Audio
This option disables audio in the exported AVI.

Export transparency
This option enable transparency in the file. The resulting file will be a 32 bit RGBA with alpha channel. Whatever is transparent in the movie will also be transparent in the alpha channel.

6.2.1.17.4 GIF Animation Export Options

The GIF Animation option of the 'Export Settings' dialog box allows you to control the generation of the .gif file for the File | Export | GIF animation... command.



GIF Animation

Dimensions

Width: 1 Height: 1

☑ Use movie size

☑ Lock aspect ratio

Make movie longer by: 0 frames

Number of colors: 256

☐ Loop playback 1 times

☑ Allow changing framerate during export

☐ Export transparency

Dimensions:

Width
Sets the width of the exported .gif file. This value will be ignored if 'use movie size' is chosen.

Height
Sets the height of the exported .gif file. This value will be ignored if 'use movie size' is chosen. Height will automatically be calculated from Width if 'Lock Aspect Ratio' is chosen.

Lock Aspect Ratio
Automatically calculates the width to maintain the aspect ratio of the exported .gif file. The default setting is to have this option turned on.

Use Movie size
Check this option to use the Movie size for the exported .gif file. The default setting is to have this option turned on.

Make movie longer by [___] frames
Enter number of frames to add at the end of the movie. The default setting is 0. If this number is 0, the exported GIF will be as long as the total frames within the main timeline of all scenes.

Number of colors
Defines the number of colors to be used in the .gif file.

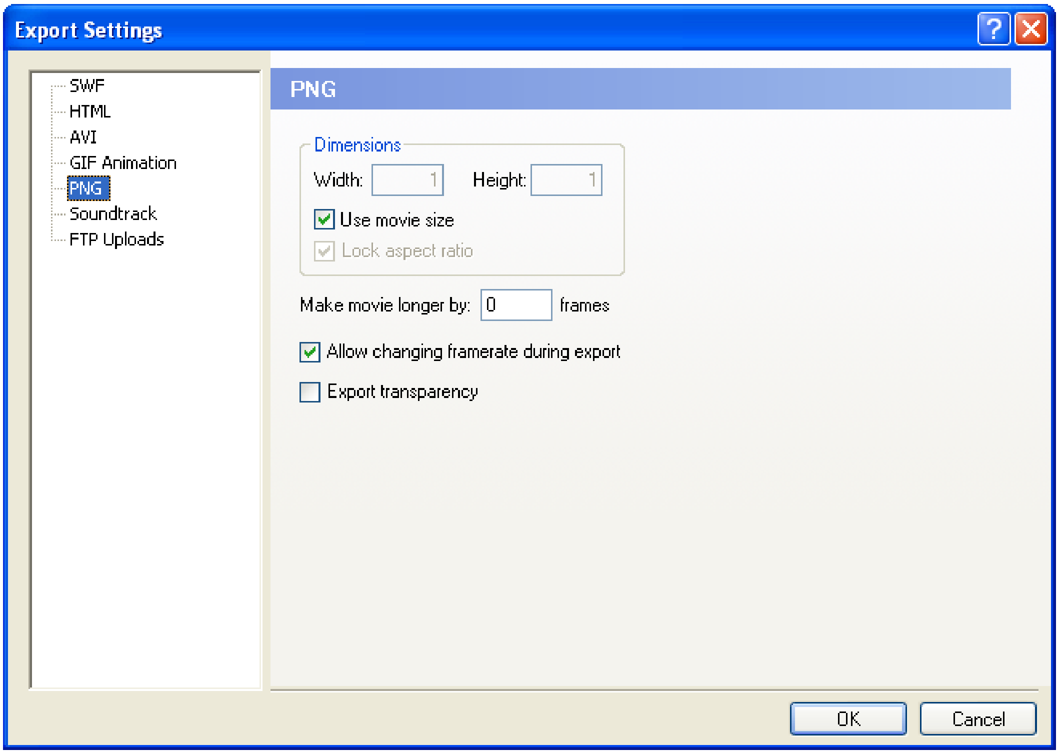
Loop Playback
Sets the exported .gif animation to play continuously.

Allow changing framerate during export
The movie Frames per second will be increased to as high as the Personal Computer can cope to speed up the export process.

Export transparency
This option enable transparency in the file. The resulting file will be a 32 bit RGBA with alpha channel. Whatever is transparent in the movie will also be transparent in the alpha channel.

6.2.1.17.5 PNG Export Options

The PNG option of the 'Export Settings' dialog box allows you to control the generation of the .png file for the File | Export | PNG Images ... command.



Width: 1 Height: 1

☑ Use movie size

☑ Lock aspect ratio

Make movie longer by: 0 frames

☑ Allow changing framerate during export

☐ Export transparency

Dimensions:

Width
Sets the width of the exported .png file. This value will be ignored if 'use movie size' is chosen.

Height
Sets the height of the exported .png file. This value will be ignored if 'use movie size' is chosen. Height will automatically be calculated from Width if 'Lock Aspect Ratio' is chosen.

Lock Aspect Ratio
Automatically calculates the width to maintain the aspect ratio of the exported .png file. The default setting is to have this option turned on.

Use Movie size
Check this option to use the Movie size for the exported .png file. The default setting is to have this option turned on.

Make movie longer by [___] frames
Enter number of frames to add at the end of the movie. The default setting is 0. If this number is 0, the exported AVI will be as long as the total frames within the main timeline of all scenes.

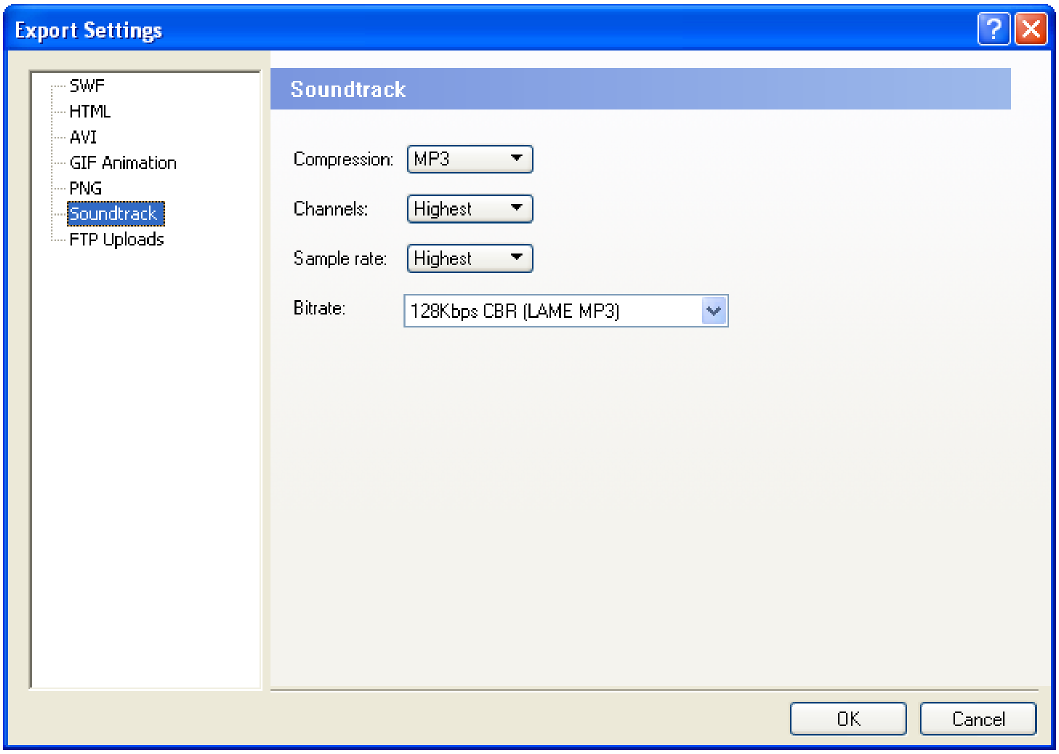
Allow changing framerate during export
The movie Frames per second will be increased to as high as the Personal Computer can cope to speed up the export process.

Export transparency
This option enable transparency in the file. The resulting file will be a 32 bit RGBA with alpha channel. Whatever is transparent in the movie will also be transparent in the alpha channel.

6.2.1.17.6 Soundtrack Export Options

The Soundtrack option of the 'Export Settings' dialog box allows you to control the generation of the sound for the File | Export | GIF animation... command.

For setting options refer to the section on Import Sound and Audio properties.



Soundtrack

Compression: MP3

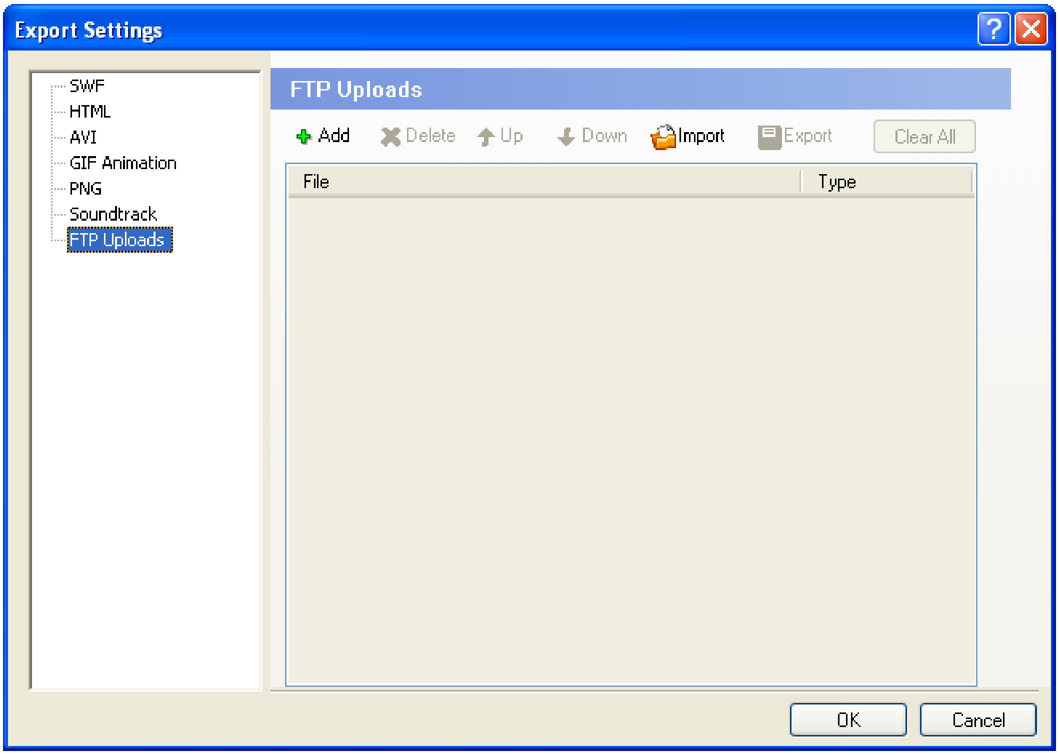
Channels: Highest

Sample rate: Highest

Bitrate: 128Kbps CBR (LAME MP3)

6.2.1.17.7 FTP Uploads Export Options

The FTP Upload option of the 'Export Settings' dialog box sets additional files which can be added to the FTP upload queue.



Add / Delete / Up / Down
The buttons at the top of this dialog allow individual files to be added to, deleted from or moved within the list.

Import / Export
It is possible to import and export the file list as a .csv file via the Import and Export commands. This facility is useful when moving the development environment to another machine.

Tip: When transferring a copy of SWiSH Max to a new or different machine, backup file lists to a .csv file as described above and restore to the new machine when SWiSH Max is installed.

Clear All
Clears all files from the list.

6.2.1.18 Convert Video to FLV

File | Convert Video to FLV
Converts an external video file to Flash Video and saves as an external FLV file. This is a tool to convert video to FLV format for later use with External Media. This does ***NOT*** create an Object within SWiSH Max.

Supported file formats:

- Adobe Flash Video (*.flv)
- AVI files (*.avi)
- Quicktime Video files (*.mov, *.qt)
- MPEG video files (*.mpg)
- Windows Media Video files (*.wmv, *.asf)

Note: Other video import operations include:

- ''File | Import to Stage | Video...'
- 'Insert | Import Video...'
- Content Panel 'Import to Stage' ...Video'

These options offer to convert AND to import the video as either External FLV (creating an External Media Object with controls) or Embedded Video Object.

For a description of the Video import dialog options see Import Video.

The FLV Export options as show in the example below are:

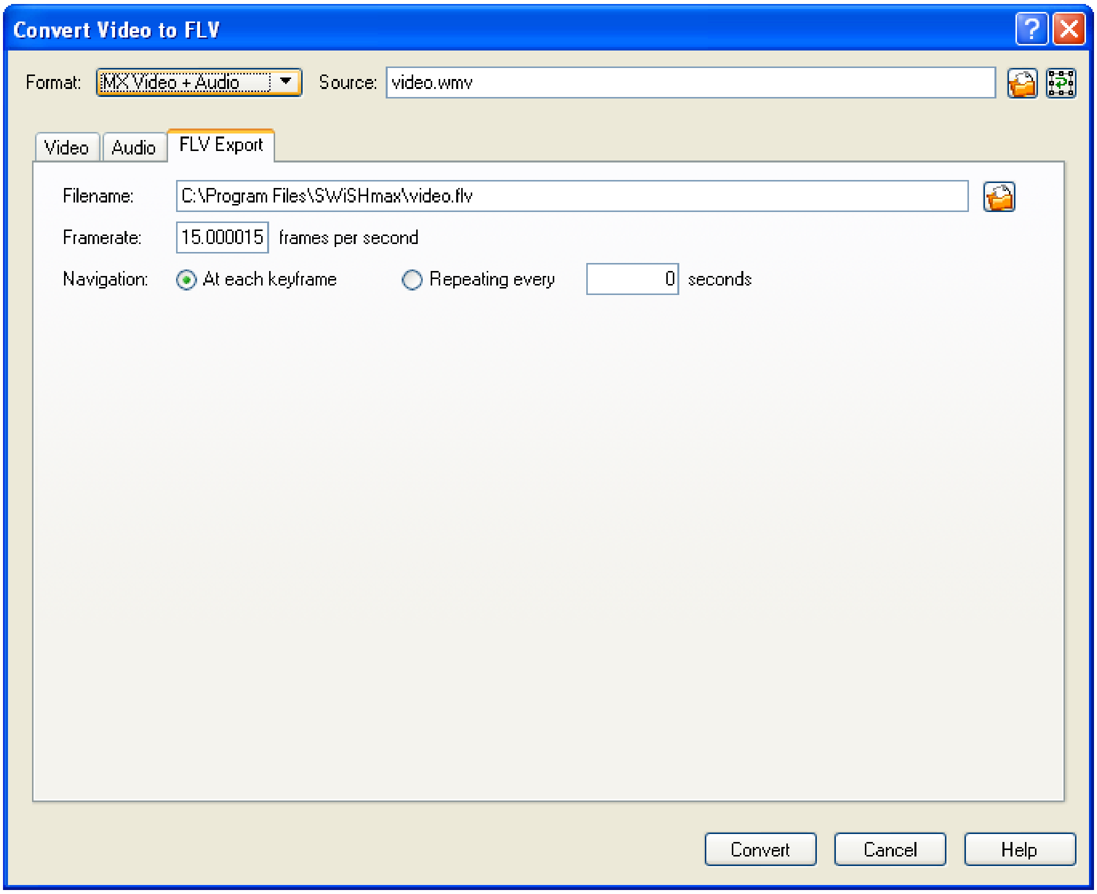
Filename
Target filename for the converted video file.

Export Video
This button opens a dialog to select a target video file path/name.

Frame rate
The Flash video (FLV) Frame rate - frames per second of play. The default value matches the source video frame rate. A lower framerate will create a smaller file but will be less smooth.

Navigation
Video is controlled (via Video components, script etc.) by moving between keyframes. Closer keyframes offers more precise control of the video but creates a larger video file. A keyframe each 1 second is recommended.

- **At each keyframe**: Creates a Flash FLV keyframe at the same position of the source video's keyframes.
- **Repeating every [___] second**: Creates a keyframe periodically throughout the video.



Format: MX Video + Audio Source: video.wmv

Filename: C:\Program Files\SWiSHmax\video.flv

Framerate: 15.000015 frames per second

Navigation: ● At each keyframe ○ Repeating every 0 seconds

6.2.1.19 Recent Files

Below the 'Convert Video to FLV...' Menu item and above the Exit Menu item, you will see a list of the seven most recently used files. Select any of these Menu options to open the file.

6.2.1.20 Exit

File | Exit (Ctrl+Q)
Exits the SWiSH Max application. All 'SWiSH Max' windows will be closed.

6.2.2 Edit Menu

The Edit Menu enables you to make changes to a Movie.

The available options are as follows:

Undo or Redo changes

- Undo
- Redo

Copy, Paste and Delete Scene, Object, Effects, etc.

- Cut
- Copy
- Paste
- Paste in Place
- Delete

Object Selection

- Select All
- Deselect All

Search

- Find...
- Replace...

Show the properties of an Object, Effect, etc.

- Properties

Preferences

- Preferences.

The exact name of the item Cut Scene, Cut Object etc will depend on the context in which they are applied. eg. If the most recently copied item was Text, then the Paste option will show "Paste Text".

6.2.2.1 Undo

Edit | Undo (Ctrl+Z)
To Undo the last change made to the current SWiSH Max Movie. SWiSH Max supports unlimited Undo/Redo commands. Almost all changes can be undone.

6.2.2.2 Redo

Edit | Redo (Ctrl+Y)
To Redo the last change made to the current SWiSH Max Movie. SWiSH Max supports unlimited Undo/Redo commands. Almost all changes can be redone.

6.2.2.3 Cut

Edit | Cut (Ctrl+X)
Deletes the currently selected Object or Effect and copies it to the Clipboard. If no Effect is selected the Object will be copied.

6.2.2.4 Copy

Edit | Copy (Ctrl+C)
Copies the currently selected Object, Effect or Scene to the Clipboard. The selected Object, Effect or Scene is *not* deleted. If no Effect is selected, the Object will be copied.

6.2.2.5 Paste

Edit | Paste (Ctrl+V)
Pastes the Object, Effect or Scene in the Clipboard to the current SWiSH Max Movie. If an Object is being pasted, it is added to the current Scene/Movie Clip/Button State/Group in the center of the Layout workspace, at the top of the stacking order. If an Effect is being pasted, it is inserted at the current Frame of the selected Object or selected group of Objects. If a Scene is being pasted, it is inserted after the currently selected Scene in the Movie.

6.2.2.6 Paste In Place

Edit | Paste in Place (Ctrl+Shift+V)
Pastes the copied Object in the same position as the original object. The position is based on the X and Y coordinates of the original object.

6.2.2.7 Delete

Edit | Delete (Delete)
Deletes the currently selected Object or Effect.

6.2.2.8 Select All

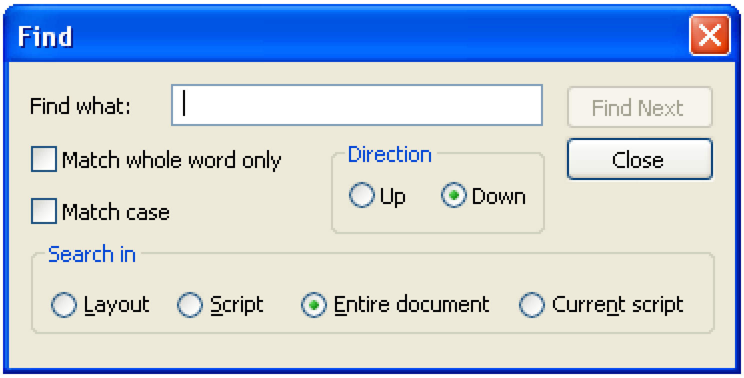
Edit | Select All (Ctrl+A)
Selects all Objects in the current Scene.

6.2.2.9 Deselect All

Edit | DeSelect All (Ctrl+A)
Deselects all Objects in the current Scene.

6.2.2.10 Find

Edit | Find (Ctrl+F)
Lets you search for text in your Movie in the Layout and script.



Find what:

□ Match whole word only

□ Match case

Direction: ○ Up ● Down

Search in: ○ Layout ○ Script ● Entire document ○ Current script

Find what
The text word(s) to look for in selected area.

'Find Next' button (F3) ***(Note: 'Find Previous' - Shift+F3)***
Finds the next result matching the specified criteria.

Match whole word only
If checked, only the whole word(s) are found.

Match case
If checked, search looks to find exact case of the entered search word(s).

Direction (Up/ Down)
Searches Up or Down through the objects as displayed in the Outline Panel, or Script Panel. Search will wrapping around to the start position.

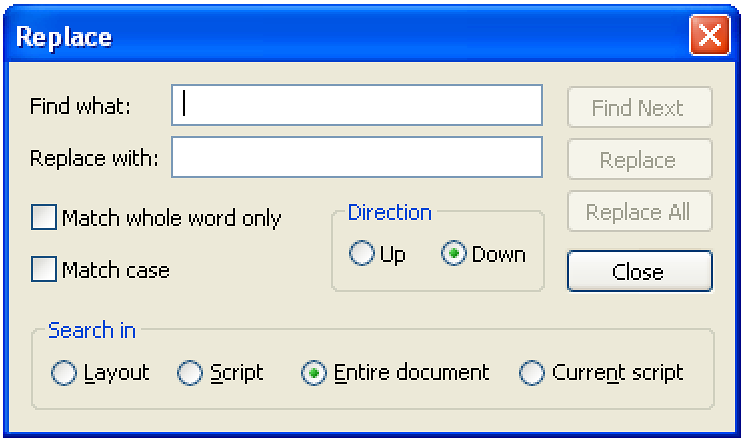
Search

- **Layout:** Searches objects in the layout. Searches in object names and text in all scenes.
- **Script:** Searches in all script in all objects (in all scenes).
- **Entire document:** Searches both Layout and Script. i.e. same as selecting both Layout and Script.
- **Current script:** Searches the script showing in the Script Panel (i.e. for the currently-selected object).

Note: Find does search script comments (e.g. `// comment`)

6.2.2.11 Replace

Edit | Replace (Ctrl+H)
Lets you search and replace text in your Movie in the Layout and script.



Find what:

Replace with:

☐ Match whole word only

☐ Match case

Direction: ○ Up ● Down

Search in: ○ Layout ○ Script ● Entire document ○ Current script

Find what
The text word(s) to look for in selected area.

Replace with
The text word(s) to replace the word(s) found.

'Find Next' button (F3) ***(Note: 'Find Previous' - Shift+F3)***
Finds the next result matching the specified criteria.

'Replace' button
Replaces the last result found.

'Replace All' button
Replaced all results found, wrapping around to cursor position in the direction specified.

Match whole word only
If checked, only the whole word(s) are found.

Match case
If checked, search looks to find exact case of the entered search word(s).

Direction (Up/ Down)
Searches Up or Down through the objects as displayed in the Outline Panel, or Script Panel. Search will wrapping around to the start position.

Search

- **Layout:** Searches objects in the layout. Searches in object names and text in all scenes.
- **Script:** Searches in all script in all objects (in all scenes).
- **Entire document:** Searches both Layout and Script. i.e. same as selecting both Layout and Script.
- **Current script:** Searches the script showing in the Script Panel (i.e. for the currently-selected object).

6.2.2.12 Properties

Edit | Properties (Alt+Enter)
Displays the properties of the selected Object or Tool in Object Panel.

6.2.2.13 Preferences

Edit | Preferences
Opens '**Preferences**' dialog boxes for setting options in SWiSH Max.

6.2.3 View Menu

The View Menu enables you to show and hide windows and change the view of the Movie.

The available options are as follows:

Zooming

- Zoom in
- Zoom out
- View at 100%
- Fit Scene in Window
- Fit Objects in Window
- Adjust Zoom on Layout Changes

Rulers, Grids, Guides and snapping

- Show Rulers
- Show Grid
- Show Guides
- Insert Horizontal Guide
- Insert Vertical Guide
- Lock Guides
- Clear All Guides
- Snap to Grid
- Snap to Pixel

- Snap to Object Handles
- Snap to Guides

6.2.3.1 Toolbars

View | Toolbars
Shows or hides a specific Toolbar. See the Toolbars topic for a list of Toolbars.

6.2.3.2 Zoom In

View | Zoom In (Ctrl+'+')
To increase the zoom factor of the Layout Panel, such that the Movie is displayed 50% larger than its previous size.

6.2.3.3 Zoom Out

View | Zoom Out (Ctrl+'-')
To decrease the zoom factor of the Layout Panel, such that the Movie is displayed at two-thirds its previous size.

6.2.3.4 View at 100%

View | View at 100% (Ctrl+1)
Views the Movie at its actual size in the Layout Panel.

Note: ***Ctrl+1*** refers to the "1" key on the keyboard only, not the keypad.

6.2.3.5 Fit Scene in Window

View | Fit Scene in Window (Ctrl+2)
To fit the entire Scene (stage area) into the Layout Panel.

Note: ***Ctrl+2*** refers to the "2" key on the keyboard only, not the keypad.

6.2.3.6 Fit Objects in Window

View | Fit Objects in Window (Ctrl+3)
To fit all selected Objects into the Layout Panel. If no Object is selected, this function will fit all Objects in the Scene into the Layout Panel.

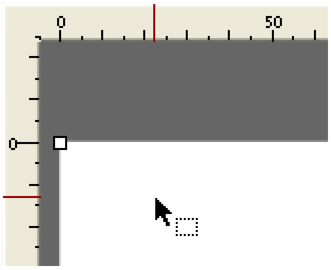
Note: ***Ctrl+3*** refers to the "3" key on the keyboard only, not the keypad.

6.2.3.7 Adjust Zoom on Layout Change

Adjust Zoom on Layout Change
Resizes the zoom view to be proportional to the size of the Layout Panel.

6.2.3.8 Show Rulers

View | Show Rulers (Ctrl+R)
Shows or hides Rulers in the Layout Panel.



The Rulers show the number of pixels from the top left hand corner of the Movie. The number of graduations displayed depends on the current magnification used.

Rulers are especially useful if Guides are used.

6.2.3.9 Show Grid

View | Show Grid (Ctrl+')
Displays or hides the Grid on the Layout Panel. The Grid is a network of evenly spaced horizontal and vertical lines that cover the canvas. It can help you to align your artwork and arrange Objects symmetrically. You can change the Grid settings on the Guides Panel.

See also: Show Grid, Snap to Grid, Guides, Guides Panel.

6.2.3.10 Snap to Grid

View | Snap to Grid (Ctrl+Shift+')
Have Objects Snap to the Grid while moving or editing. It will automatically align Objects along your Grid when they come within a certain distance of it. You can change the Grid settings on the Guides Panel.

See also: Show Grid, Snap to Grid, Guides, Guides Panel.

6.2.3.11 Snap to Pixel

View | Snap to Pixel
Have Objects Snap to Pixels while moving or editing. If Show Grid is selected, a Pixel Grid will also be displayed when the Movie is viewed at >=400% zoom .

6.2.3.12 Snap to Object Handles

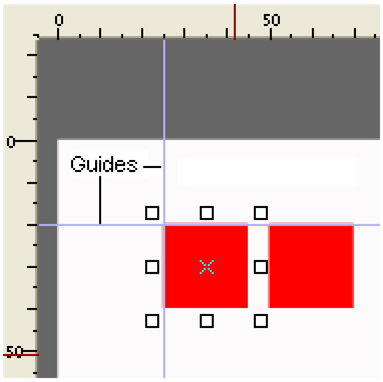
View | Snap to Object Handles (Ctrl+Shift+/)
To have an Object Snap to another Object's Handles while moving or editing. You can change the settings on the Guides Panel.

Note: To snap the Transformation Point to Object Handles hold the ***Shift*** key when moving near the handle.

6.2.3.13 Guides

Guides are a feature that allow the user to specify vertical and horizontal alignment points. They could be considered to be a 'custom grid'. The Guides need not align with the currently selected Grid and work independently of the Grid and Snap to Grid system.

While moving or editing, Objects will automatically align with the Guide when they come within a certain distance of it. This applies if the Snap to Guides option is on.



The following options are provided in the View Menu to allow viewing and setting of various Guide options:

- Show Guides
- Lock Guides
- Clear All Guides
- Snap to Guides.
- Insert Horizontal Guide
- Insert Vertical Guide

The options can also be altered in the Guides Panel.

Although Guides can be used without Rulers, they are easiest to use if the Show Rulers option is used.

Creating New Guides
With the Show Guides option on, Guides can be created in two ways:

- **Dragging:** move the Select tool over the horizontal or vertical Ruler (or over the corresponding edge if the Ruler is not displayed), then press and hold the left mouse button. You can now drag a new Guide line to the desired position on the 'Layout' Panel. Holding down the Shift key while dragging will cause the Guide position to lie on the nearest ruler graduation
- **Ruler Click:** move the Select tool over the horizontal or vertical Ruler. Select the position where you want the guide to be created. Click and release the left mouse button. Clicking on the Horizontal ruler will create a vertical Guide line, while clicking on the Vertical ruler will create a horizontal Guide line. Holding down the Shift key while clicking the mouse button will cause the Guide line to be placed on the nearest ruler graduation.

Moving Guides
To move a Guide, hover the Select tool above the Guide until the cursor changes shape. Then press and hold the left mouse button. You can now drag the Guide to the desired position on the 'Layout' Panel. Holding down the Shift key while dragging will cause the Guide position to lie on the nearest ruler graduation. If an Object is snapped to a Guide, then dragging the guide will drag the Object along with it (provided that Snap To Guides) is toggled on.

Note: It is only possible to move an existing Guide if the Lock Guides option is not set

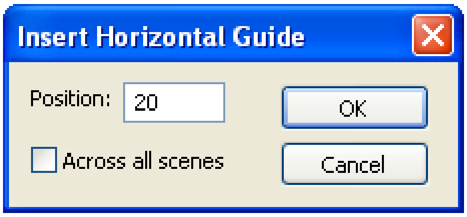
6.2.3.13.1 Show Guides

View | Show Guides (Ctrl+;)
Shows or hides Guides in the Layout Panel.

See also: Show Grid, Snap to Grid, Guides, Guides Panel.

6.2.3.13.2 Insert Horizontal Guide

View | Insert Horizontal Guide
Places a Horizontal Guide at a specific position.



Insert Horizontal Guide

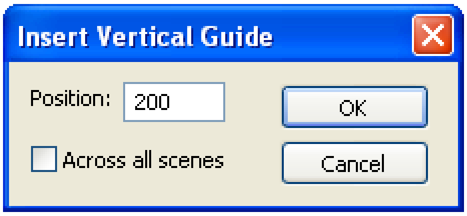
Position: 20

□ Across all scenes

Across all scenes
The Guide will be viewable across all Scenes.

6.2.3.13.3 Insert Vertical Guide

View | Insert Vertical Guide
Places a Vertical Guide at a specific position.



Insert Vertical Guide

Position: 200

□ Across all scenes

Across all scenes
The Guide will be viewable across all Scenes.

6.2.3.13.4 Lock Guides

View | Locks Guides (Ctrl+Alt+;)
If set, prevents currently displayed Guide lines from being moved by the Select Tool in the Layout Panel.

Note: It is still possible to add new Guide lines

6.2.3.13.5 Clear All Guides

View | Clear Guides
Clears all existing Guides in the Layout Panel.

6.2.3.13.6 Snap to Guides

View | Snap to Guides (Ctrl+Shift+;)
Have Objects Snap to the nearest Guide while moving or editing. If set, Objects will automatically align to the Guide when they come within a certain distance of it.

See also: Show Grid, Snap to Grid, Guides, Guides Panel.

6.2.4 Insert Menu

The Insert Menu enables you to add things to the Movie.

The available options are as follows:

Add Scenes

- Scene

Add Objects

- Button
- Movie Clip
- External Media
- Library Symbol
- Import Sound...
- Import Video...
- Import Image...
- Import Animation...
- Import Vector...
- Import Text...

Effects and Scripts

- Effect
- Script

Frames

- Insert Frame(s)
- Delete Frame(s).
- Insert Second
- Delete Second

Keyframes

- Insert Keyframe

6.2.4.1 Scene

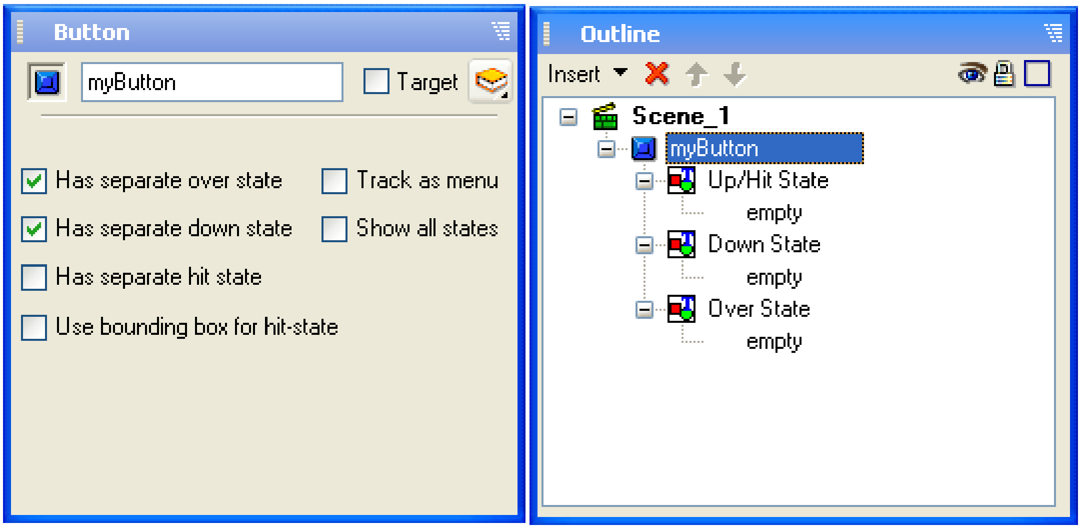
Insert | Scene

Inserts a new Scene into the Movie.

6.2.4.2 Button

Insert | Button

Inserts the structure of a button into the current Scene/Movie Clip/Group. To create the button check the required states in the Button Object Panel. The Outline Panel view will display the state as *empty*. Paste a Shape, Group or Movie Clip into the required button states to complete the button.



Button

myButton ☐ Target

☑ Has separate over state ☐ Track as menu

☑ Has separate down state ☐ Show all states

☐ Has separate hit state

☐ Use bounding box for hit-state

Outline

Insert

Scene_1
myButton
Up/Hit State
empty
Down State
empty
Over State
empty

See Button Object Panel for more detail on Button state options.

6.2.4.3 Movie Clip

Insert | Movie Clip
Inserts a Movie Clip into the current Scene/Movie Clip/Group.

6.2.4.4 External Media

Insert | External Media
Inserts an external media container which can be used to load and play an external file in the current Scene/ Movie Clip/Group.

Note: At the moment only Flash Video (.flv) video can loaded.

Once an external media container is placed the External Media Panel will be visible. See the section on External Media Properties Panel for more information.

6.2.4.5 Library Symbol

Insert | Library Symbol
Insert a symbol from the Library into the current Scene/Movie Clip/Group.

6.2.4.6 Import Sound...

Insert | Import Sound...
Adds a Soundtrack to the selected Scene. Select multiple files in Open sound dialog to import multiple soundtracks.
All Soundtracks use common export settings which is can be set at 'File | Export Settings | Soundtrack'.

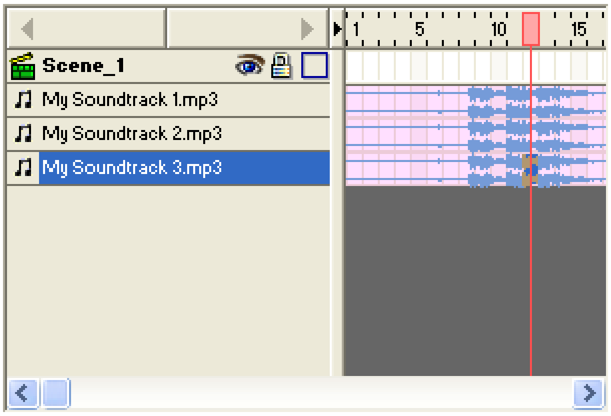
Supported file formats:

- Wave files (*.wav)
- MP3 files (*.mp3)
- Windows Media Audio (*.wma)
- Other audio files (*.au;*.aif;*.aiff;*.snd)

Note:

- SWiSH Max supports DirectShow so if a filter (codec) is installed for a particular audio file SWiSH Max will be able to import that file type. As a general check if Microsoft® Windows Media Player® can play the file, SWiSH Max will be able to import it.

- When importing a sound, SWiSH Max will add the sound as a Soundtrack.
- MP3 sounds are imported without SWiSH Max processing it, so external programs can be used for optimum compression or quality



The Timeline Panel Menu option 'Show Waveforms' toggle display of the audio waveform.

About SWiSH Max sounds: Soundtracks and Event sounds

Event sounds are played in response to some event such as a mouse-click, or when Flash Player reaches a certain frame. Event sounds must be defined (downloaded) before they are used. They can be reused for multiple events if desired. Event sounds may also have a sound 'style' that modifies how the basic sound is played. Streaming sounds are downloaded and played in tight synchronization with the timeline. In this mode, sound packets are stored with each frame.

SWiSH Max supports two types of sound: Soundtracks (or streaming sounds) and event sounds:

- **A Soundtrack** is a sound which plays for the duration of the scene. Soundtracks can be controlled using Place, Remove, Play. The (audio) Play effect allows the user to control the audio track over the desired frame range and the settings are shown in the Audio Effect Panel. A Soundtrack is synchronized with the main timeline so if the main timeline is paused, the soundtrack will also be paused. For example, if a (streaming) Soundtrack plays from frame 10 to frame 20, and an effect in another object plays from frame 10 to frame 20, the effect and the streaming sound are guaranteed to start and end at the same time regardless of how fast the machine is. However, a playSound() action is not synchronized with the timeline and results may differ from machine to machine. To place a Soundtrack select 'Insert | Import Sound...' (as above) or select and drag from the Library to the Layout Panel (stage).

Note: Soundtracks inside Movie Clip will ***NOT*** stream because a Movie Clip has to be fully loaded before it can play at all. Only soundtracks on the main timeline of the Scene are streamed.

Tip: Sometimes it may be necessary to synchronize an event or action with a sound. A the soundtrack position can be found by either:

- playing and watching the playhead move (scroll the timeline manually to follow it) and note the frame numbers during playback. This only works for scenes.
- right-click and drag to select a range of frames in the top row of the timeline and release to select 'Play Frame(s). This will will loop over the selected frames (and for scenes, the playhead will move). Sound will only play if more than one frame is selected.

- **Event Sounds** are played in response to some event such as a mouse-click, or when Flash Player reaches a certain frame. Event sounds must be defined (downloaded) before they are used. They can be reused for multiple events if desired. Event sounds may also have a sound ‘style’ that modifies how the basic sound is played. An event sound is controlled by the events playSound(), stopSound() and stopAllSounds(). To place an event sound import a sound to the Library, then use the menu sequence 'Insert | Script | Sound | playSound()' to add a playSound() event.

6.2.4.7 Import Video...

Insert | Import Video...
Imports a video media file and converts it to Flash Video as either External FLV (creating an External Media Object with controls) or Embedded Video Object.

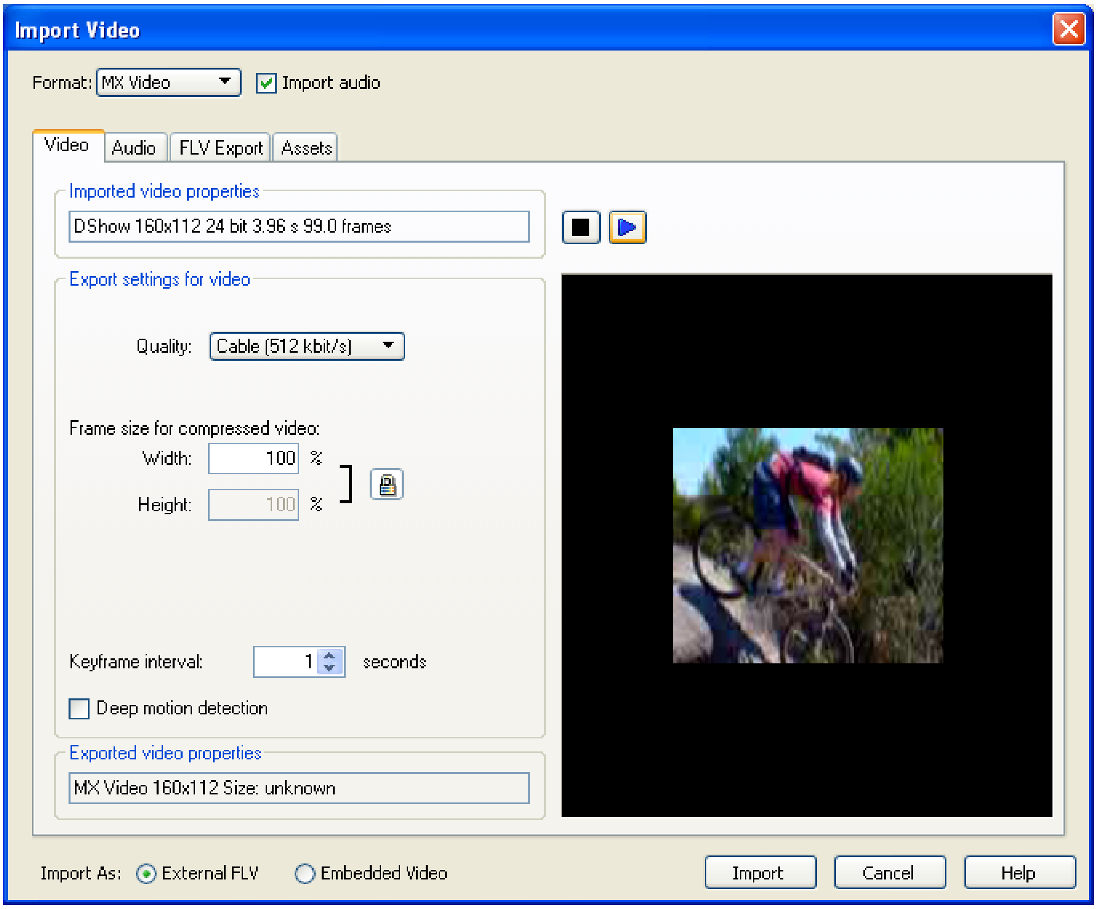
Note: Manipulation of the video file within SWiSH Max is processor (CPU) and memory intensive and you will notice degraded performance.

Supported file formats:

- Adobe Flash Video (*.flv)
- AVI files (*.avi)
- Quicktime Video files (*.mov, *.qt)
- MPEG video files (*.mpg)
- Windows Media Video files (*.wmv, *.asf)

Note: This is the same operation as:

- ''File | Import to Stage | Video...'
- Content Panel 'Import to Stage' ...Video'
- File | Convert Video to FLV *(only converts to an external FLV. Does not make an Object in SWiSH Max.)*



Import Video

Format: MX Video ☑ Import audio

Imported video properties

DShow 160x112 24 bit 3.96 s 99.0 frames

Export settings for video

Quality: Cable (512 kbit/s)

Frame size for compressed video:

Width: 100 %

Height: 100 %

Keyframe interval: 1 seconds

□ Deep motion detection

Exported video properties

MX Video 160x112 Size: unknown

Import As: ● External FLV ○ Embedded Video

Format

Flash Video is available in two formats:

- **MX Video**: So named as it was available with the release of Macromedia Flash MX and Player version 6 (SWF6). This is a variant of the H263 video standard and is based on the Sorenson Spark codec. This is the recommended format.
- **Screen Video**: This is a simple screen capture codec. The aim of this format is to encode "screenshot like" videos, like desktop capture where: there are many hard edges (frames, borders etc.); there is a few things that change in scenes, it can be compared to animated PNG. If used to encode movies with "photo like" content it will generates large files.
- Flash 8 Video
- Audio only

Audio can be enabled or disabled via the **Import audio** checkbox.

Video / Audio / FLV Export / Assets tabs

See 'Media Details'. The properties presented in the Media Details dialog will change depending on the Format selected. The settings in these tabs set the properties of the imported video. These properties are also available from the Video Object 'Media Details' dialog and are explained in full at the sections:

- **Video properties tab**
- **Audio properties tab**
- **FLV Export properties tab**
- **Assets tab**

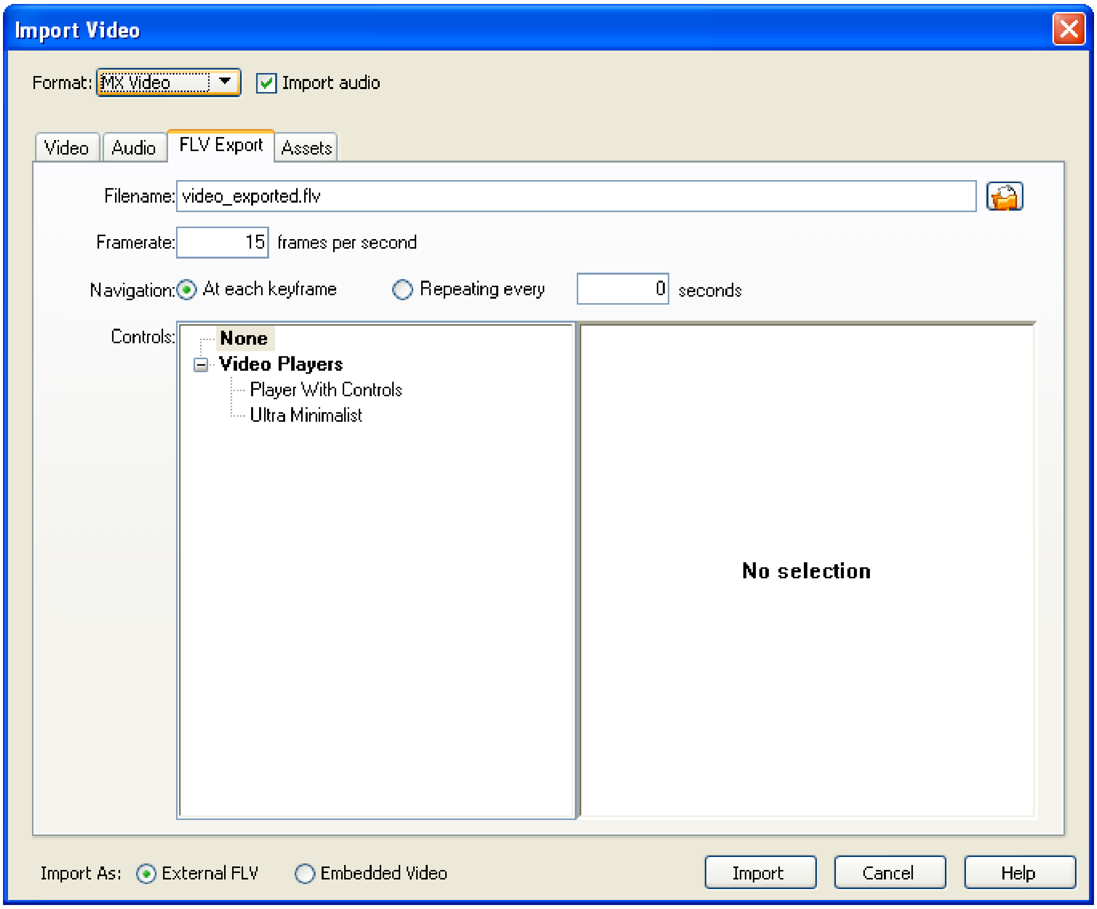
Import As ...

- **Import As 'External FLV'**: The video file is converted as per the settings and saved to an external file. The video file is not saved within the SWiSH Max SWI file. An External Media Object is created with reference to the converted FLV. A video control can be attached to the External Media Object. See the section on FLV Export properties tab for settings specific to FLV conversion. This is the ***default*** setting.
- **Import As 'Embedded Video'**: The video file is embedded within the Movie as a separate video track. Use this option is you want to apply effects or script to the video. It may be necessary to add script to control the movie.

See Video Effect Settings for more details on effects and see Embedded Video Panel or External Media Panel for details on setting up the video when imported.

6.2.4.7.1 FLV Export Properties

This section discusses the FLV Export dialog for the Import Video option. See separate sections for an explanation of settings for Video and Audio.


Import Video
Format: MX Video
☑ Import audio
Video
Audio
FLV Export
Assets
Filename: video_exported.flv
Framerate: 15 frames per second
Navigation: ● At each keyframe ○ Repeating every 0 seconds
Controls:
None
Video Players
Player With Controls
Ultra Minimalist
No selection
Import As: ● External FLV ○ Embedded Video
Import
Cancel
Help

Also see the section 'File | Convert Video to FLV'.

Filename
Target filename for the converted video file.

Note that the full path is normally specified. When uploading your flash movie to your website, you will need to upload flv, html and .swf files.
It is recommended that you export your flash movie to the same folder as your html and swf files.

If you are using the player with controls, you will have to modify the URL reference in the Player component so that it references the relative file name. Failure to do this will prevent the uploaded webpages from displaying the video.

Export Video
This button opens a dialog to select a target video file path/name.

Frame rate
The Flash video (FLV) Frame rate - frames per second of play. The default value matches the source video frame rate. A lower framerate will create a smaller file but will be less smooth.

Navigation
Video is controlled (via Video components, script etc.) by moving between keyframes. Closer keyframes offers more precise control of the video but creates a larger video file. A keyframe each 1 second is recommended.

- **At each keyframe**: Creates a Flash FLV keyframe at the same position of the source video's keyframes.
- **Repeating every [___] second**: Creates a keyframe periodically throughout the video.

Controls
This dialog offers the option to place a video player Component and link the exported FLV file to the component. Resulting component has the video link details in it's Parameters.

6.2.4.8 Import Image...

Insert | Import Image...
Inserts an image from an external file into the current Scene/Movie Clip/Group. This action creates a Shape Object and uses the Image as a fill. See Image Shape Object Properties Panel and Shape Object.

SWiSH Max currently supports the following image formats:

Note: Raster images, for vector images see 'Vector Graphics' below.

Supported file formats:

- Windows bitmap (*.bmp; *.dib)
- GIF image (*.gif)
- JPEG image (*.jpg; *.if, *.jpeg)
- PNG image (*.png)

6.2.4.9 Import Animation...

Insert | Import Animation...
Imports an animation as a sequence of shapes.

Supported file formats:

- Animated GIF (*.gif) (imports as a frame sequence)
- Flash Animation (*.swf both compressed and uncompressed)
- Flash Projector (*.exe)
- SWiSH Max Movie (*.swi)

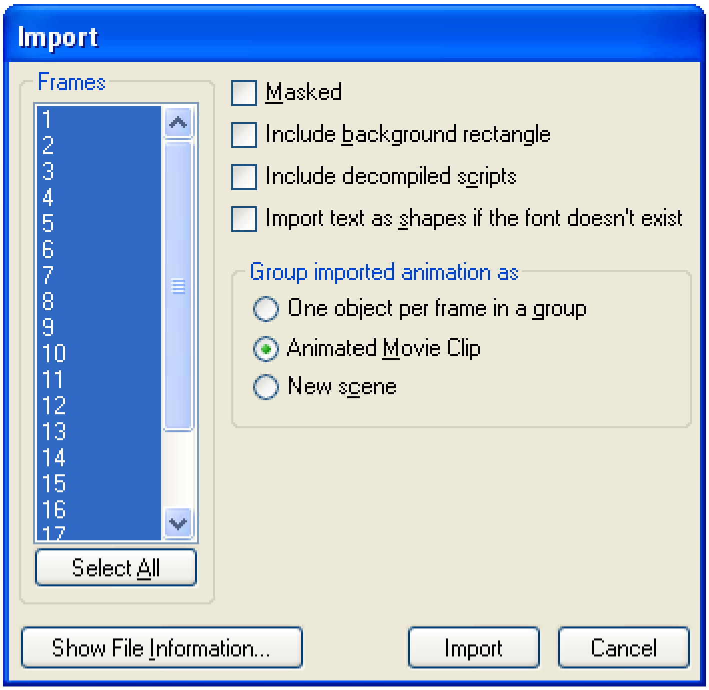
Flash Animation
When importing a Flash Animation (*.swf) file, SWiSH Max does not import any sounds, morph shapes, external video streams or Flash clip events. Clipping (mask) layers are imported, but are not automatically set as a Mask Object in SWiSH Max. The following Actions are imported, all other Actions are ignored:

- setLabel
- play

- stop
- gotoAndPlay
- gotoAndStop
- stopAllSounds
- nextFrame
- prevFrame
- getURL
- fscommand
- javascript
- mailto
- loadMovie
- if frame loaded
- name = expression;

Import Frames

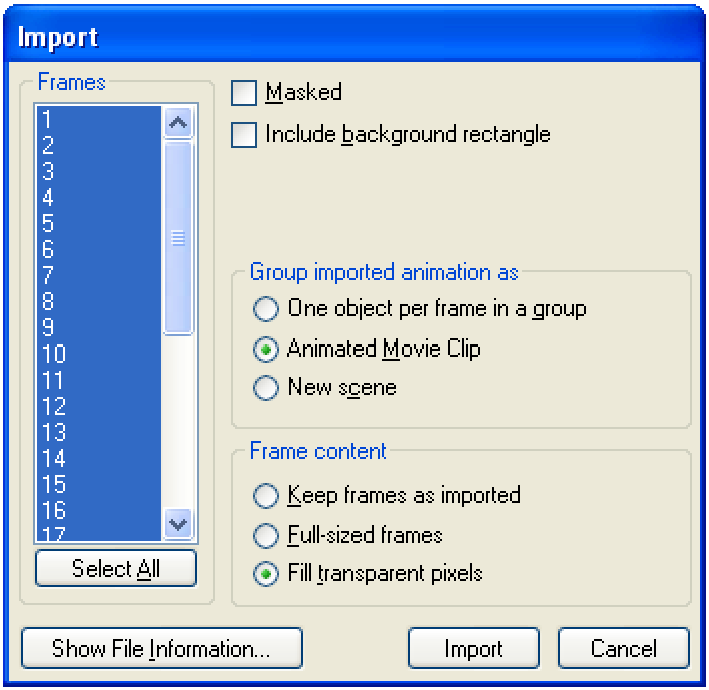
The 'Import Frames' dialog box appears when importing an animation, use this dialog box to select the Frames to import. Options will vary depending on the content imported.



□ Masked
□ Include background rectangle
□ Include decompiled scripts
□ Import text as shapes if the font doesn't exist

○ One object per frame in a group
● Animated Movie Clip
○ New scene

Options for importing a SWF



□ Masked

□ Include background rectangle

Group imported animation as

○ One object per frame in a group

● Animated Movie Clip

○ New scene

Frame content

○ Keep frames as imported

○ Full-sized frames

● Fill transparent pixels

Options for importing a GIF animation

Select All Frames button
To select all Frames in the Frame list.

Masked
The imported Movie Clip will be masked by the background rectangle if this option is checked.

Import background rectangle
The background rectangle will be imported if this option is checked.

Include decompiled scripts
SWiSH Max will attempt to decompile the compiled script back in to SWiSH Max script. Support for this is limited and only simple script without any conditions or loops will decompile successfully.

Import text as shapes if the font doesn't exist
Will convert all fonts not currently installed on the user's system to shapes (text will not be editable but will retain its original aesthetics).

Group Imported animation as:

One object per frame in a group
With this option, a separate group is generated for each frame of the movie which contains all the objects visible in that frame. All of those individual frame groups are placed in one parent group. The result will not be animated; however, effects can be applied to the group to play the frames in sequence.

Animated Movie Clip
The selected Frames will be imported as a Movie Clip if this option is checked. Otherwise, the selected Frames will be imported as a group of pictures.

New Scene
Will import content as a new scene with individual objects in the Outline tree and animated in the timeline.

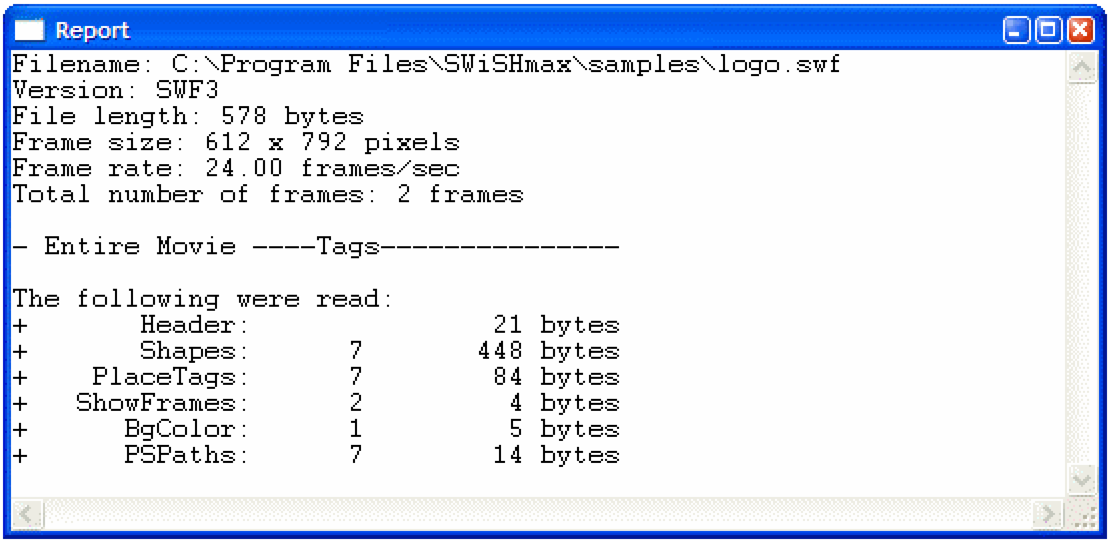
Frame content:

Keep Frame as Imported
Leaves frames as defined in the file. Frames may have a transparent background and be smaller than the size of the whole image.

Full-sized frames
Forces all frames to be opaque and makes them the same (full) size, so the whole image is updated.

Fill transparent pixels
Makes each frame opaque but keeps original sizes, so only part of image is updated.

Show File Information...
Displays the details of the file to be imported.

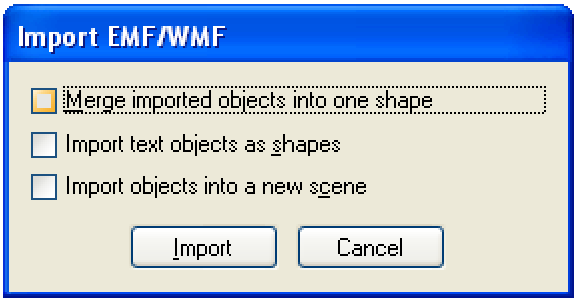


```
Filename: C:\Program Files\SWiSHmax\samples\logo.swf
Version: SWF3
File length: 578 bytes
Frame size: 612 x 792 pixels
Frame rate: 24.00 frames/sec
Total number of frames: 2 frames

- Entire Movie ----Tags---------------

The following were read:
+        Header:              21 bytes
+        Shapes:       7     448 bytes
+     PlaceTags:       7      84 bytes
+    ShowFrames:       2       4 bytes
+       BgColor:       1       5 bytes
+       PSPaths:       7      14 bytes
```

6.2.4.10 Import Vector...

Insert | Import Vector...
Imports vector graphics as a vector shape.

Supported file formats:
- Windows Metafile (*.wmf)
- Enhanced Metafile (*.emf)
- Flash Graphics (*.swf both compressed and uncompressed)



□ Merge imported objects into one shape
□ Import text objects as shapes
□ Import objects into a new scene

Merge imported objects into one shape
Merges all vector information into one Shape Object, otherwise creates a Group Object containing multiple shape objects.

Import text objects as shapes
Imports text characters as separate Shape Objects, otherwise creates a Group Object containing multiple shape objects.

Import objects into a new scene
Creates a new scene and places new objects.

6.2.4.11 Import Text...

Insert | Import Text
Inserts text into the current Scene/Movie Clip/Group. You can edit the text in the Text Panel.

Supported file formats:

- Plain text (*.txt)

6.2.4.12 Effect

Insert | Effect
Inserts an Effect to the selected Object or selected group of Objects at the current Frame. The Submenu has a list of the available Effects. Some of the Effects have a further submenu where you can select the basic Effect, or one of the preset Effects you have recently used (saved or loaded).

6.2.4.13 Script

Insert | Script
Adds a Script to the selected Object or the current Frame. You can either add Actions, Events or Define Functions. The Script can then be edited in the Script Panel. See the Scripting section for more information about general scripting concepts.

6.2.4.14 Insert Frame(s)

Insert | Insert Frame(s) (F5)
Inserts a new Frame before the current Frame in the Timeline.

To insert multiple frames, right-click with the mouse and drag along the timeline to select the range then select the 'Insert Frames' from the context menu.

6.2.4.15 Delete Frame(s)

Insert | Delete Frame(s) (Shift+F5)
Deletes the current Frame from the Timeline.

To delete multiple frames, right-click with the mouse and drag along the timeline to select the range then select the 'Delete Frames' from the context menu.

6.2.4.16 Insert Second

Insert | Insert Second (Ctrl+F5)
Inserts a seconds worth of Frames from the Timeline (one second is based on the number of frames set for the Framerate on the Movie panel).

6.2.4.17 Delete Second

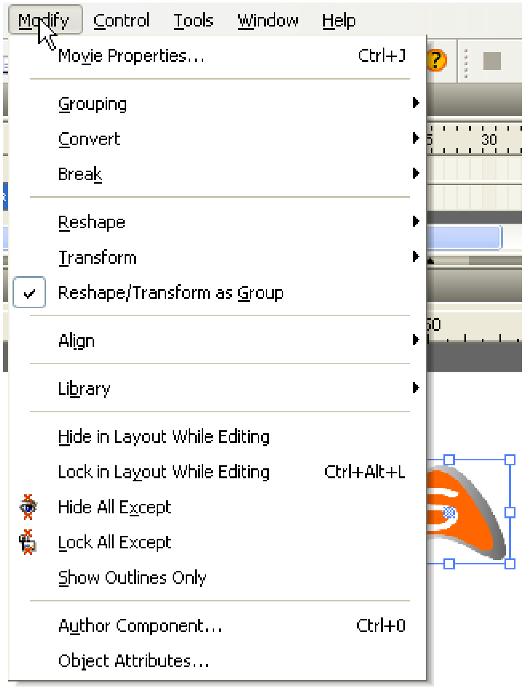
Insert | Delete Second (Ctrl+Shift+F5)
Deletes a seconds worth of frames from the Timeline (one second is based on the number of frames set for the Framerate on the Movie panel).

6.2.4.18 Insert Keyframe

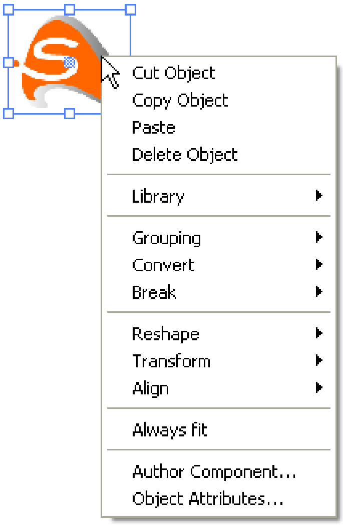
Insert | Insert Keyframe
Used to add new Move effects or split a current Move effect into two separate Move effects in the Timeline.

6.2.5 Modify Menu

The Modify Menu enables you to change the properties of the currently selected Object(s). The Modify Menu is available from the main menu and from the (right click) context menu.


Modify
Control
Tools
Window
Help
Movie Properties... Ctrl+J
Grouping
Convert
Break
Reshape
Transform
✓ Reshape/Transform as Group
Align
Library
Hide in Layout While Editing
Lock in Layout While Editing Ctrl+Alt+L
Hide All Except
Lock All Except
Show Outlines Only
Author Component... Ctrl+0
Object Attributes...

Modify Menu for selected Image



Modify Context Menu for selected Image

The properties available depend on the selected Object(s). The following may be available:

Properties and authoring

- Author Component...
- Object Attributes...
- Movie Properties

Grouping, Convert and Break

- Grouping
- Convert
- Break

Change layer and Transform

- Order
- Transform
- Reshape/Transform as Group

Arranging and display

- Arrange
- Align

- Hide in Layout While Editing
- Lock in Layout While Editing
- Hide All Except
- Lock All Except
- Show Outlines Only

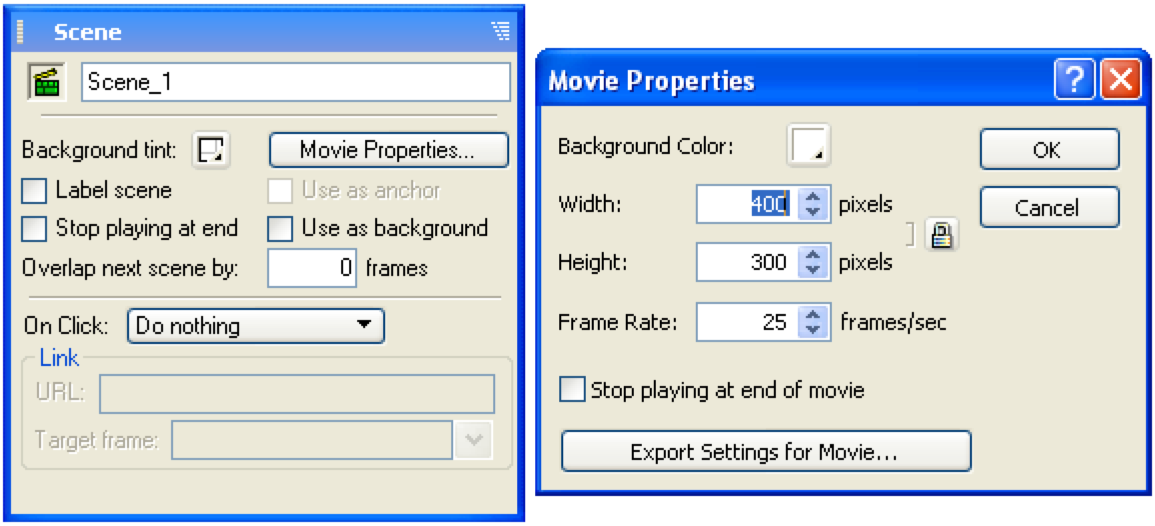
Library

- Library

6.2.5.1 Movie Properties

Modify | Movie Properties (Ctrl+J)

The Movie Properties dialog contains properties that are common to the entire Movie. The dialog is accessible from the Scene Panel or from the hot key function Ctrl+J.



Scene

Scene_1

Background tint:

□ Label scene

□ Use as anchor

□ Stop playing at end

□ Use as background

Overlap next scene by: 0 frames

On Click: Do nothing

Link

URL:

Target frame:

Movie Properties

Background Color:

Width: 400 pixels

Height: 300 pixels

Frame Rate: 25 frames/sec

□ Stop playing at end of movie

Movie Properties appears when the 'Movie Properties' button on the Scene Panel is pressed

Background Color
Lets you change the background color of the Movie using the Color Picker. There is no alpha value for this color selection. This color is also used as the player background color when you export to HTML (unless you manually edit the HTML code). The default color is white.

Width and Height
Lets you change the size of the stage of the Movie in pixels. This size is also used when you export to HTML, unless you use the Size 100% option, or manually edit the HTML code. The default setting is 400x300 pixels. The maximum size for the movie is 3277x3277 pixels. This is due to the limitations of the SWF format.

Frame Rate
Lets you change the speed of the Movie in Frames per second. The greater the number of Frames per second, the faster the animation will play. The default setting is 12 Frames per second.

Stop playing at end of movie

Checking this option automatically adds a Stop Movie Action at the end of the Movie.
Note: This checkbox will only stop the movie in the main timeline, it will not stop any Movie Clips that are currently playing. This is a Flash limitation. If necessary, use the stop() action to stop any Movie Clips.

Export Settings for Movie...
This button opens the Export settings also available from File | Export Settings....

6.2.5.2 Split Into Smaller Scenes

Modify | Split Into Smaller Scenes
This operation will split a Scene into two Scenes with the Scene name suffixed <*Scenename*>_1, and <*Scenename*>_2. Only a Scene which has a number of Objects in obviously separate positions (on the Timeline) can be split. The option will report if it is, or isn't, possible to do the split operation. It is possible to split with some crossover; in this situation the Scenes overlap.

Note:

- Scene crossover is visible if the option 'Show Scene Overlaps' is selected in the Timeline Panel Menu
- To see the effect of splitting scenes, first merge two scenes, then split them again

6.2.5.3 Merge Scenes Into One

Modify | Merge Scenes Into One
This operation is visible if more than one sequential Scene is selected. Only sequential Scenes can be merged.

Note: Multiple Scenes can be selected on the Outline Panel using Ctrl+click for individual Scenes and Shift+click to select a range of Scenes (Shift+click first and last scene in the range).

6.2.5.4 Grouping

Modify | Grouping | Group as Group (Ctrl+G)
To Group selected Objects together. Complex Effects can then be applied to the Group.

Note: Because a Group does not have its own Timeline, SWiSH Max converts any Objects with Effects into Movie Clips before grouping.

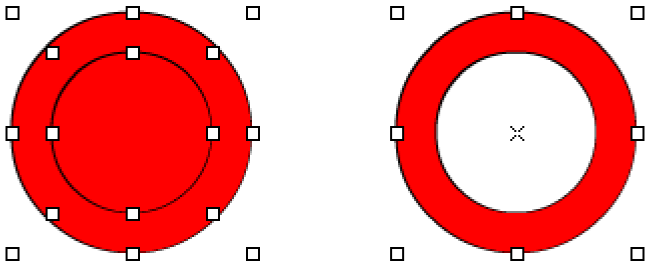
Modify | Grouping | Group as Button
Groups the selected Objects together as a Button.

Modify | Grouping | Group as Movie Clip
Groups the selected Objects together as a Movie Clip.

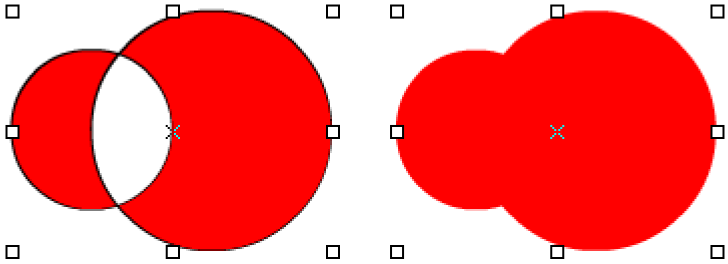
Modify | Grouping | Group as Shape
Groups the selected Objects together as a single shape.

Note:

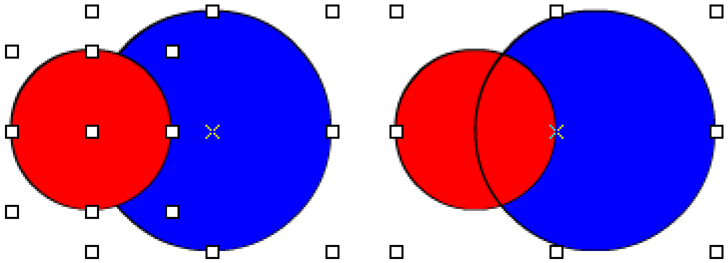
- If you want to use multiple Text or Shape Objects as a mask, you should group them as a shape
- Objects with Effects/Events cannot be grouped as a single shape, as you will lose the Effects and Events. You should manually remove these before grouping, if desired
- When grouping as a single shape, you will be asked whether to 'Make the overlapped regions of objects with the same fill style empty?'. If you select yes, the overlapped regions of Objects with the same fill style will become empty. You can use this feature to punch a hole in a shape as shown below



- If you use the merged shape as a mask, any overlapping regions will be empty (see below left). If you want the overlapping regions to be filled, then check the 'Fill overlaps' option in the Shape Panel. If you use this setting on a shape that is *not* being used as a mask, then the shape will always have a solid fill with no outline, regardless of the fill and line styles specified



- After grouping as a single shape, the edges of the Object behind are no longer covered by the object in front, as shown below



Modify | Grouping | Ungroup (Ctrl+U)
Splits a Group or Movie Clip into separate Objects.

6.2.5.5 Convert

Modify | Convert | Convert to Button
Converts the selected Objects into individual Buttons.

Modify | Convert | Convert to Movie Clip
Converts the selected Object into individual Movie Clips.

6.2.5.6 Break

Modify | Break | Break into Letters
Converts a Text Object into a Group Object containing separate individual Text Objects for each letter of the text. You can edit each letter Object individually (for example to change the position or orientation of a

letter). You can apply an Effect to the Group Object to animate each letter.

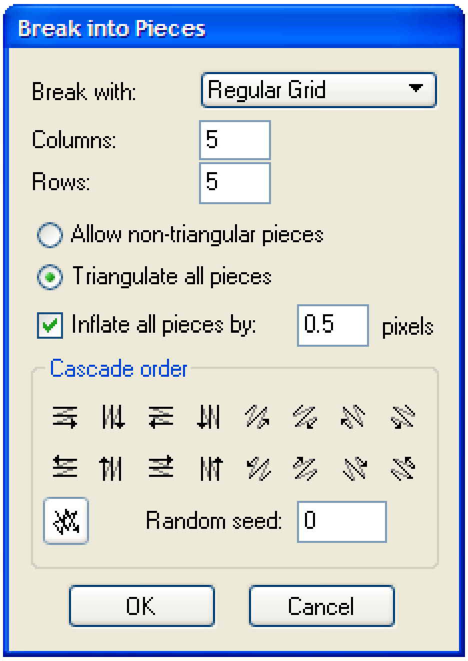
Modify | Break | Break into Shapes

Converts a complex shape into a Group Object containing separate individual Shape Objects for each simple component of the original complex shape.

Converts a Text Object into a Group Object containing separate individual Shape Objects for each letter of the text. You can edit each Shape Object individually (for example to change the fill etc). You can apply an Effect to the Group Object to animate the shapes elements.

Modify | Break | Break into Pieces

Converts the selected Object into a Group Object of smaller shapes.



Break with: Regular Grid
Columns: 5
Rows: 5
○ Allow non-triangular pieces
● Triangulate all pieces
☑ Inflate all pieces by: 0.5 pixels
Random seed: 0

Break With
The type of shape to break the Object into. The available options are Regular Grid, Triangular Mesh, Random Triangles and Random Polygons.

Column (regular grid or triangular mesh only)
Defines the number of columns the Object is broken into when breaking into a Regular Grid or Triangular Mesh.

Row (regular grid or triangular mesh only)
Defines the number of rows the Object is broken into when breaking into a Regular Grid or Triangular Mesh.

Allow non-triangular pieces
When checked, pieces may or may not include rectangles.

Triangulate all pieces
When checked, all non-triangular pieces will be split into triangles further.

Inflate all pieces by # pixels
Will allow you to increase the size of each piece slightly so that there are no gaps in between each

piece.

Number (Random Polygons only)
Defines the number of pieces the Object is broken into when breaking into a Random Triangles or Triangular Mesh.

Minimum Number (Random Triangles only)
Defines the minimum number of pieces that the shape will be broken in to.

Random Seed
Every distinct value for random seed will give a different result for the random arrangement. Although using the same random seed will produce the same arrangement each time.

Cascade Order
The order things will animate when using the Cascade options in an Effect.

6.2.5.7 Reshape

Modify | Reshape | Rotate 90
Rotates the selected Object by 90 degrees.

Modify | Reshape | Rotate 180
Rotates the selected Object by 180 degrees.

Modify | Reshape | Rotate -90
Rotates the selected Object by -90 (270) degrees.

Modify | Reshape | Flip Horizontal
Flips the selected Object horizontally.

Modify | Reshape | Flip Vertical
Flips the selected Object vertically.

6.2.5.8 Transform

Modify | Transform | Rotate 90 (Ctrl+Shift+7)
Rotates the selected Object by 90 degrees.

Modify | Transform | Rotate 180
Rotates the selected Object by 180 degrees.

Modify | Transform | Rotate -90 (Ctrl+Shift+9)
Rotates the selected Object by -90 (270) degrees.

Modify | Transform | Flip Horizontal

Flips the selected Object horizontally.

Modify | Transform | Flip Vertical
Flips the selected Object vertically.

Modify | Transform | Reset
Resets the Transform to default settings, except for position.

6.2.5.9 Reshape/Transform as Group

Reshape / Transform as Group
'Reshape / Transform as Group' toggles the mode to change the selected, or grouped, objects as either a group or independently.

6.2.5.10 Arrange

Modify | Arrange | Bring to Front
Moves the selected Object in front of all other Objects.

Modify | Arrange| Send to Back
Moves the selected Object behind all other Objects.

Modify | Arrange| Bring Forward
Moves the selected Object one step closer to the top of the stack of Objects.

Modify | Arrange| Send Backward
Moves the selected Object one step closer to the bottom of the stack of Objects.

6.2.5.11 Align

Align specifies what you mean by the 'position' of the Object. For example, when you talk about the position of the Object, you may be referring to where the center of the Object is. The alignment position can be one of nine preset positions on the corners, edges and center of the object. The alignment position is also the center for rotation and scaling of the Object. You can see the coordinates of the alignment position in the 'Transform' Panel.

Modify | Align | Left (Ctrl+Alt+1)
Align selected Objects to the left.

Modify | Align| Center (Horizontally) (Ctrl+Alt+2)
Align selected Objects to the center horizontally.

Modify | Align | Right (Ctrl+Alt+3)
Align selected Objects to the right.

Modify | Align | Anchor (Horizontally)
Align selected Objects horizontally by anchor point.
Note: "Anchor" is the Reference point anchor.

Modify | Align | Top (Ctrl+Alt+4)

Align selected Objects to the top.

Modify | Align | Center (Vertically) (Ctrl+Alt+5)
Align selected Objects to the center vertically.

Modify | Align | Bottom (Ctrl+Alt+6)
Align selected Objects to the bottom.

Modify | Align | Anchor (Vertically)
Align selected Objects vertically by anchor point.
Note: "Anchor" is the Reference point anchor.

Modify | Align | Distribute | Left
Distribute selected Objects horizontally by left side.

Modify | Align | Distribute | Center (Horizontally)
Distribute selected Objects horizontally by center point.

Modify | Align | Distribute | Right
Distribute selected Objects horizontally by right side.

Modify | Align | Distribute | Anchor (Horizontally)
Distribute selected Objects horizontally by anchor point.
Note: "Anchor" is the Reference point anchor.

Modify | Align | Distribute | Top
Distribute selected Objects vertically by top side.

Modify | Align | Distribute | Center (Vertically)
Distribute selected Objects vertically by center point.

Modify | Align | Distribute | Bottom
Distribute selected Objects vertically by bottom side.

Modify | Align | Distribute | Anchor (Vertically)
Distribute selected Objects vertically by anchor point.
Note: "Anchor" is the Reference point anchor.

Modify | Align | Space Evenly | Horizontally (Ctrl+Alt+7)
Space selected Objects evenly horizontally.

Modify | Align | Space Evenly | Vertically (Ctrl+Alt+8)
Space selected Objects evenly vertically.

Modify | Align | Space Evenly | Both
Space selected Objects evenly both horizontally and vertically.

Modify | Align | Make Same | Width (Ctrl+Shift+Alt+7)
Make selected Objects the same width.

Modify | Align | Make Same | Height (Ctrl+Shift+Alt+9)
Make selected Objects the same height.

Modify | Align | Make Same | Both
Make selected Objects the same height and width.

Modify | Align | Make Same | By Scaling
Scale selected Objects when changing size.

Modify | Align | Make Same | By Resizing
Resize selected Objects when changing size.

Modify | Align | Relative To All Selected
Makes all the alignment options align relative to all the selected Objects.

Modify | Align | Relative To Last Selected
Makes all the alignment options align relative to all the last selected Object.

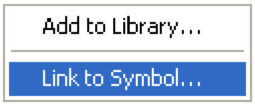
Modify | Align | Relative To Parent
Makes all the alignment options align relative to selected the Object's parent. This is useful for positioning/ aligning Objects within a Movie Clip.

Modify | Align | Relative To Stage (Ctrl+Alt+8)
Makes all the alignment options align relative to the stage.

6.2.5.12 Library

Modify | Library ...

'Add to Library' (F8) / Link to Symbol



'Add to Library' option adds the object to the library or links to an existing Library Object.
Selecting 'Link to Symbol' opens the 'Insert from Library' dialog to select an appropriate Symbol or Resource.

See more details in the section on Library.

6.2.5.13 Hide in Layout while Editing

Modify | Hide in Layout while Editing
Hides the selected Object(s). However, hidden Objects will still appear when the Movie is played. The Object can also be hidden via the Outline Panel.

6.2.5.14 Lock in Layout while Editing

Modify | Lock in Layout while Editing
Locks the selected Object(s) from being edited. The Objects will remain locked until the Lock option is reselected to unlock them.
The Object can also be locked via the Outline Panel.

6.2.5.15 Hide all Except

Modify | Hide all Except

Hides all Objects except the selected Object(s). However, hidden objects will still appear when the Movie is played.

6.2.5.16 Lock all Except

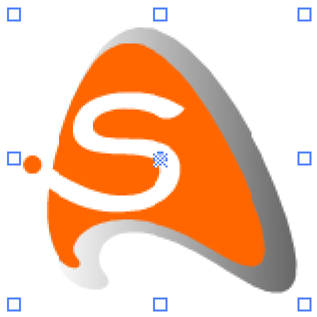
Modify | Lock all Except

Locks all Objects except the Object(s) selected to be edited. The Objects will remain locked until the Lock option is reselected to unlock them.

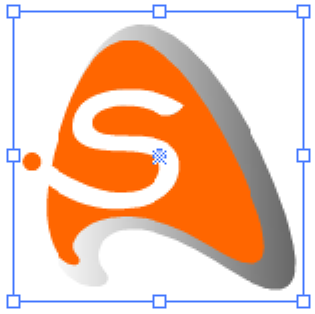
6.2.5.17 Show Outlines only

Modify | Show Outline only

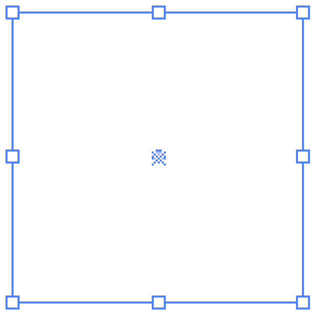
Hides the selected object and displays only the outline of the object.



Object displaying no outlines



Object displaying outlines
View | Show Outlines



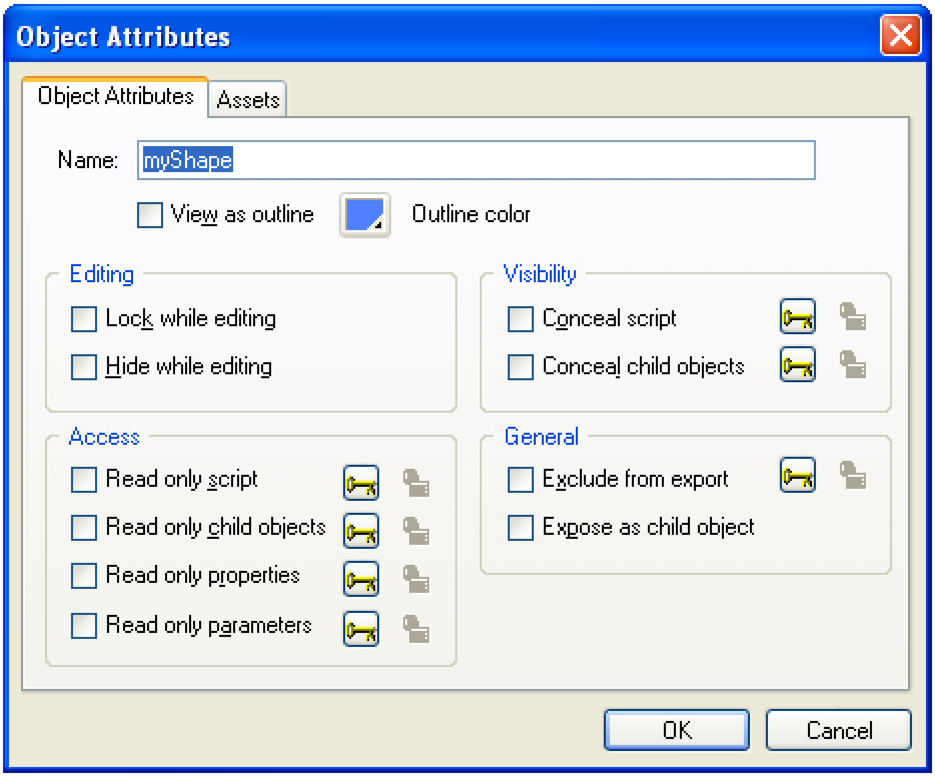
Object displaying only outlines
Modify | Show Outline only

6.2.5.18 Object Attributes

Modify | Object Attributes

This dialog allows the Object attributes to be set or cleared. Though applicable to all Object it is particularly important to the development of Components since locking attributes of a Component can secure it from unwanted change.

Attributes that have a key button can have passwords applied to them so that they cannot be altered by other users unless the password is known. See Password Locking for more information.



Object Attributes

Name: myShape

□ View as outline Outline color

Editing

□ Lock while editing

□ Hide while editing

Visibility

□ Conceal script

□ Conceal child objects

Access

□ Read only script

□ Read only child objects

□ Read only properties

□ Read only parameters

General

□ Exclude from export

□ Expose as child object

Note: for a description of the Assets tab on the Object Attributes dialog see the 'Asset' section of Library.

Name
Object name also shown in the Object Properties Panel (shown here as "myShape").

View as outlines and Outline color
This attribute if set shows the Object as an outline while in the edit mode. This can be useful to allow underlying objects to be shown while in the edit mode. This attribute can also be set by setting the rectangle icon in the Outline Panel. The color of the outline can be selected with the corresponding color selection tool.

There are four attribute groups: Editing, Access, Visibility and General.

Editing
These are attributes associated with the placement and editing of the Object.

Lock while editing This attribute prevents modification of the Object if it is set. The attribute can also be set by setting the padlock icon on the Outline Panel.

Hide while editing
If this attribute is set, the associated Object is hidden from the stage while editing. The attribute can also be set by setting the eye icon in the Outline Panel. The hide attribute can be used to temporarily hide a Object allowing editing of the Objects that are placed below it.

Access

These are attributes associated with the user accessing the parameters and properties of the Object.

Read only script
If set, this attribute prevents other users from modifying the script of the current Object. Note that the script of child Objects can still be modified if "Read only child objects" is not set and the attributes of a child object allow script modification.

Read only child objects
If set, prevents user modification of a child object name and other properties and parameters that may prevent the Object from working correctly. Note that some parameters such as color, font, etc. may still be editable.

Read only properties
If set, prevents the user from modifying any properties of the current Object.

Read only parameters
If set, prevents the user from modifying the component parameters via the component parameters panel.

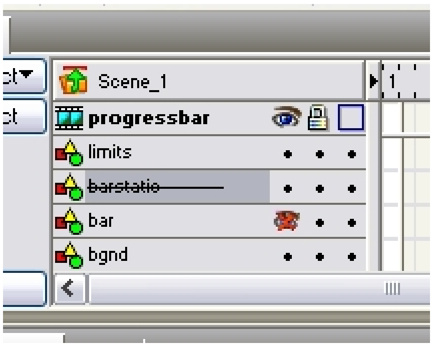
Visibility

Conceal script
This attribute, if set, conceals the script associated with the Object.

Conceal child objects
This attribute if set, conceals all child objects unless a specific child object has the "Expose as child object" attribute set. Setting this attribute prevents people from viewing or altering the contents of a Component.

General

Exclude from export
If set, prevents the current Object from being exported when the .swf file is created. The object appears with a strikethrough in the Outline panel view.



Object with strikethrough is excluded from export

This allows shapes to be added to a Component for the purposes of positioning or instruction without any overhead in the final exported movie.

Expose as child object
If set allows an object to be seen even if the parent object had the "Conceal child objects" attribute set. This is a convenient way to allow end user tailoring of Component parameters without exposing the bulk of the Component scripting and structure.

Note: If any of the following attributes are set, the user cannot access the Author Component dialog: Read only script, Read only properties, Read only parameters or Conceal script.

Useful combinations

Attributes can be used in various combinations to achieve different levels of protection of the intellectual property within your components. Some examples are given below:

1. High Security
Set and password protect the following attributes:

- Conceal script
- Conceal child objects
- Read only script - this is necessary to prevent access to the Author Component dialog.

With these options set, the Component can be resized and the Component parameters can still be modified however the internal structure of the component cannot be seen or modified.

2. Medium Security
Set and password protect the following attributes:

- Conceal child objects
- Read only child objects
- Read only script
- Select child objects have Expose as child object set

The Read only script attribute allows the methods defined in the parent script to be observed. The scripting within the parent object can also contain copyright information which cannot be altered by users without the password. Script that is intended to be kept secret can be placed in a child movie clip that is hidden. The child objects that have the "Expose as a child object" attribute set can be modified by the end user possibly to set final colors, font etc.

3. Trial Distribution
Use either of High or Medium Security options described above and add 'Exclude from export' attribute with a different password.
This configuration allows a users to paste the Component onto a .swi and examine how it works, however the Component cannot be exported in a .swf until the password to unlock the 'Exclude from export' attribute is entered.

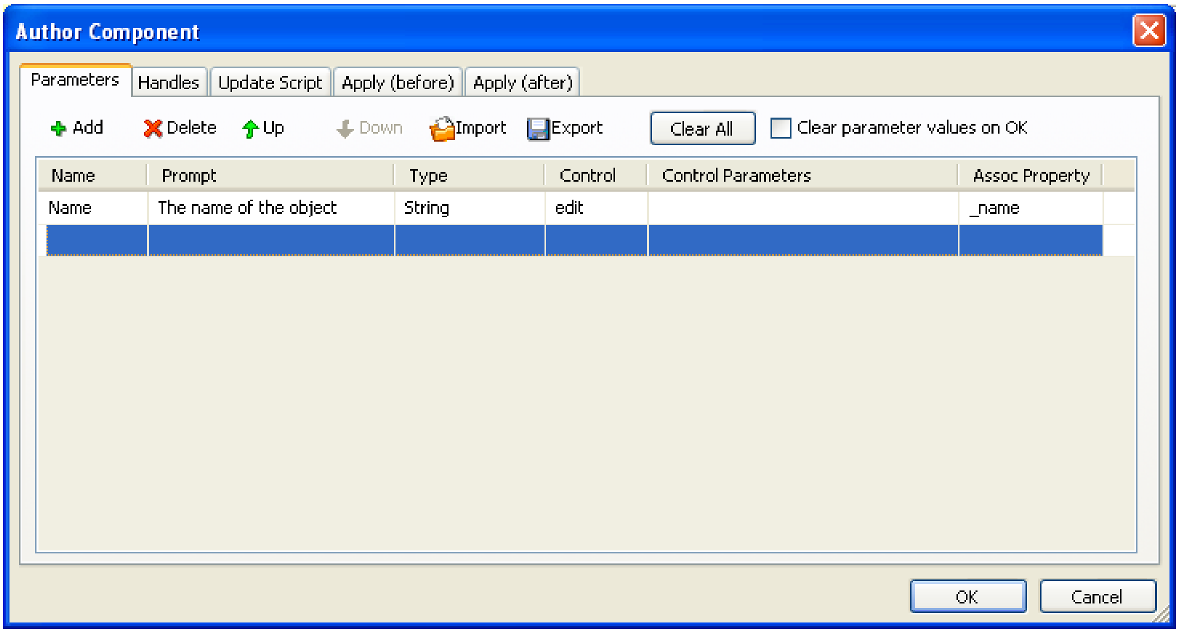
6.2.5.19 Author Components

Modify | Author Component ...(Ctrl+0)

To create a Component, select an Object then open the Author Component Dialog (as shown below). The dialog is also available via the mouse right-click context menu when an Object is selected. See sections on Components and Authoring components.

Each of the tabs in the Author Component dialog is discussed in detail in the following sections: Parameters tab, Scripting tab, Apply (before) tab, Apply (after) tab.

Note: The dialog may not be available if specific Object Attributes have been set. See Object Attributes for more information.



Name	Prompt	Type	Control	Control Parameters	Assoc Property
Name	The name of the object	String	edit		_name

6.2.6 Control Menu

The Control Menu enables you to control how the Movie is played and previewed.

The available options are as follows:

Play

- Play Movie
- Play Timeline
- Play Effect
- Preview Frame

Stop

- Stop

Move the play head

- Step Forward
- Step Back
- Cue to End
- Rewind to Start.

6.2.6.1 Play Movie

Control | Play Movie
Plays the Movie.

6.2.6.2 Play Timeline

Control | Play Timeline
Plays the current Scene or Movie Clip only.

6.2.6.3 Play Effect

Control | Play Effect
Plays the part of the Scene containing the currently selected Effects.

6.2.6.4 Preview Frame

Control | Preview Frame
Switches to 'Preview Frame' mode to preview the current Frame in the 'Layout' window.

In Preview Frame Mode, you can move, scale, rotate and skew Objects in Keyframes using the Select Tool or Reshape Tool.

6.2.6.5 Stop

Control | Stop
Stops playing the Movie, Scene or Effect.

6.2.6.6 Step Forward

Control | Step Forward
Steps forward to the next Frame in 'Preview Frame' mode.

6.2.6.7 Step Back

Control | Step Back
Steps back to the previous Frame in 'Preview Frame' mode.

6.2.6.8 Cue to End

Control | Cue to End
Jumps to the last Frame in 'Preview Frame' mode.

6.2.6.9 Rewind to Start

Control | Rewind to Start
Rewinds to the first Frame in 'Preview Frame' mode.

6.2.7 Tools Menu

The Tools Menu enables you to set application-wide preferences and add your own commands to the SWiSH Max User Interface.

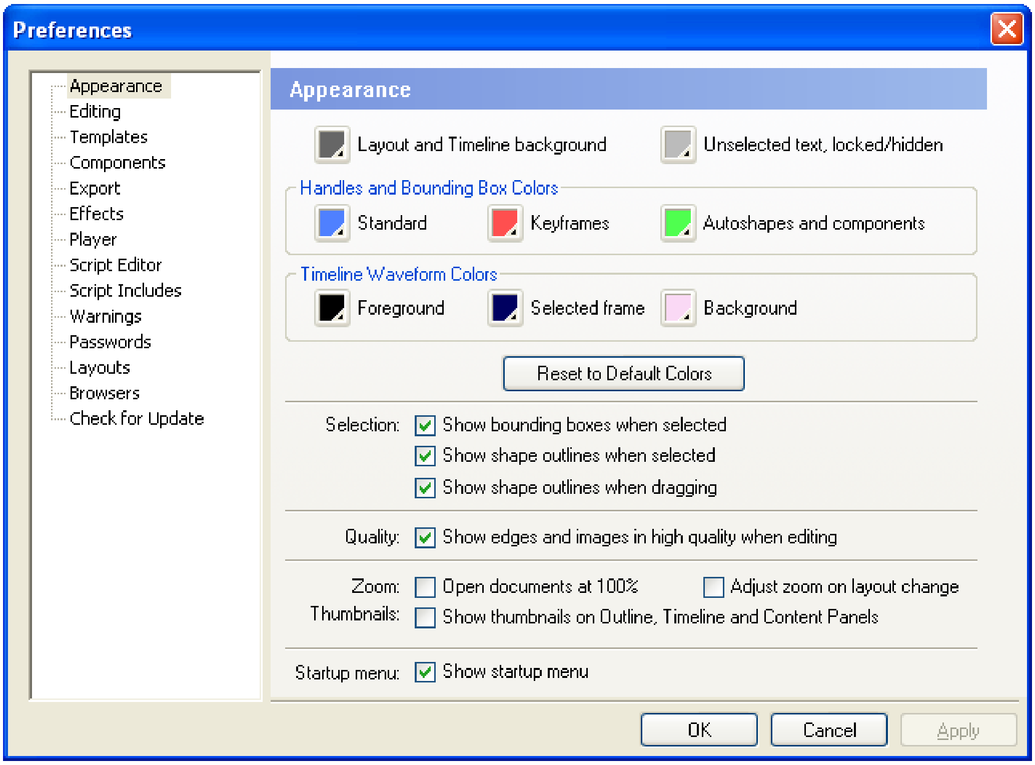
The available options are as follows:

- Preferences
- Customize
- Keyboard Map
- FTP Connection....

You can add additional options to the Tools Menu by selecting the Customize option, and then choosing the Tools tab.

6.2.7.1 Preferences

Tools | Preferences
Opens 'Preferences' dialog boxes for setting options in SWiSH Max.



Layout and Timeline background
Unselected text, locked/hidden

Handles and Bounding Box Colors: Standard, Keyframes, Autoshapes and components

Timeline Waveform Colors: Foreground, Selected frame, Background

Selection: ☑ Show bounding boxes when selected
☑ Show shape outlines when selected
☑ Show shape outlines when dragging

Quality: ☑ Show edges and images in high quality when editing

Zoom: □ Open documents at 100% □ Adjust zoom on layout change

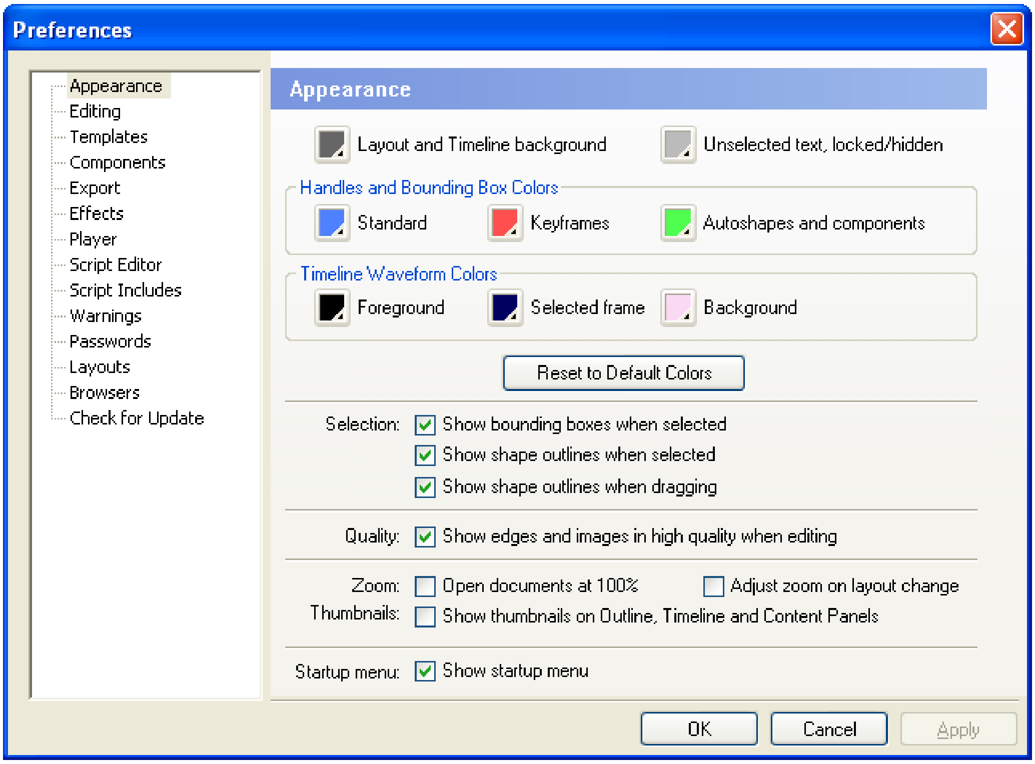
Thumbnails: □ Show thumbnails on Outline, Timeline and Content Panels

Startup menu: ☑ Show startup menu

The SWiSH Max preferences available are separated into the following tabs:

- Appearance
- Editing
- Templates
- Components
- Export
- Effects
- Player
- Script Editor
- Script Includes
- Warnings
- Passwords
- Layouts
- Browsers
- Check for Update

6.2.7.1.1 Appearance



Appearance

Layout and Timeline background | Unselected text, locked/hidden

Handles and Bounding Box Colors: Standard | Keyframes | Autoshapes and components

Timeline Waveform Colors: Foreground | Selected frame | Background

Selection: ☑ Show bounding boxes when selected
☑ Show shape outlines when selected
☑ Show shape outlines when dragging

Quality: ☑ Show edges and images in high quality when editing

Zoom: □ Open documents at 100% □ Adjust zoom on layout change
Thumbnails: □ Show thumbnails on Outline, Timeline and Content Panels

Startup menu: ☑ Show startup menu

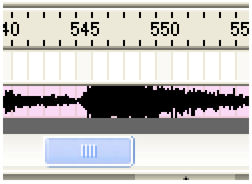
Color Options

Background color for Timeline and Layout panels
Sets the background color of the window for areas outside the Movie in the Layout Panel, and empty space in the Timeline Panel. This uses the Color Picker. Change this setting if there is difficulty seeing window outlines when Panels are dragged, or the Movie background is the same color as the workspace area.

Color for unselected text and locked/hidden objects
Sets the color for unselected text and locked/hidden objects when shown in the Layout Panel. This uses the Color Picker.

Handles and Bounding Box Colors
Sets the color for all handles, bounding-boxes, autoshapes, outlines and Motion Paths and when an object is at a Keyframe as shown in the Layout Panel. This uses the Color Picker.

Timeline Waveforms
Color options for the soundtrack and video sound waveforms shown in the Timeline. These include: Foreground, Foreground (selected frame), Background.

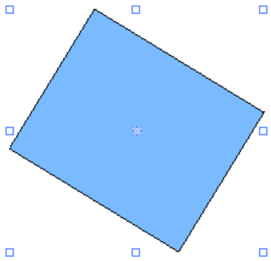


Reset to Default Colors
Resets all colors as per the original SWiSH Max installation.

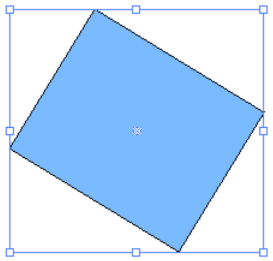
Selection

- **Show Bounding Boxes when selected**

When this option is checked a bounding box is displayed around an Object or Group.



Selected Object with Show Bounding Box option off

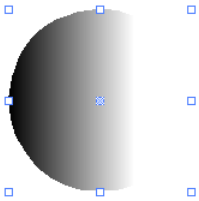


Selected Object with Show Bounding Box option on

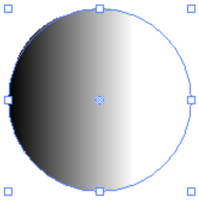
Note: Empty Text Objects will not be visible if this option is not checked. It is recommended that this option be turned on at all times when using Text Objects.

- **Show shape outlines when selected**

When this option is checked the Outline is displayed for the Object.

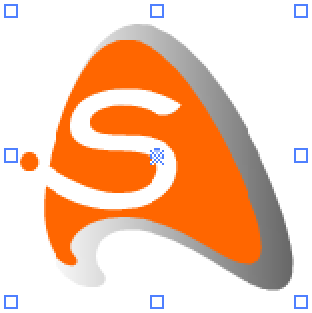


Selected Object with Show Outline option off

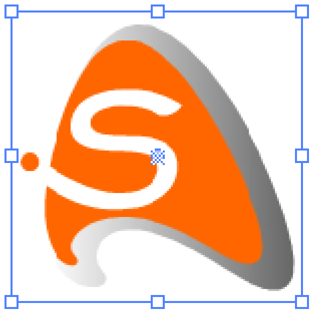


Selected Object with Show Outline option on

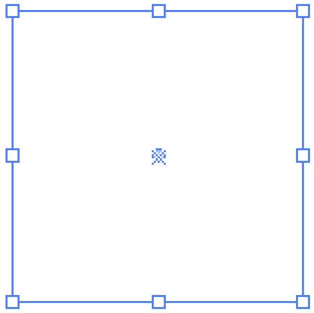
Also see 'Modify | Show Outline only' which shows only the outline and not the object.



Object displaying no outlines



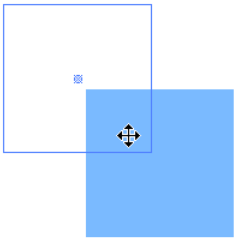
Object displaying outlines
View | Show Outlines



Object displaying only outlines
Modify | Show Outline only

- **Show Shape Outlines When Dragging**

When this option is checked the Outline is displayed whilst dragging an Object.



Outline of Object is shown when dragged

Zoom:

- **Open documents at 100%** - Use this checkbox so that documents always open at 100% zoom.
- **Adjust zoom on layout change** - Use this checkbox so that zoom is adjusted if layout changes.

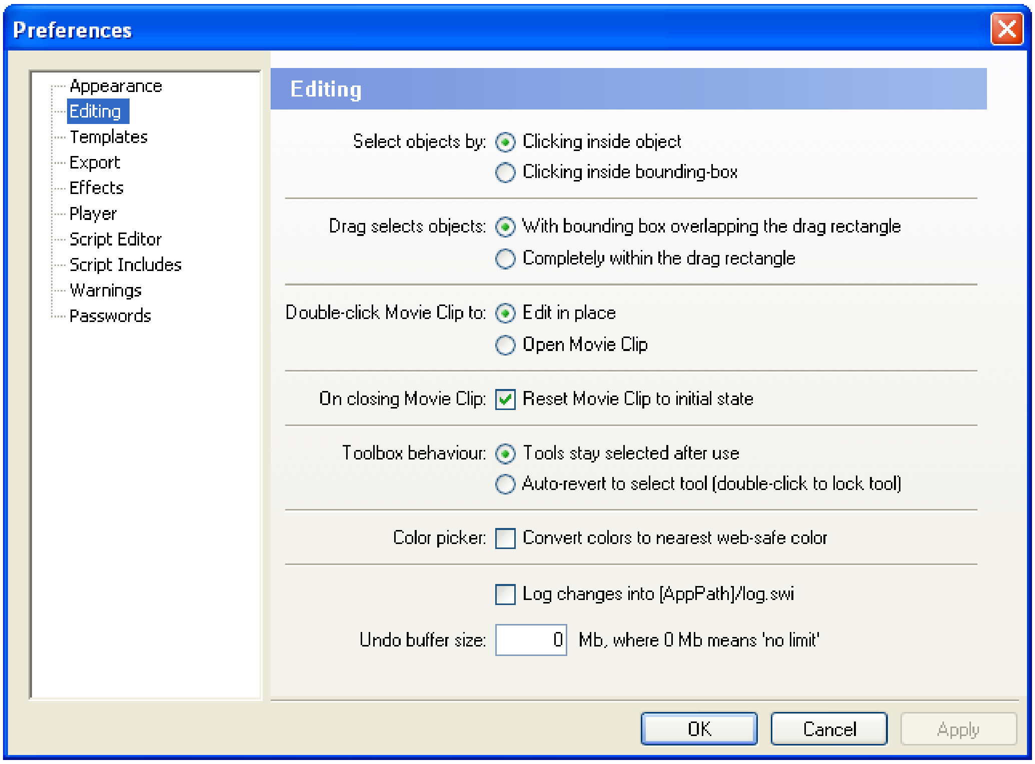
Thumbnails: Show Formatting: show thumbnails and text fonts and colors in panels

Shows a thumbnail of all images in the Outline Panel and the color and font of Text Objects in the Outline, Timeline and Text Panels.

Show startup menu

Enables/disables display of the Startup menu.

6.2.7.1.2 Editing



Select objects by: ● Clicking inside object
○ Clicking inside bounding-box

Drag selects objects: ● With bounding box overlapping the drag rectangle
○ Completely within the drag rectangle

Double-click Movie Clip to: ● Edit in place
○ Open Movie Clip

On closing Movie Clip: ☑ Reset Movie Clip to initial state

Toolbox behaviour: ● Tools stay selected after use
○ Auto-revert to select tool (double-click to lock tool)

Color picker: □ Convert colors to nearest web-safe color

□ Log changes into [AppPath]/log.swi

Undo buffer size: 0 Mb, where 0 Mb means 'no limit'

Select Objects By

- **Clicking Inside Object**
  An Object can only be selected by clicking inside the Object itself.

- **Clicking Inside Bounding-box**
  An Object can be selected by clicking anywhere inside its bounding-box. The bounding-box of an Object is a rectangle that completely encloses the Object. Selecting this option makes selecting small Text Objects make easier.

Drag Selects Objects

- **With Bounding-Box Overlapping the Drag Rectangle**
  When selecting a Drag Rectangle in the 'Layout' Panel an Object will be selected if any part of the Object's bounding-box is inside the Drag Rectangle.

- **Completely Within the Drag Rectangle**
  When selecting a Drag Rectangle in the 'Layout' Panel an Object will only be selected if the Object is completely inside the Drag Rectangle.

Double-Click Movie Clip To

- **Edit in Place**
  Double-clicking a Movie Clip will open it for editing in place. Movie Clips opened to edit in place are shown with a border around them.

- **Open Movie Clip**
  Double-clicking a Movie Clip will open it for editing.

On Closing Movie Clip: Reset Movie Clip to Initial State
Resets Movie Clips to Frame 1 after closing in Frame Preview Mode.

Toolbox Behaviour

- **Tools stay selected after use**
  Each time a tool is selected, it will remain active until another tool in the Toolbox is chosen.

- **Auto-revert to select tool (double-click to lock tool)**
  When a tool is used once, it will automatically re-activate the Selection tool. Double-clicking on any tool in the Toolbox will lock that tool until a different tool is chosen.

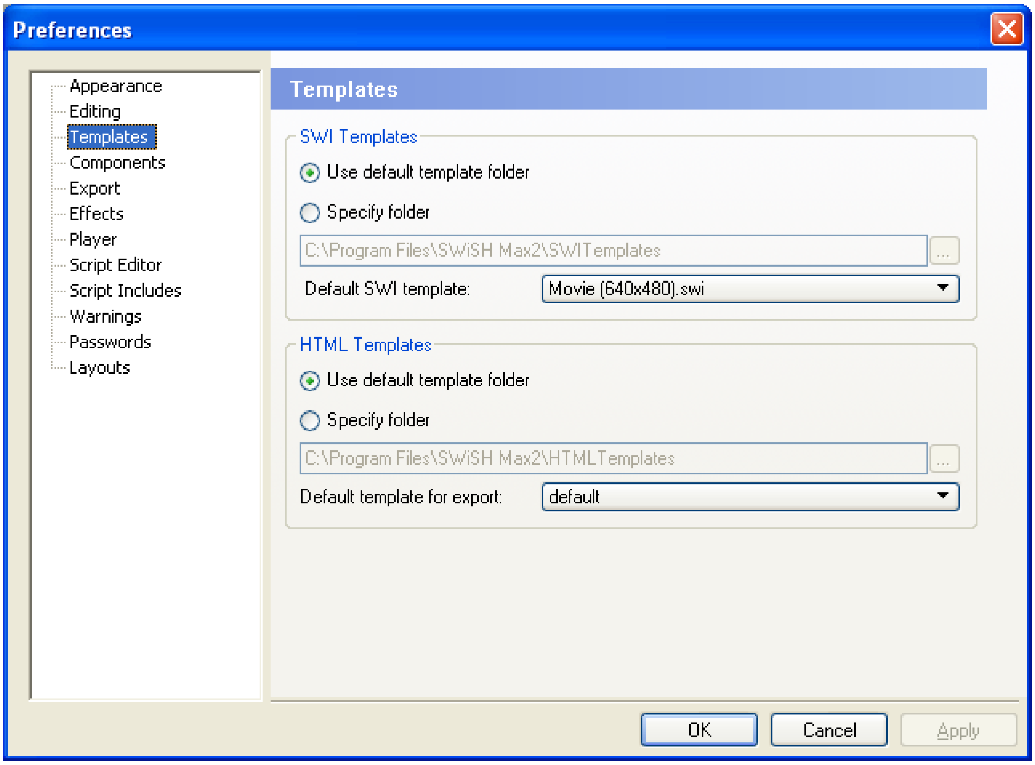
Color Picker: Convert Colors to Nearest Web-Safe Color
Converts the selected color to the nearest websafe color. This may be a color selected from the Color Picker, the screen or the Windows Color picker.

Debugging Options - Log changes into [AppPath]/log.swi
Creates a duplicate of the file and keeps track of changes. This can decrease the performance of SWiSH Max.

Undo buffer size
Sets the maximum size of the buffer used for the Edit | Undo and Edit | Redo options.

6.2.7.1.3 Templates

This dialog allows default templates and template folders to be defined. Templates are described more fully in the section Working with Templates.



Templates

SWI Templates

● Use default template folder

○ Specify folder

C:\Program Files\SWiSH Max2\SWITemplates

Default SWI template: Movie (640x480).swi

HTML Templates

● Use default template folder

○ Specify folder

C:\Program Files\SWiSH Max2\HTMLTemplates

Default template for export: default

SWI Templates
Sets the source path for the SWI Templates.

Use default template folder
Default option to use the installed location, typically 'c:\Program Files\SWiSHmax2\SWITemplates'.

Specify folder
Select a particular source folder path.

Default SWI template
Sets the name of the default template saved when File | Save as Default Template is selected.

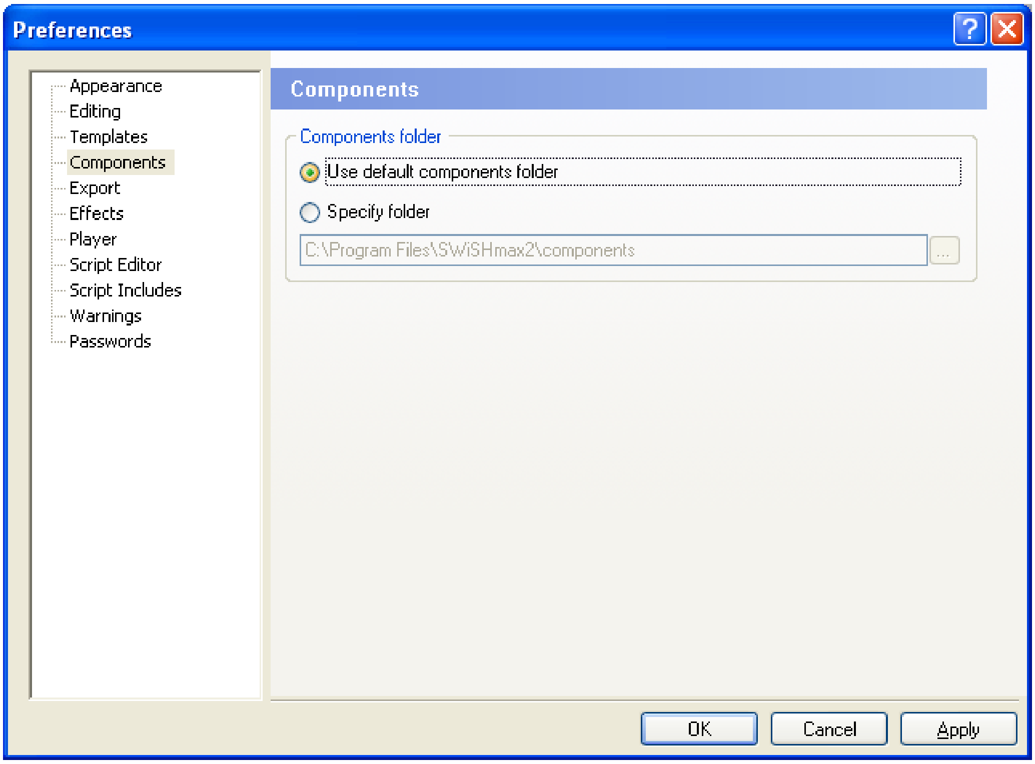
HTML Template
Sets the target files and path for the HTML Templates. Also see File | Export Settings | HTML.

Use default template folder
Default option to use the installed location, typically 'c:\Program Files\SWiSHmax2\HTMLTemplates'.

Specify folder
Select a particular target folder path.

Default template for export
Sets the name of the default template used when File | Export | HTML + SWF ... is selected.

6.2.7.1.4 Components



Components folder

● Use default components folder

○ Specify folder

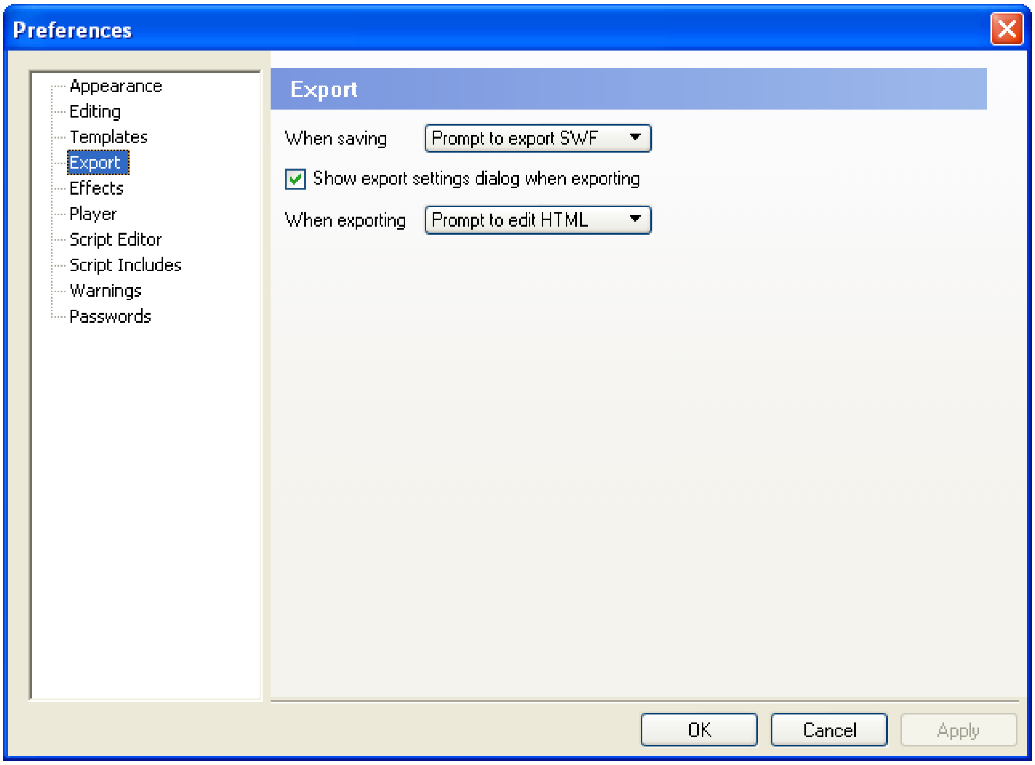
C:\Program Files\SWiSHmax2\components

Components folder
Sets the source path for the Components folder.

Use default template folder
Default option to use the installed location, typically 'c:\Program Files\SWiSHmax2\components.

Specify folder
Select a particular source folder path.

6.2.7.1.5 Export



When saving: Prompt to export SWF

☑ Show export settings dialog when exporting

When exporting: Prompt to edit HTML

Export options

When saving files (SWI) the available options are:

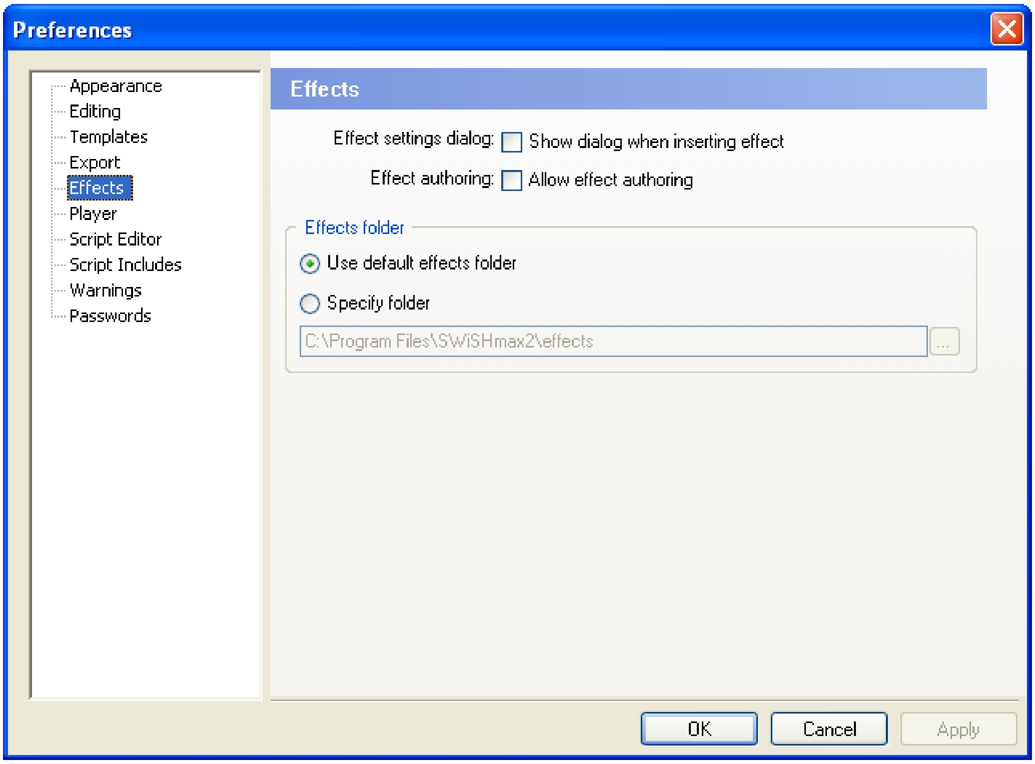
- **Always Export SWF**: always exports (without a prompt) the file SWF upon save
- **Prompts to Export SWF**: asks for confirmation prior to exporting SWF
- **Do not export SWF**: never ask and do not export SWF

Show export settings dialog when exporting
Enable/disable display of the Export Preferences dialog shown when exporting.

When exporting files (SWF), SWiSH Max can open the exported HTML file withe PC's associated HTML editor. The available options are:

- **Always edit HTML**: always open HTML file for editing (without a prompt) when exporting
- **Prompt to edit HTML**: asks for confirmation before opening HTML file
- **Do not edit HTML**: never ask and do not open HTML file

6.2.7.1.6 Effects



Effects

Effect settings dialog: □ Show dialog when inserting effect

Effect authoring: □ Allow effect authoring

Effects folder

● Use default effects folder

○ Specify folder

C:\Program Files\SWiSHmax2\effects

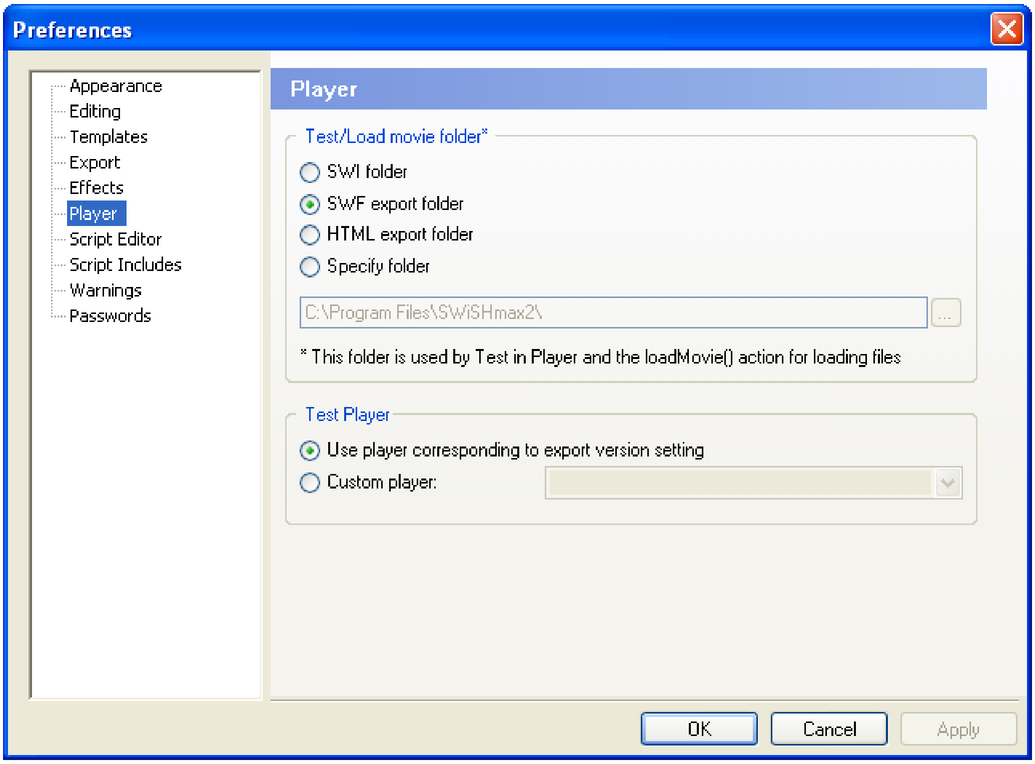
Effect settings dialog: show dialog when inserting effect
Automatically opens the Effects Settings Dialog box when new Effects are inserted into the Movie.

Effect authoring: allow effect authoring
Makes the Author Mode option available in the Effects dialog box. See Authoring Effects for more details.

Effects folder
Sets the location where SWiSH Max looks for effect files (.sfx):

- **Use default effects folder**: Uses SWiSH Max's default Effects folder for Authored Effects.
- **Specify folder**: Select a particular source folder path for Authored Effects.

6.2.7.1.7 Player



Player

Test/Load movie folder*

○ SWI folder
● SWF export folder
○ HTML export folder
○ Specify folder

C:\Program Files\SWiSHmax2\

* This folder is used by Test in Player and the loadMovie() action for loading files

Test Player

● Use player corresponding to export version setting
○ Custom player:

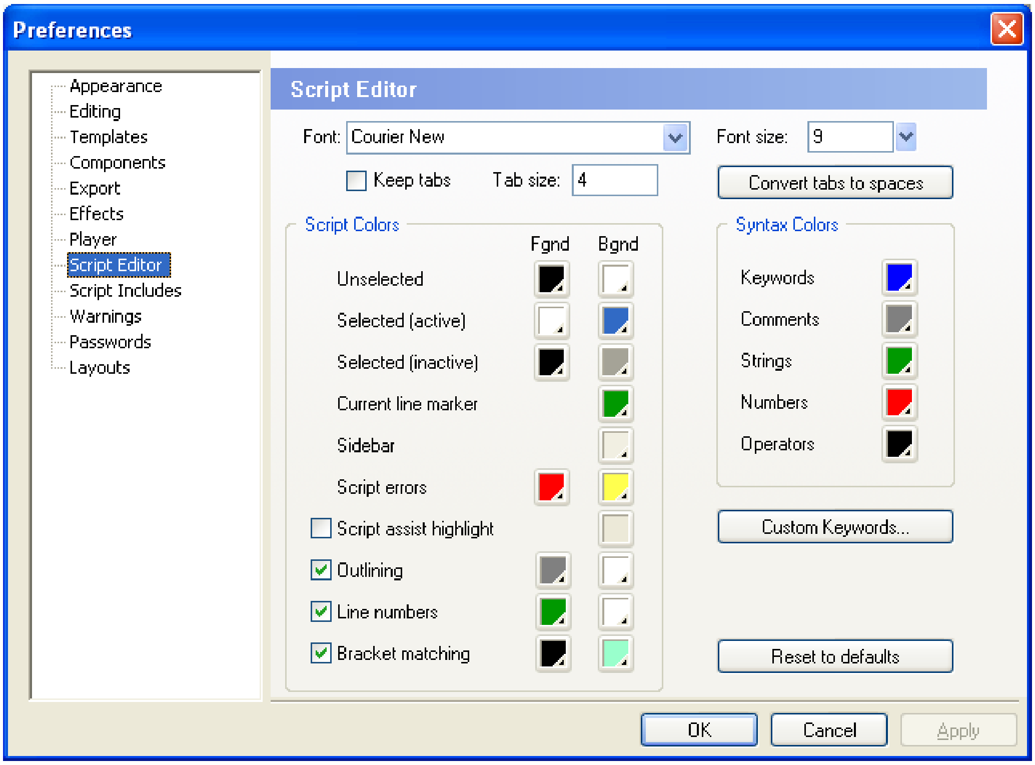
Test/Load Movie folder
Sets the folder to search for .swf files when a Load Movie Action is executed while previewing in SWiSH Max or testing in the player or browser. The following options are available:

- **SWI folder**: the folder where a .swi was last loaded or saved
- **SWF export folder**: the folder where a .swf was last exported
- **HTML export folder**: the folder where a .htm was last exported
- **Specify folder**: a custom folder

Test Player
These options setup the standalone Flash player used for the File | Test | SWF in Flash Player. The following options are available:

- **Use player corresponding to export version setting**: SWiSH Max automatically selects the appropriate player based on the SWF version selected in File | Export Settings | SWF.
- **Custom Player**: Select a particular Flash player version. SWiSH Max offers a range of players denoted by the file name "SWiSHpla_n.exe" where "n" denotes the Flash player version. e.g. "SWiSHpla_9.exe" is Flash player version 9.

6.2.7.1.8 Script Editor



Script Editor

Font: Courier New

Font size: 9

□ Keep tabs

Tab size: 4

Script Colors

	Fgnd	Bgnd
Unselected	■	□
Selected (active)	□	🟦
Selected (inactive)	■	▨
Current line marker		🟩
Sidebar		□
Script errors	🟥	🟨
□ Script assist highlight		□
☑ Outlining	▨	□
☑ Line numbers	🟩	□
☑ Bracket matching	■	🟩

Syntax Colors

Keywords	🟦
Comments	▨
Strings	🟩
Numbers	🟥
Operators	■

Font
Sets the font used in the Script Editor.

Font size
Defines the font size used in the Script Editor.

Keep tabs
When checked, the tab key inserts a tab character. Otherwise, the tab key inserts the number of spaces specified by the 'Tab size' setting.

Tab size
Defines the tab size (in character spaces) used in the Script Editor.

Convert tabs to spaces
Converts existing tabs to spaces according to the currently defined Tab size.

Script Colors
Sets the foreground (fgnd) and background (bgnd) colors for text and editing functions in the Script Editor.

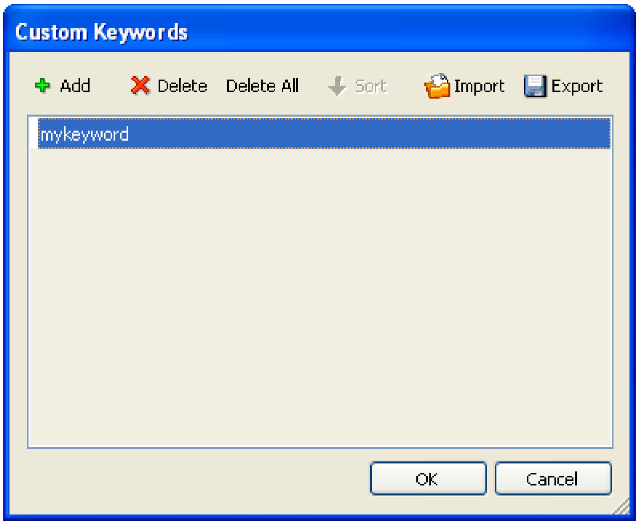
- Unselected
- Selected (active)
- Selected (inactive)
- Current line marker
- Sidebar
- Script errors
- Script Assist highlight
- Outlining

- Line numbers
- Bracket matching

Syntax Colors
Sets the syntax coloring for text in the Script Editor.

- Keywords
- Comments
- Strings
- Numbers
- Operators

Custom Keywords
The Custom Keyword dialog is used to define keywords used in Script Editor which are highlighted or color coded (see 'syntax colors' above).

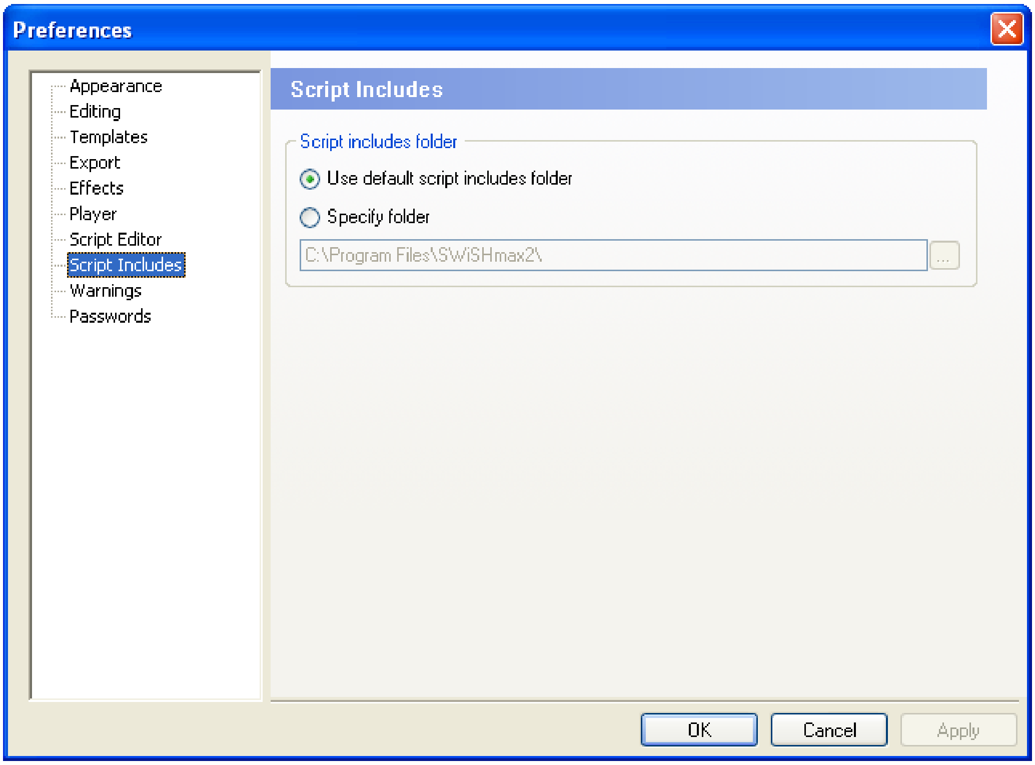


mykeyword

Import and Export can be used to load or save keywords from another installation of SWiSH Max

Reset to defaults
Resets all script editor options as per the original SWiSH Max installation.

6.2.7.1.9 Script Includes



Preferences

Script Includes

Script includes folder

● Use default script includes folder

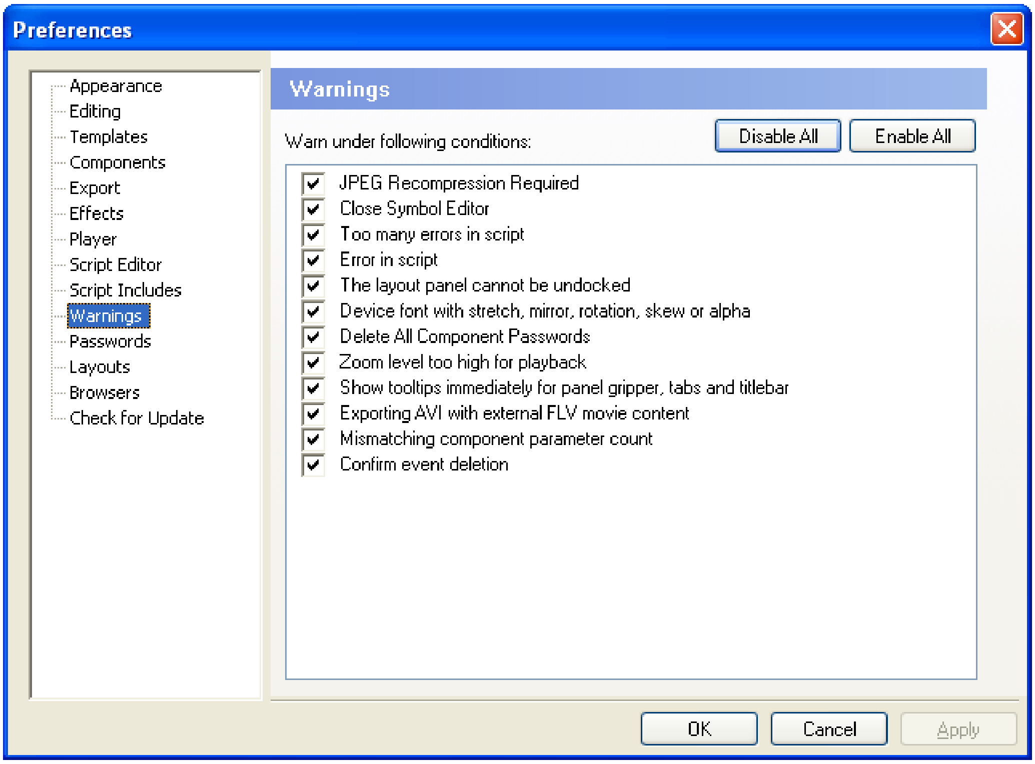
○ Specify folder

C:\Program Files\SWiSHmax2\

Script includes folder
SWiSH Max looks in this folder for files specified by 'include' (in Script). Options are:

- **Use default script includes folder**: Uses SWiSH Max's default script includes folder
- **Specify folder:** Select a particular source folder path for the include files

6.2.7.1.10 Warnings



Warn under following conditions:

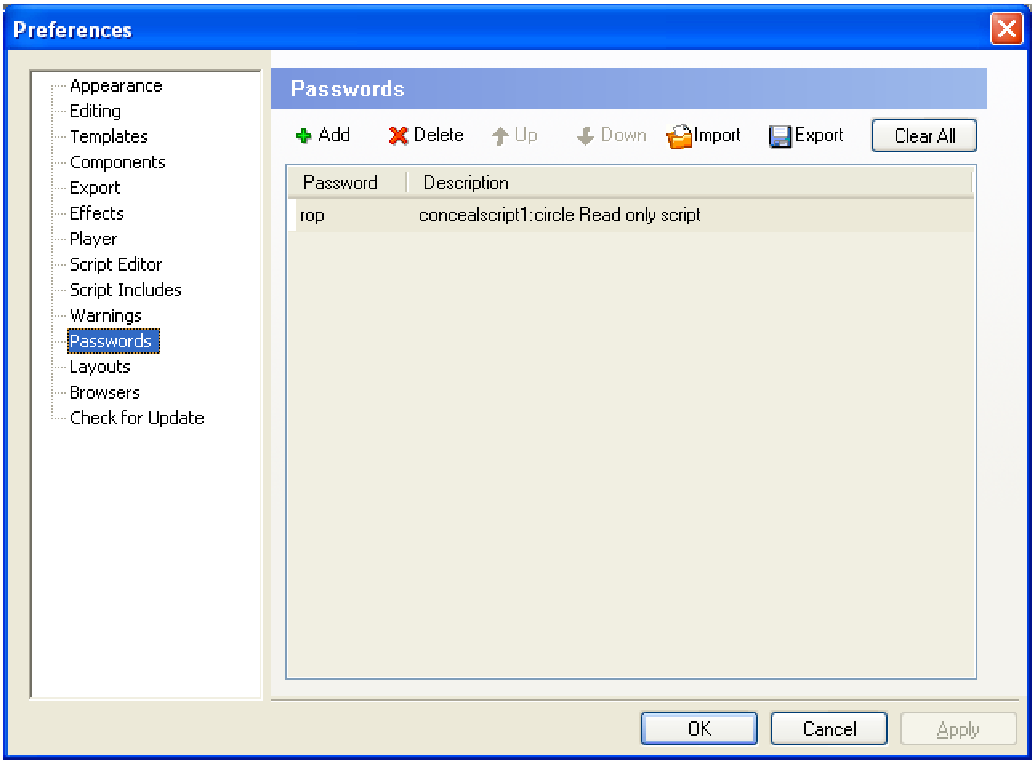
☑ JPEG Recompression Required
☑ Close Symbol Editor
☑ Too many errors in script
☑ Error in script
☑ The layout panel cannot be undocked
☑ Device font with stretch, mirror, rotation, skew or alpha
☑ Delete All Component Passwords
☑ Zoom level too high for playback
☑ Show tooltips immediately for panel gripper, tabs and titlebar
☑ Exporting AVI with external FLV movie content
☑ Mismatching component parameter count
☑ Confirm event deletion

Enables warning dialogs which appear when using SWiSH Max:

- JPEG Compression required
- Close symbol editor
- Too many errors in script
- Error in script
- The layout panel cannot be undocked
- Device font with stretch, mirror, rotation, skew or alpha
- Delete All Component Passwords
- Zoom level too high for playback
- Show tooltips immediately for panel gripper, tabs and titlebar
- Exporting AVI with external FLV movie count ***(SWiSH Max only)***
- Confirm event deletion

Disable All / Enable All
Enables or disables all warnings.

6.2.7.1.11 Passwords



Password	Description
rop	concealscript1:circle Read only script

When a password is associated with an object attribute, it will appear in this table. While the password exists in the table, the associated attribute can be altered.

The description field is initialized with the .swi name, the object name and the name of the attribute that the password is associated with. This description field can be edited as required.

Add / Delete / Up / Down
The buttons at the top of this dialog allow individual passwords to be added to, deleted from or moved within the list.

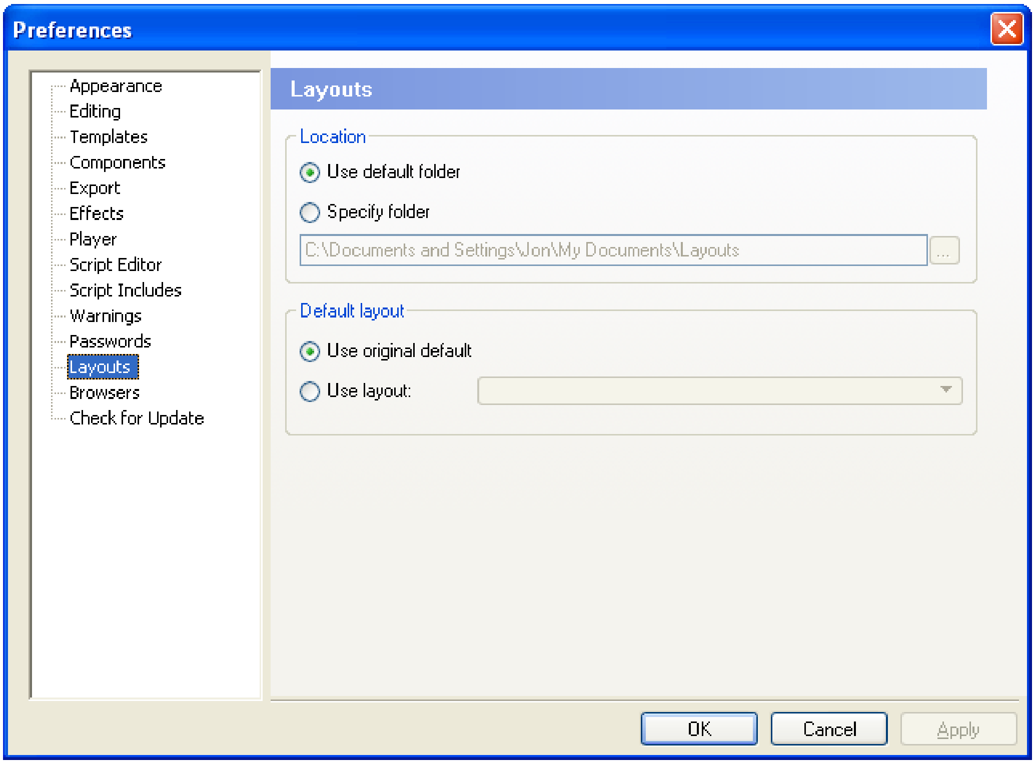
Import / Export
It is possible to import and export the password list as a .csv file via the save and load commands. This facility is useful when moving the development environment to another machine.

Tip: When transferring a copy of SWiSH Max to a new or different machine, backup passwords to a .csv file as described above and restore to the new machine when SWiSH Max is installed.

Clear All
Clears all passwords from the list.

Tip: Clear all passwords to examine how password protected components behave on a system that does not know the passwords. Backup all passwords using the Import button described above so they can be re-loaded at a later stage.

6.2.7.1.12 Layouts



Location

● Use default folder

○ Specify folder

C:\Documents and Settings\Jon\My Documents\Layouts

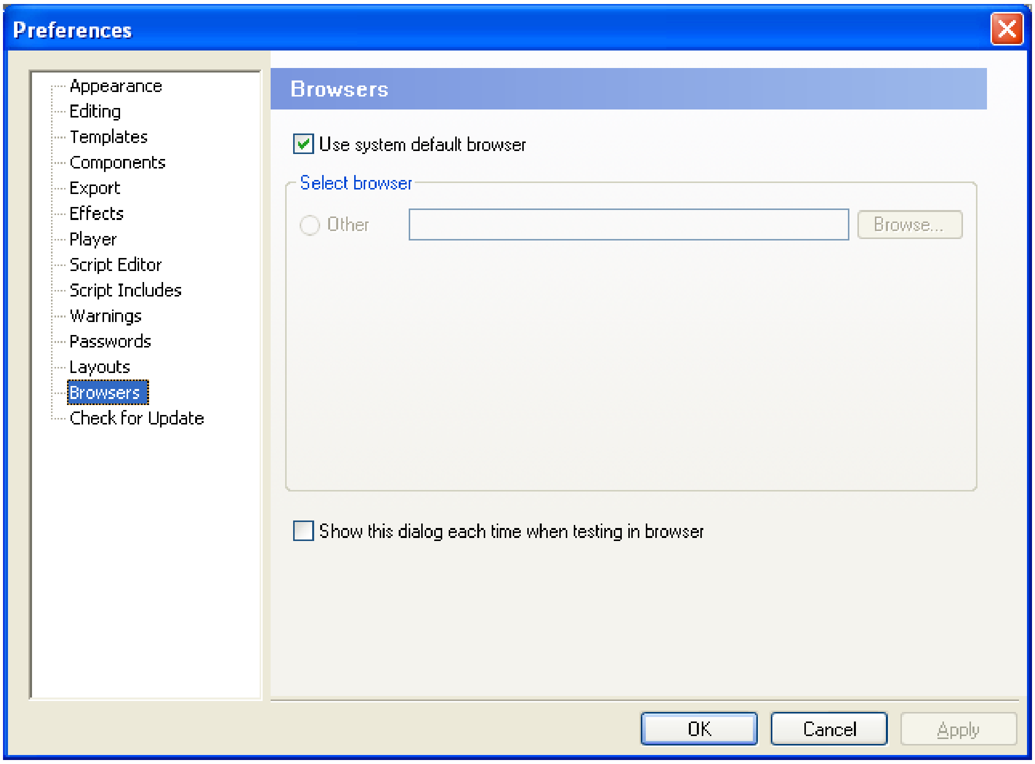
Default layout

● Use original default

○ Use layout:

Allows selection of the default layout configuration and the location of the layouts folder.

6.2.7.1.13 Browser



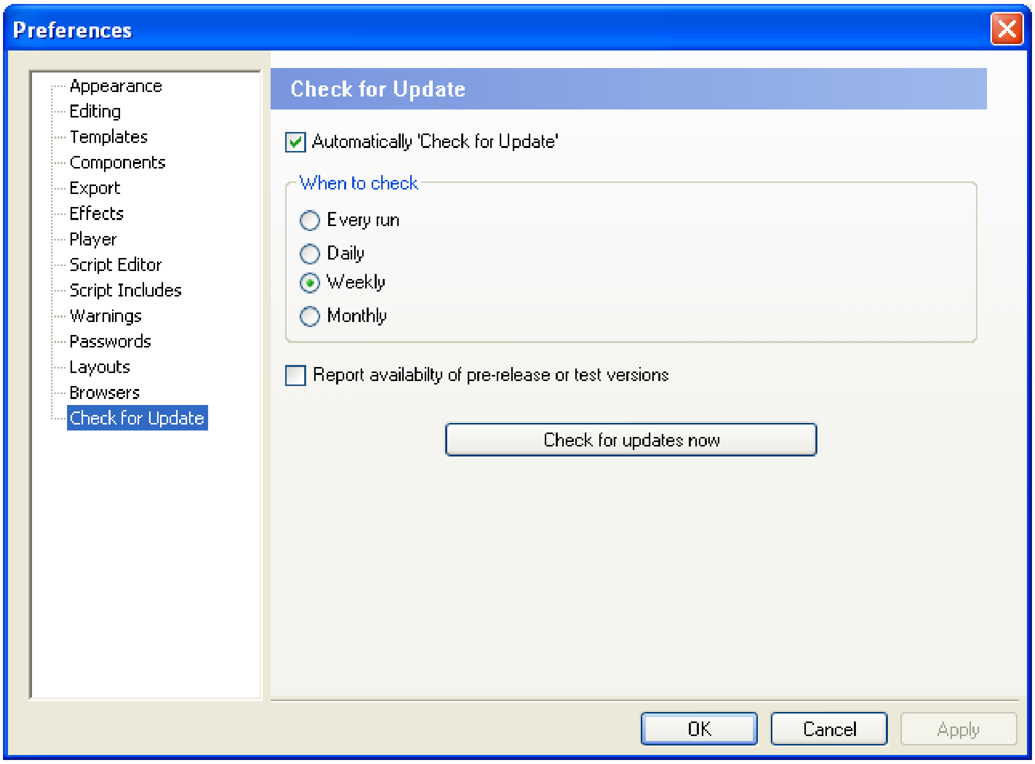
☑ Use system default browser

○ Other

□ Show this dialog each time when testing in browser

This dialog allows the user to select the default browser to be used with the Test / Test in Browser option.

6.2.7.1.14 Check for Update



Check for Update

☑ Automatically 'Check for Update'

When to check

○ Every run
○ Daily
● Weekly
○ Monthly

□ Report availabilty of pre-release or test versions

Check for Update
This preference enables a prompt for the user to check for updates. This is only a reminder to the user and SWiSH Max does not communicate directly with any remote computers. If accept by the user the process will launch the default and goes to the SWiSHzone.com web site at http://www.swishzone.com to check if there is an updated build of SWiSH Max.

When to check
Sets when to prompt for check for update.

- **Every run:** User will be prompted to check every time SWiSH Max is run.
- **Daily:** User will be prompted the first time SWiSH Max is run each day.
- **Weekly:** User will be prompted the first time SWiSH Max is run each week.
- **Monthly: :** User will be prompted the first time SWiSH Max is run each month.

Report availability of pre-release or test versions
The user is offered information about the availability of pre-release or test versions of SWiSH Max.

Check for updates now
Launches the browser and goes to the SWiSHzone.com web site at http://www.swishzone.com to check if there is an updated build of SWiSH Max.

6.2.7.2 FTP Connection...

Tools | FTP Connection ...

SWiSH Max includes an FTP tools, SWiSH FTP. Files exported using File | Export | SWF... and File | Export | HTML + SWF will queue for upload. See Using FTP for more details.

6.2.7.2.1 FTP Transfer

SWiSH Max can upload files to your webserver using FTP (File Transfer Protocol) via the Main FTP Dialog
.

Before using the FTP dialog, you should know or have access to the following information:

Host name	The name or Intenet address of your FTP server. eg. users.bigpond.net.au
Port	This is normally port 21 unless your ISP (Internet Service Provider) has chosen to use a different port.
Server upload protocol	Currently this dialog only supports FTP. If your ISP requires you to use a different protocol for uploading (eg SFTP) we recommend that you use a 3rd party application to upload your files. Filezilla is an excellent free application that is available from http://filezilla.sourceforge.net/ that will handle other protocols.
Username and password	This is provided by your ISP. If you are using "Anonymous" access then the user name could be either: Anonymous or Ftp. For Anonymous access your password is normally blank although some sites request that you use your email address.
Public URL	The URL from which your uploaded files are available to the public. See FAQ for more information.

If you do not know any of the above information, please contact your ISP.

Once you know the above information, enter it into the Main FTP Dialog then press the Upload button to start the upload process.

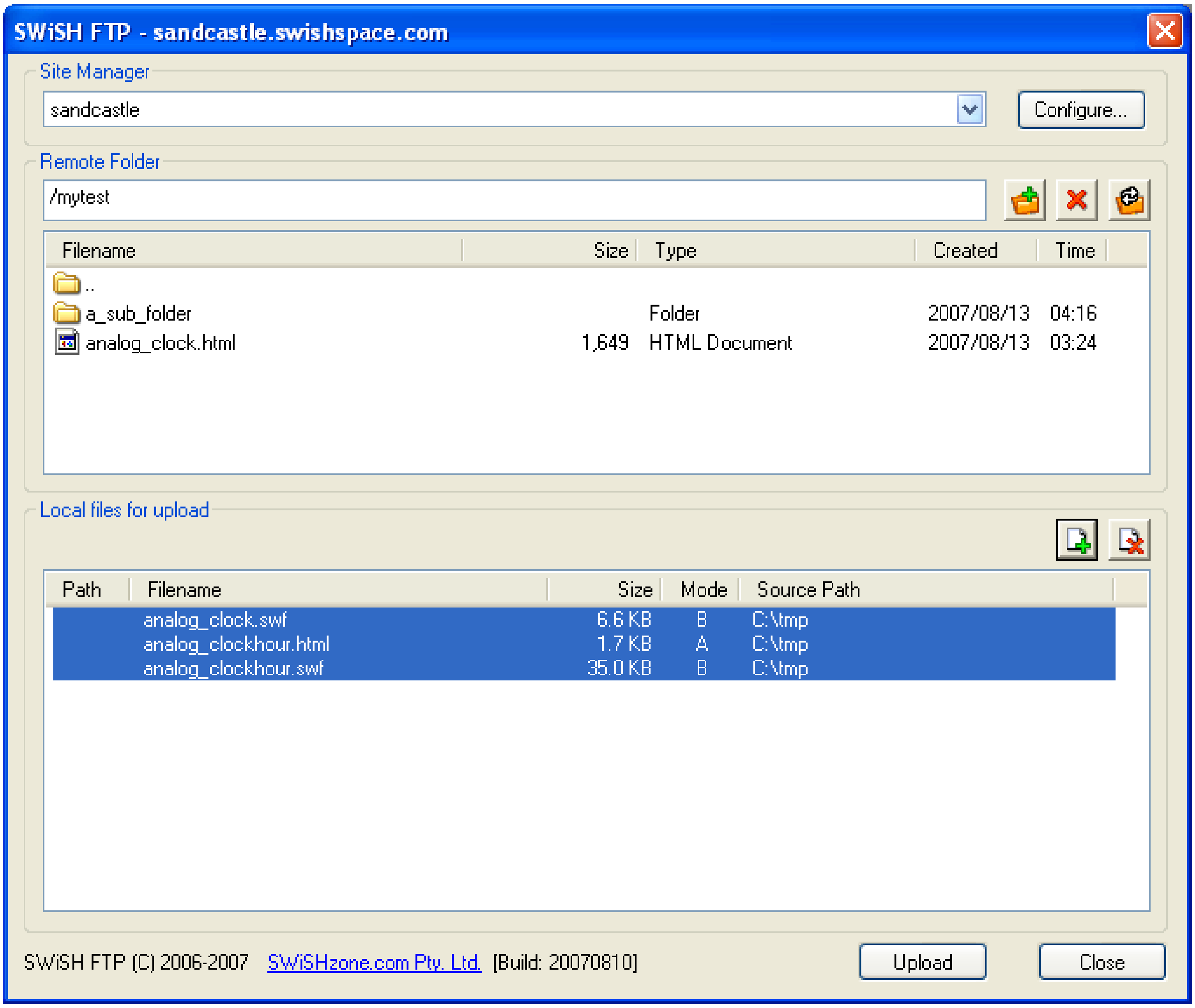
6.2.7.2.1.1 Main FTP Dialog

The dialog consists of 3 main sections. These are:
1. The **Site Manager section**.Site specific configuration data can be loaded via the list box. New sites can be configured via the **Configure...** button which will open the Site Configuration Dialog.
2. The Remote Folder section lists the files in the currently selected folder on the remote site.
3. The Upload File List section. This shows the files currently selected for upload. This section is normally loaded with files as part of the previous publish process.

Below the Upload File List section there are the **Upload** and **Close** buttons as well as copyright and build information. Please quote the build number if you need to contact Support about this dialog. The SWiSHzone.com website can be contacted by clicking on the Blue Hyperlink (an internet connection is required).

If the information contained in the Site Details section is valid, pressing the **Upload** button will cause the Upload File Progress dialog to be displayed while the files are being uploaded.

Pressing the **Close** or **x** buttons will close the main dialog window. The currently selected site will be saved. The site details for the current site will be automatically loaded if the dialog is re-displayed.

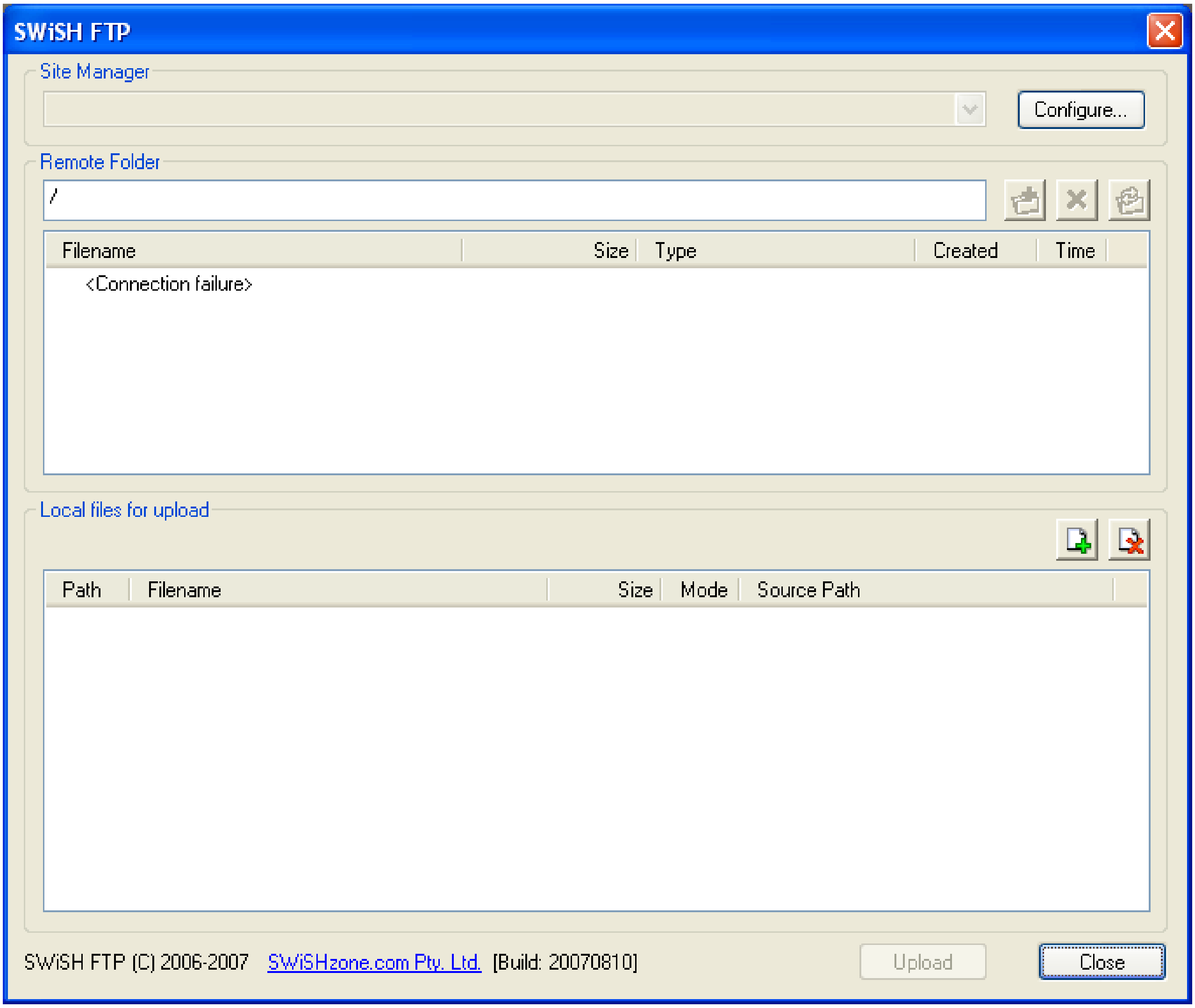

SWiSH FTP - sandcastle.swishspace.com
Site Manager
sandcastle
Configure...
Remote Folder
/mytest

Filename	Size	Type	Created	Time
..				
a_sub_folder		Folder	2007/08/13	04:16
analog_clock.html	1,649	HTML Document	2007/08/13	03:24

Path	Filename	Size	Mode	Source Path
	analog_clock.swf	6.6 KB	B	C:\tmp
	analog_clockhour.html	1.7 KB	A	C:\tmp
	analog_clockhour.swf	35.0 KB	B	C:\tmp

SWiSH FTP (C) 2006-2007 SWiSHzone.com Pty. Ltd. [Build: 20070810]
Upload
Close

When used for the first time, the Site Manager section and Remote Folder Details section may be empty.



If this is the case, press then use the **Configure...** button to access the Site Configuration Dialog that will allow you to define the connection details of your remote site(s). Simply enter the appropriate information in the Site Details section then Save the site details using the Save or Save As buttons.

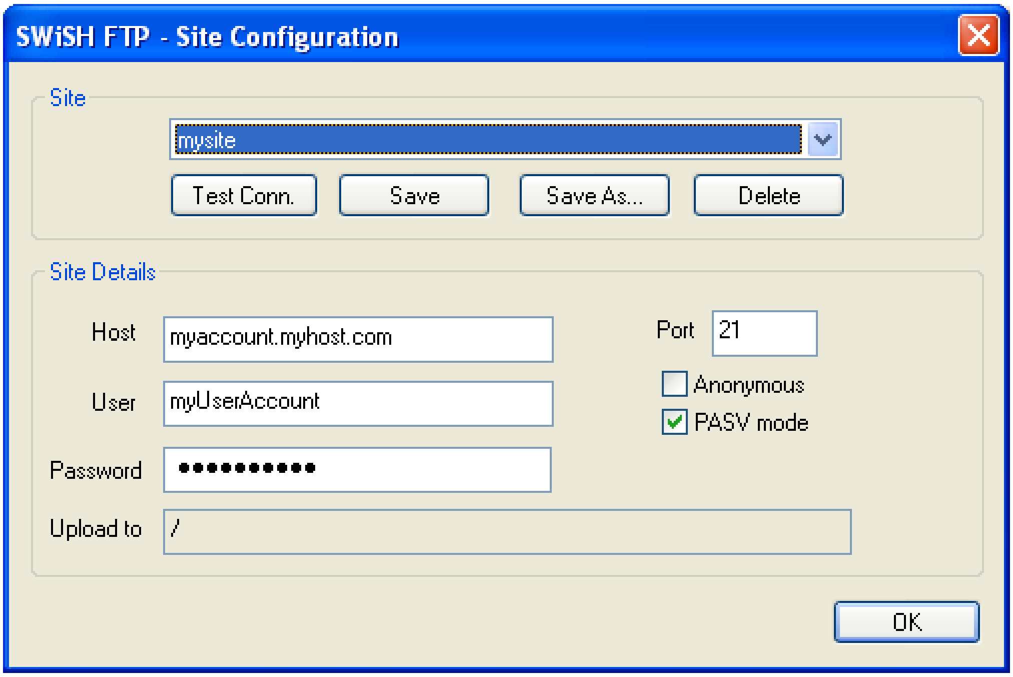
Once information has been entered into the Site Details section and saved, the FTP connection can be tested using the **Test Conn.** button.
If files are shown in the Upload File List, then pressing the **Upload** button will initiate the file transfer.

When used for the first time, it is possible that you will receive firewall messages. If this occurs please see the Firewall Management section in the Frequently Asked Questions area.

During the file transfer the Upload File Progress dialog is displayed to show the upload progress.

On completion of the upload process, the Upload File List is cleared and the Main FTP Dialog window can be closed.

This dialog consists of two main sections:
1. The Site section which allows selection and saving of site details under a named configuration and
2. The Site Details section. Site specific configuration can be entered and edited using this section.



SWiSH FTP - Site Configuration

Site

mysite

Site Details

Host: myaccount.myhost.com

Port: 21

User: myUserAccount

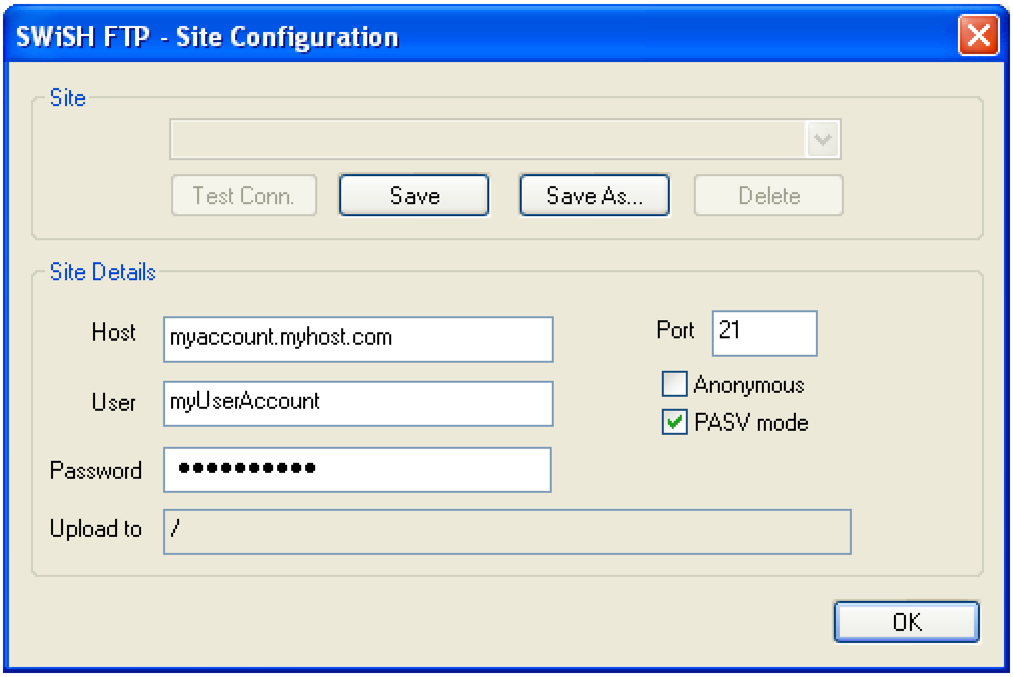
□ Anonymous

☑ PASV mode

Password: ●●●●●●●●●●●

Upload to: /

If no sites have previously been configured then the dialog will look like this:



SWiSH FTP - Site Configuration

Site

Site Details

Host: myaccount.myhost.com

Port: 21

User: myUserAccount

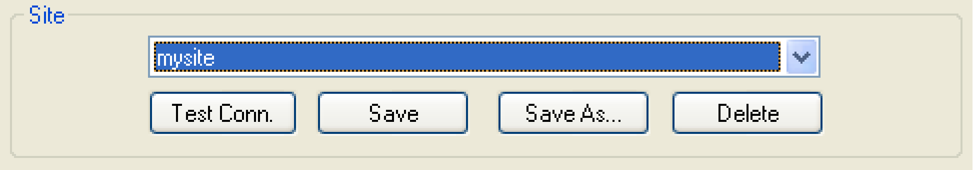
□ Anonymous

☑ PASV mode

Password: ●●●●●●●●●●●

Upload to: /

Use the **Save** or **Save As...** button to save your configuration under a specific name.



Site

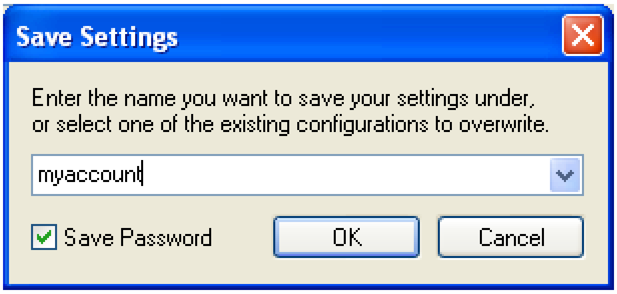
mysite

This section is used to manage configuration data from the Site Details section. Configuration data can be saved under a single item name using the **Save As...** button. Once saved, the data can be retrieved by selecting the appropriate name from the drop down list.

The configuration data of an existing site can be updated and re-saved using the **Save** button and the current configuration can be deleted using the **Delete** button.

The **Test Conn.** button means test connection. Pressing this button will cause the FTP connection to be verified based on the saved site information.
If the site information has not been saved, use the **Save** or **Save As...** buttons to save the site configuration under a site name.

When using the **Save As...** button the following dialog will be displayed:



Save Settings

Enter the name you want to save your settings under, or select one of the existing configurations to overwrite.

myaccount

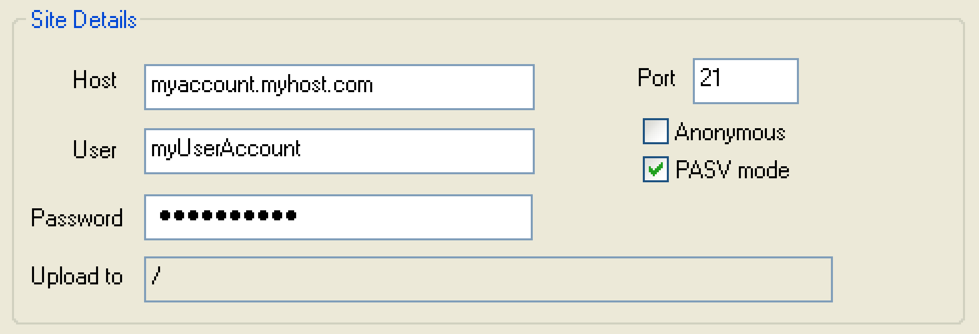
☑ Save Password

A new site name can be entered, or one of the existing names can be selected using the drop down list.
If you do not want to save the password from the Site Details section, uncheck the **Save Password** checkbox. Uncheck the **Save Password** checkbox if you are using login that is shared with other users.

Once you have selected or entered an appropriate site name, press the **OK** button to save the details.

The Save dialog is similar except that it is not possible to select a different account.

Saved details are written to the Windows Registry in the HKEY_CURRENT_USER section.



Site Details

Host myaccount.myhost.com

User myUserAccount

Password •••••••••••

Upload to /

Port 21

□ Anonymous

☑ PASV mode

Use this section to enter your FTP login information. This information is normally provided by your ISP. Once you have entered this information you can verify the connection using the Test Conn. button in the Site section.
You can also save it for later use using the **Save** or **Save As...** buttons in the Site section.

Port

This will normally be 21. Use a different port number only if you are directed to do so buy your ISP.

Host

This is the name or IP address of your FTP host. Files are uploaded to this site.

User

This is the user name that you supply to your FTP host. If you are using an anonymous connection you can either check the Anonymous checkbox or use the user name "Anonymous". In some cases an Anonymous site may require you to use the User name "ftp".

Password

This is the password that is associated with your user name. If you are using an anonymous connection you can check the Anonymous checkbox to remove the need for this field.
Note: Some websites will request you to enter your email address in this field. If this is the case, uncheck the Anonymous checkbox and enter your email address in this field and either Anonymous or ftp into the User field (as directed by the website).

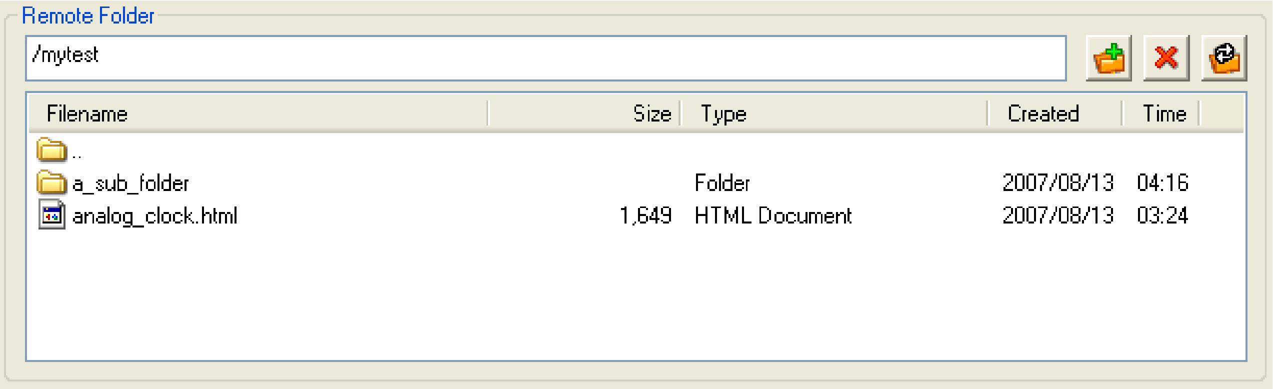
Upload to

This is the default upload directory. This directory is selected via the Remote Folder section of the Main FTP Dialog.

PASV mode

Use this mode if it is supported by your ISP (most ISP's support this mode). Use of this mode will simplify potential firewall issues that would normally prevent a successful file transfer.

This section allows the currently selected upload folder to be viewed and altered.



Remote Folder

/mytest

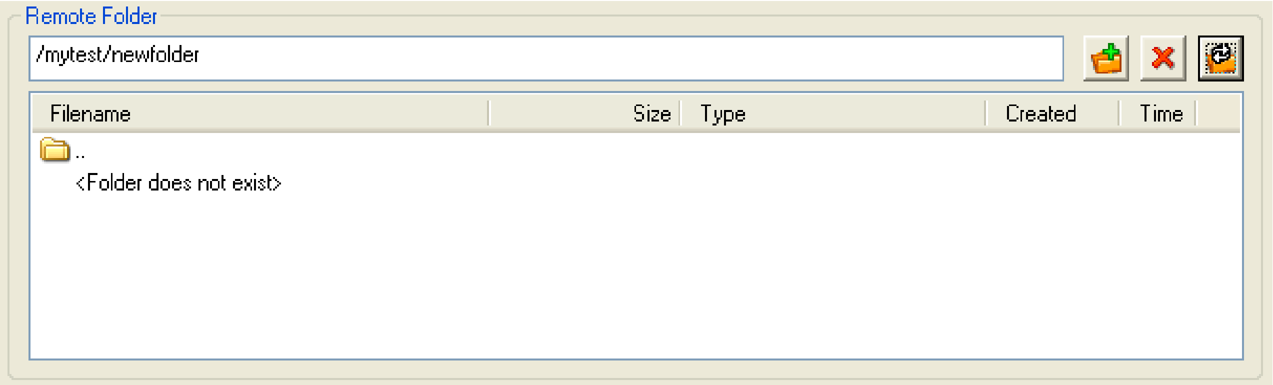
Filename	Size	Type	Created	Time
..				
a_sub_folder		Folder	2007/08/13	04:16
analog_clock.html	1,649	HTML Document	2007/08/13	03:24

The contents of the current folder are shown in the list panel. The following items are displayed:

Filename The name of the file. Folders are shown with a folder icon.
Size The size of the file in bytes.
Type The type of file based on the extension (if known).
Created The Year, Month and Date that the file was created or uploaded.
Format: YYYY/MM/DD
Time The Time that the file was uploaded to the server.
Format: 24 Hour, HH:MM
Note: The time will be based on the servers local time.
Note: Depending on the server, files older than one year may be shown with a time of 00:00

Navigation to other folders is possible by either:
1. Double clicking one of the available folders (**..** or **a_sub_folder** in the example above) or

2. By entering the name directly in the edit box and then pressing the **Refresh** button. If the folder does not exist then a message will be displayed as shown below:

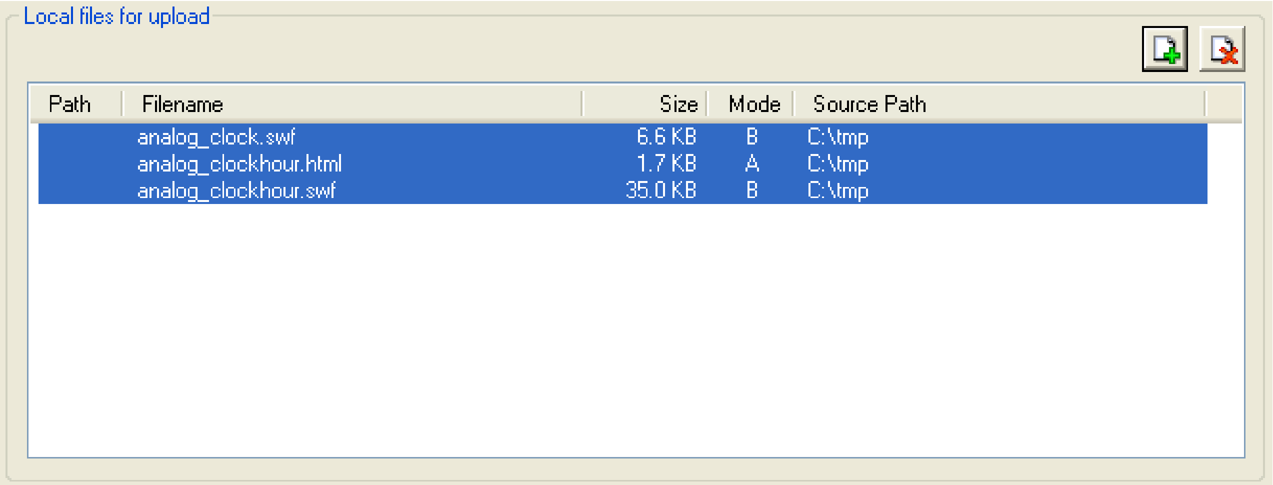


/mytest/newfolder

..
<Folder does not exist>

Files and folders can be deleted by selecting one or more items then pressing the **Delete** ✖ button.
Note: only empty folders can be deleted.

New sub folders can be created using the **New Folder** button.



Path	Filename	Size	Mode	Source Path
	analog_clock.swf	6.6 KB	B	C:\tmp
	analog_clockhour.html	1.7 KB	A	C:\tmp
	analog_clockhour.swf	35.0 KB	B	C:\tmp

This section contains a list of files that will be uploaded.

Files can be added to the list using the **Add File** button.
Files can be removed from the list by selecting the file with a single click then pressing the **Remove File** button.

The default transfer mode ASCII (A) or Binary (B) can be toggled by double clicking on the file.

The file list contains the following information:

Path
This section could be blank. Otherwise it contains the sub folder that the corresponding file will be uploaded to.
eg. If the **Upload to** folder in the Remote Folder section was ***/junk*** and **Path** contained ***foo***, then the corresponding file will be uploaded to ***/junk/foo***

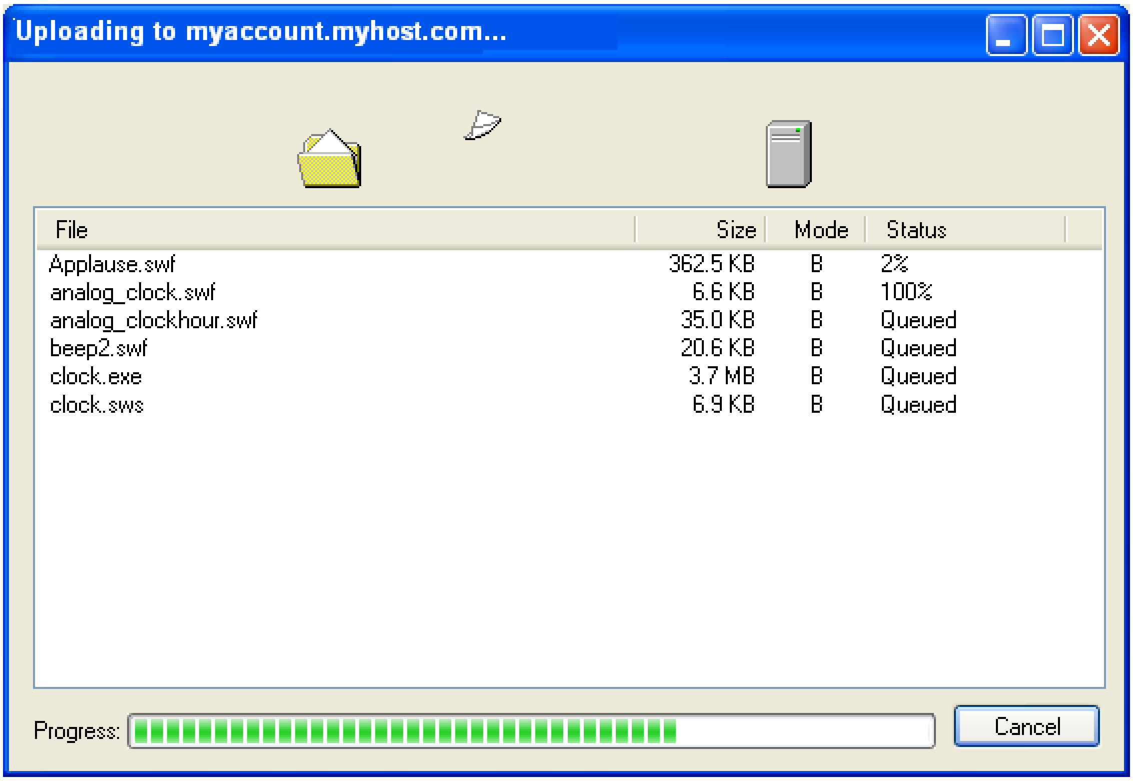
File
The name of the file that will be uploaded.

Size
The size of the file.

Mode
FTP allows transfer in Binary or ASCII modes.
In Binary mode, the file is transferred without translation.
In ASCII mode, the file is assumed to be an ASCII text file. Formatting characters such as Newline are mapped into the appropriate characters for the destination system.

In general .swf and image files should be transferred as Binary.
.html, .xml, and .txt files should be transferred as ASCII.

Source Path
This is the folder that holds the file on the local system.


Uploading to myaccount.myhost.com...

File	Size	Mode	Status
Applause.swf	362.5 KB	B	2%
analog_clock.swf	6.6 KB	B	100%
analog_clockhour.swf	35.0 KB	B	Queued
beep2.swf	20.6 KB	B	Queued
clock.exe	3.7 MB	B	Queued
clock.sws	6.9 KB	B	Queued

When the upload process is in progress, this dialog displays the current stage and status of the upload(s). It is possible to stop the upload process by pressing the Cancel button.

The **Status** section displays the current upload status of each file. The possible values are:
A percentage complete indicating that the upload for the associated file is **Active**.
Queued indicating that the file will be uploaded but it has not yet started the upload process.
Error indicates that an error has occurred during the upload.

Files displayed in this dialog are sorted according to Status and File name. Active (in progress) transfers are listed first, then **Queued**

Files who's upload is complete are removed from the list.

File, **Size** and **Mode** have the same meanings as discussed in the Upload File List section.

6.2.7.2.1.2 FTP Frequently Asked Questions

Q. I am having problems getting the upload to work in Active Mode?
A. Try using PASV mode. Also see the section Firewall Management.

Q. What is Anonymous FTP?
A. See Anonymous FTP.

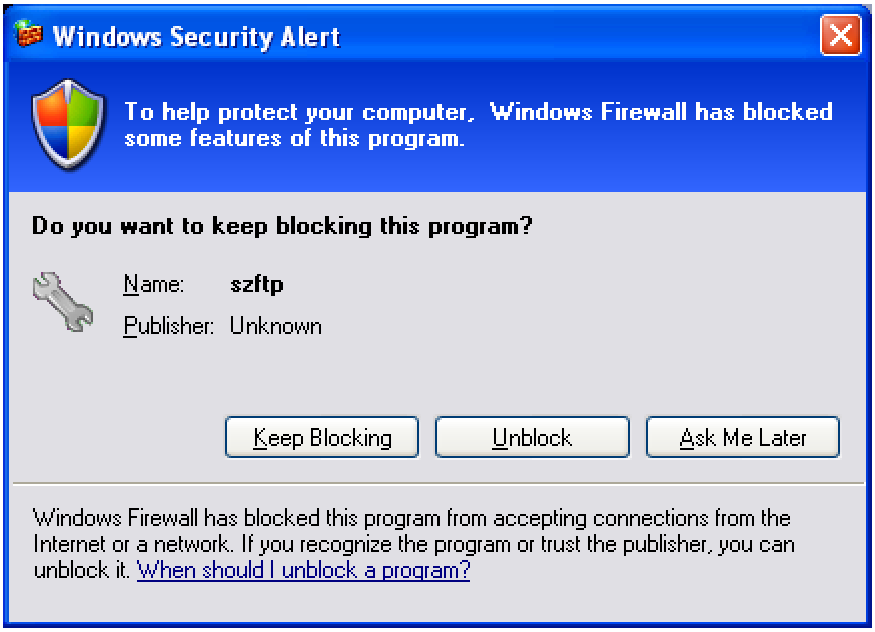
Q. What is file Transfer Mode?
A. See Data Format.

Q. I have uploaded myfile.swf to a folder /test123. How can I view my uploaded file?
A. This will depend on your ISP.
Note that as the file is a .swf file, you should use Binary mode to transfer the file.
If the FTP server is "**myIspFtpArea.myIsp.com**" then depending on your ISP server configuration, your uploaded files may appear at:
http://myIspFtpArea.myIsp.com/username/test123/myfile.swf or
http://myIspFtpArea.myIsp.com/~username/test123/myfile.swf or
http://myIspFtpArea.myIsp.com/test123/myfile.swf

Your files may also appear at a different location as determined by your ISP. Please refer to your ISP for further information.

Use of the PASV mode should prevent most firewall issues. If using PASV mode, the dialog shown below should not appear. It should not be necessary to create an exception for SWiSH Max in the Windows XP firewall.

If you use Actiive mode (PASV checkbox is not ticked) then you may receive the following message from the Windows XP firewall:



Windows Security Alert

To help protect your computer, Windows Firewall has blocked some features of this program.

Do you want to keep blocking this program?

Name: szftp

Publisher: Unknown

Windows Firewall has blocked this program from accepting connections from the Internet or a network. If you recognize the program or trust the publisher, you can unblock it. When should I unblock a program?

Unfortunately, pressing the Unblock button is not sufficient to allow correct operation in the active mode. SWiSH Max will timeout when attempting to obtain directory listings or upload files. The Windows XP firewall must be disabled to allow active FTP transfers. This behaviour is due to the way that the FTP client must accept connections on arbitrary ports if in active mode and is not a software defect associated with SWiSH Max

Note: Some other firewalls (eg. Sygate) can be configured to allow Active FTP transfers, however a detailed listing of those firewalls and configuration description is beyond the scope of this document. We recommend that you use PASV mode for your transfers.

6.2.7.2.1.3 FTP Definitions

Item	**Description**
ASCII file	This is typically a text file. Files transferred in ASCII mode have their format converted to match the destination computer.
BINARY files	These files are used to hold non textual information. Files transferred in BINARY mode are not converted by the transfer process. .swf and .jpg are examples of BINARY files.
FTP	Stands for File Transfer Protocol. This is a commonly used open standard that defines a method of exchanging files between computers.
.swf	Flash (.swf) file format.
ISP	Internet Service Provider. This is the company that provides you with internet access. They may also provide you with a webspace area that allows you to upload published help files.
MD5	A checksum created according to the MD5 standard.
Output folder	The published help file is a combination of multiple files. All of the published files are placed in this directory. All of the files in this folder need to be uploaded to your webspace for public viewing.
packet sniffer	computer software or hardware that intercepts and logs passing network traffic.
SHA1	A checksum created according to the SHA1 standard.

6.2.7.2.1.4 FTP Notes

FTP stands for **File Transfer Protocol**. This is a commonly used open standard that defines a method of exchanging files between computers.

FTP was first defined in 1971. Because of its age, it is commonly supported on most computers.

There are Security, Firewall and integrity issues associated with the use of FTP, however FTP is often the only protocol provided by some ISP's.
FTP transfers user account information and files in a non encrypted fashion. This means that under most network conditions, this information could potentially be seen by other network users using packet sniffers.

In its standard mode, FTP works in an active mode. In this mode, the client must open an arbitrary port to receive the connection. This has obvious firewall issues. This problem can be resolved using passive mode (PASV).

PASV is a later version of the protocol. Unfortunately it is possible that the remote host may not support PASV mode.

PASV mode can be enabled or disabled via the PASV checkbox in the Site Details area of the main FTP dialog.

See also
Firewall Management.
FTP makes no checks to see if the transferred file matches the original file.
eg. If a file transfer is canceled part way through the transfer, the partially transferred file is not marked as incomplete.

It is possible to manually check the integrity of BINARY files by manually calculating MD5 of SHA1 checksums at both the local and remote computer.

As ASCII files are potentially altered by the transfer method to match the end computer configuration, manual checking via checksum is not possible.

This method of transfer is provided where a user account on the server is not required.
The user name for anonymous access is typically 'anonymous' or 'ftp'. The account typically does not need a password although in some cases a server may ask you to supply your email address as the password for verification and logging purposes. If this is the case, enter your user and password information in the normal manner.

If no password entry is required then anonymous mode can be selected via the Anonymous checkbox in the Site Details section of the main FTP dialog.

FTP supports the transfer of data in two modes: ASCII mode and BINARY mode.

ASCII mode is typically used to transfer text files. These files are converted by the protocol into a format that is convenient for the remote machine. Typically this means changing New Line characters (NL) into a New Line Carriage Return sequence when files are transferred between a Unix and Dos / Windows system.

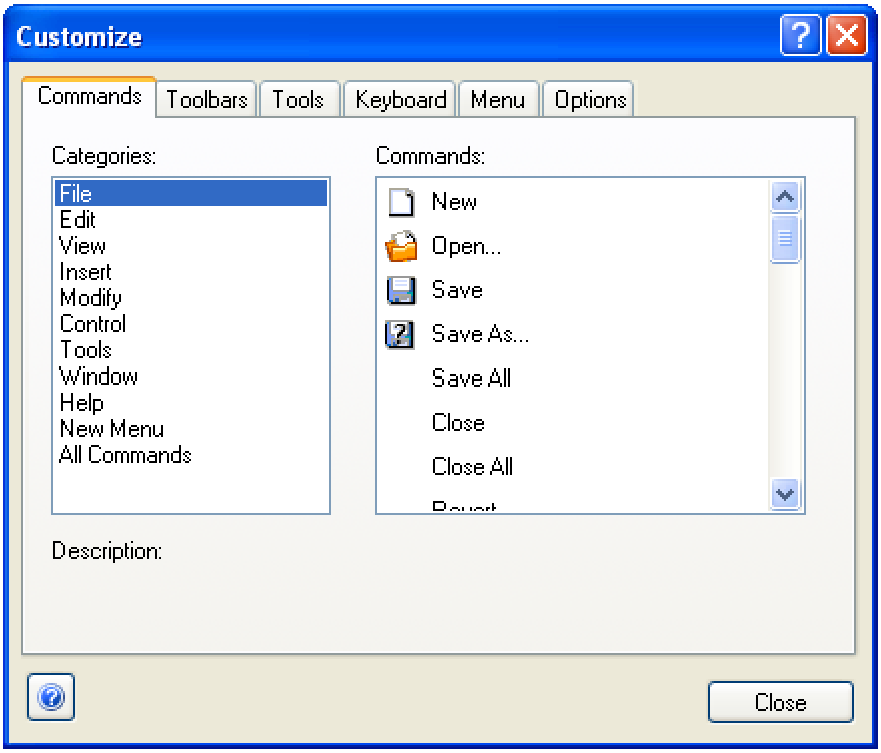
SWiSH Max will automatically assume that all .txt, .xml, .htm and .html files are ASCII.

BINARY mode is used to transfer files where conversion is not wanted. Typically this is all .swf, image and executable files.

SWiSH Max will automatically assume that all files that are not .txt, .xml, .htm or .html are BINARY.

6.2.7.3 Customize

Tools | Customize...
Opens a dialog box to customize the Toolbars and Menus.

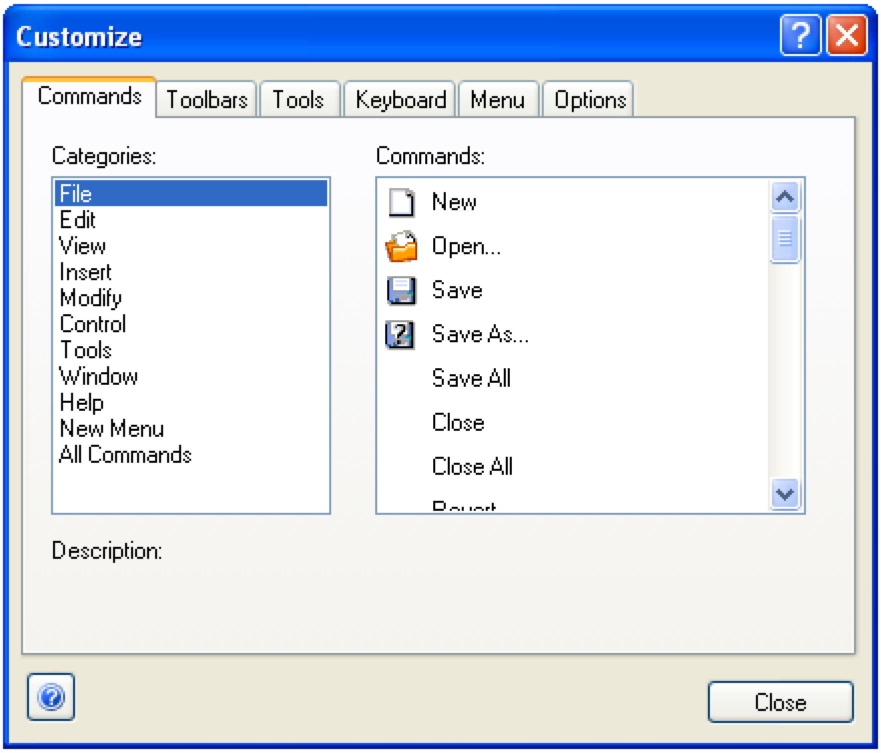


The available tabs are as follows:

- Commands: to add/remove a command to/from a Toolbar
- Toolbars: to show/hide a Toolbar or add/rename/delete a custom Toolbar
- Tools: to add/edit/remove an external command to/in/from the Tools Menu
- Keyboard: to add/remove a shortcut key to/from a command
- Menu: to add/remove a command to/from a Menu
- Options: to configure the 'look' of Menus and Toolbars.

Button appearance can also be changed using the Customize menu tools.

6.2.7.3.1 Commands



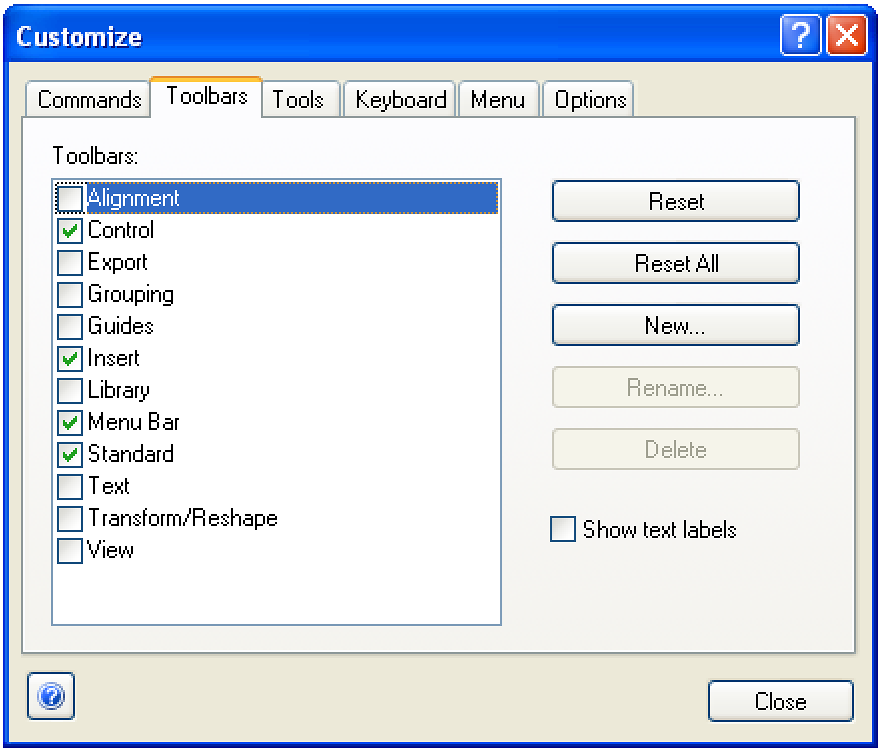
To add a command to a Toolbar

1. Make a selection from the Menu in the 'Categories' box
2. Drag the command you want from the 'Commands' box to the displayed Toolbar

To remove a command from a Toolbar

Simply drag the icon off the Toolbar.

6.2.7.3.2 Toolbars



To show/hide a Toolbar

1. Select an option from the 'Toolbars' box
2. Tick/Untick the check box to show/hide the selected Toolbar

To add a custom Toolbar

1. Click the 'New...' button, and in the 'Toolbar Name' box, type a name for the new Toolbar
2. Click the Commands tab
3. In the 'Categories' box, click the category that contains the command that you want to add to the new Toolbar
4. From the 'Commands' options, drag the command to the new Toolbar

To rename a custom Toolbar

1. In the 'Toolbars' box, click the Toolbar you want to rename
2. Click the 'Rename...' button and type the new name in the 'Toolbar Name' box

To delete a custom Toolbar

1. In the 'Toolbars box', click the Toolbar you want to delete
2. Click the 'Delete' button

To reset a Toolbar to the default settings

1. In the 'Toolbars' box, click on the Toolbar you want to reset
2. Click the 'Reset' button

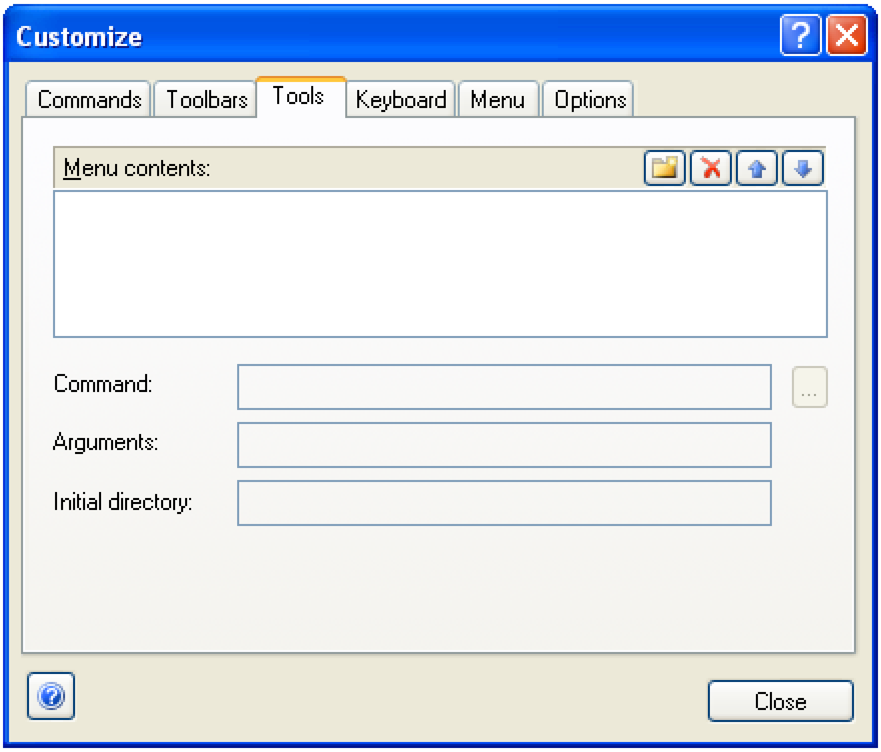
To reset all Toolbars to the default settings

1. Click the 'Reset All' button.

To display text labels on Toolbars

1. In the 'Toolbars' box, click the Toolbar you want the text labels be displayed on
2. Check the 'Show text labels' checkbox

6.2.7.3.3 Tools



To add a command to the Tools Menu

1. Click the 'New' button or double-click the blank area of the 'Menu contents' box
2. Type the name of the Tool as you want it to appear on the Tools Menu. To specify a letter in the Menu title as an access key, precede that letter in the 'Menu contents' box with an ampersand (@). The first letter in the title is the keyboard access key by default
3. Highlight the name of the tool you just entered in the 'Menu contents' box
4. In the 'Command' box, browse using the 'Browse' button or type the path and name of the program
5. In the 'Arguments' text box, browse or type any arguments to be passed to the program
6. In the 'Initial directory' box, type the file directory where the command is located

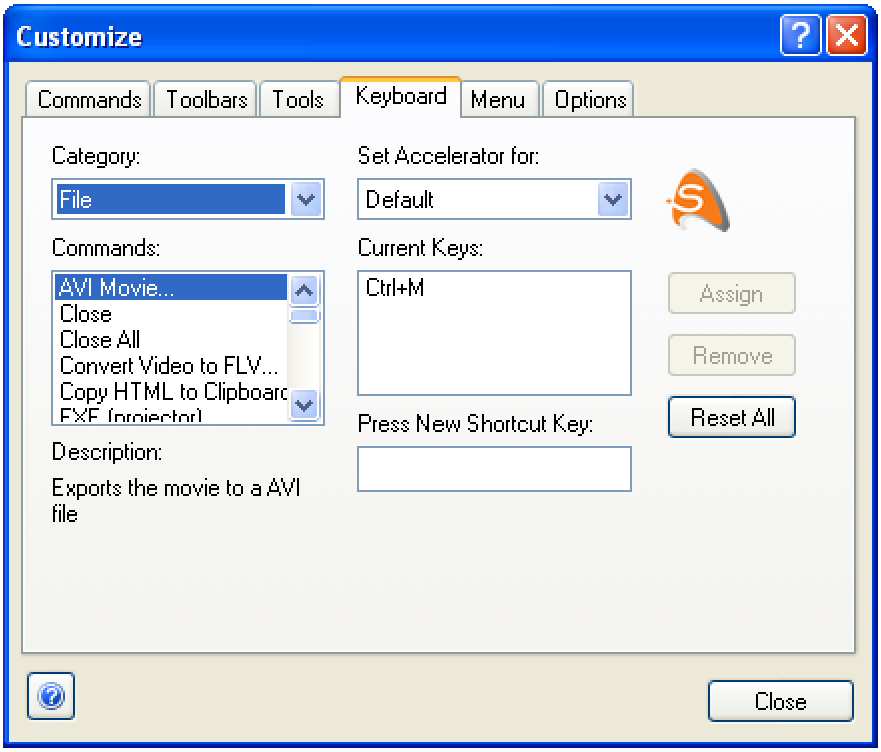
To remove a command from the Tools Menu

1. In the 'Menu contents' box, select the command you want to delete
2. Click the 'Delete' button or press the Delete key

To edit a command in the Tools Menu

1. In the 'Menu contents' box, select the command you want to edit
2. To move the command up/down one position in the Menu, click the 'Move Up' or 'Move Down' buttons, respectively. To change the Menu text, 'Command' line (tool path and file name), command-line 'Arguments', or the 'Initial directory', type the new information in the appropriate text box

6.2.7.3.4 Keyboard



To assign a shortcut key

1. In the 'Category' list, select the Menu that contains the command that you wish to assign a shortcut key
2. In the 'Commands' list, select the command that you wish to assign a shortcut key
3. In the 'Press New Shortcut Key' box, press the shortcut key or key combination that you want and click the 'Assign' button. If you press a key or key combination that is currently assigned to another command, an 'Assigned To' section will appear under the 'Press New Shortcut Key' box

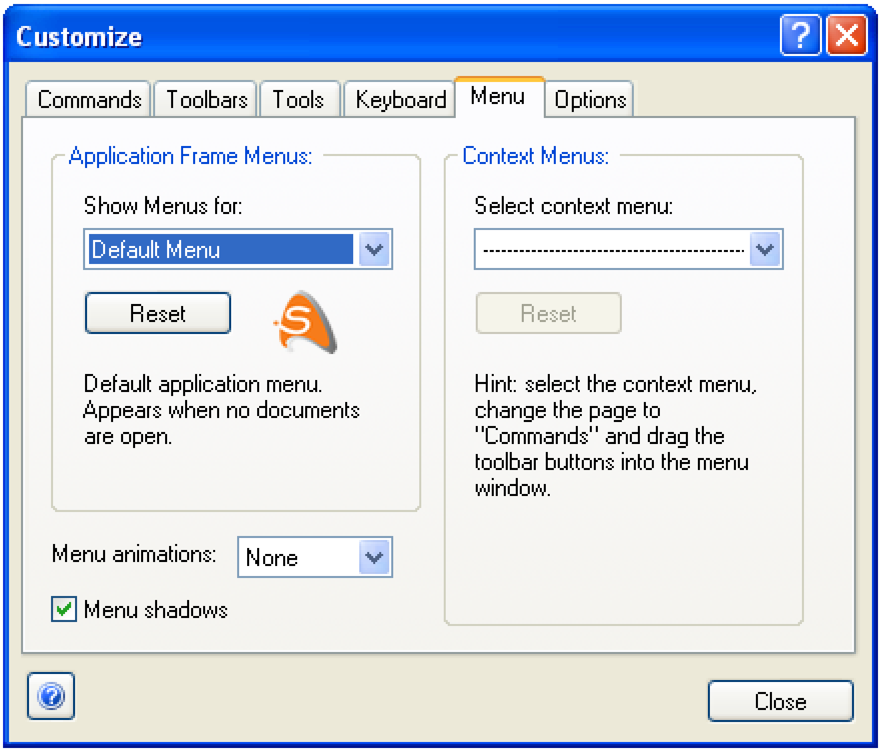
To delete a shortcut key

1. In the 'Category' list, select the Menu that contains the command you wish to delete the shortcut key to
2. From the 'Commands' list, select the command you wish to delete the shortcut key to and click the 'Remove' button

To reset all shortcut keys to their default values

Click the 'Reset All' button.

6.2.7.3.5 Menu



Application Frame Menus:

Show Menus for: Default Menu

Default application menu. Appears when no documents are open.

Context Menus:

Select context menu: --

Hint: select the context menu, change the page to "Commands" and drag the toolbar buttons into the menu window.

Menu animations: None

☑ Menu shadows

To add a command to an Application Frame Menu
1. Use the 'Show Menus for' drop-down list to select the Menu you want to customize
2. Click the Commands tab and drag the commands into the Menu

Note: To remove items, just drag them off the Menu

To reset the Menus to their original configuration
Click the 'Reset' button.

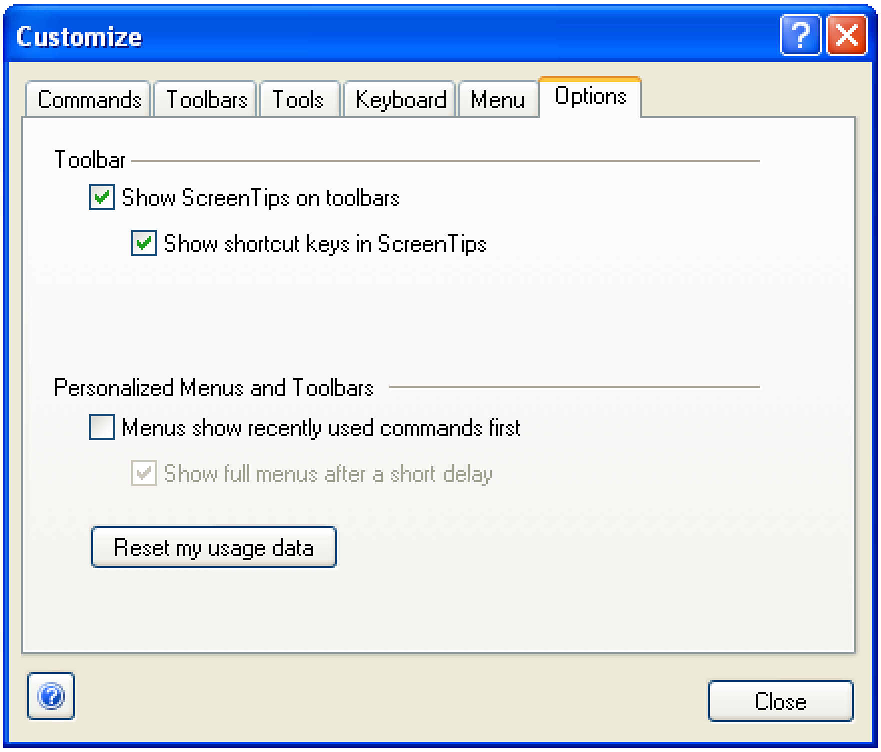
To add a command to a context Menu
1. Use the 'Select context menu' drop-down list to select the pop-up Menu you want to customize
2. Click the Commands tab and drag the commands into the Menu

Note: To remove items from the pop-up Menu, just drag them off

To select the type of Menu animations
Use the 'Menu animations' drop-down list to select the type of Menu animations. The available options are None, Unfold, Slide and Fade.

To display shadows under open Menus
Tick the 'Menu shadows' box.

6.2.7.3.6 Options



Toolbar

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

Personalized Menus and Toolbars

□ Menus show recently used commands first

☑ Show full menus after a short delay

Reset my usage data

Show ScreenTips on Toolbars
Select the 'Show ScreenTips on toolbars' checkbox to display short help text when the mouse pointer is positioned over a 'Toolbar' button.

Show shortcut keys in ScreenTips
Select the 'Show shortcut keys in ScreenTips' checkbox to see the possible keyboard shortcut displayed in addition to the short help text.

Menus show recently used commands first
Select the 'Menus show recently used commands first' checkbox to show only basic and frequently used commands on Menus.

Show full Menus after a short delay
Select the 'Show full menus after a short delay' checkbox to show all commands on the Menu after you rest the mouse pointer over the open Menu for a brief period of time.

Reset my usage data
Click the 'Reset my usage data' button to clear the list of recently used commands saved by SWiSH Max.

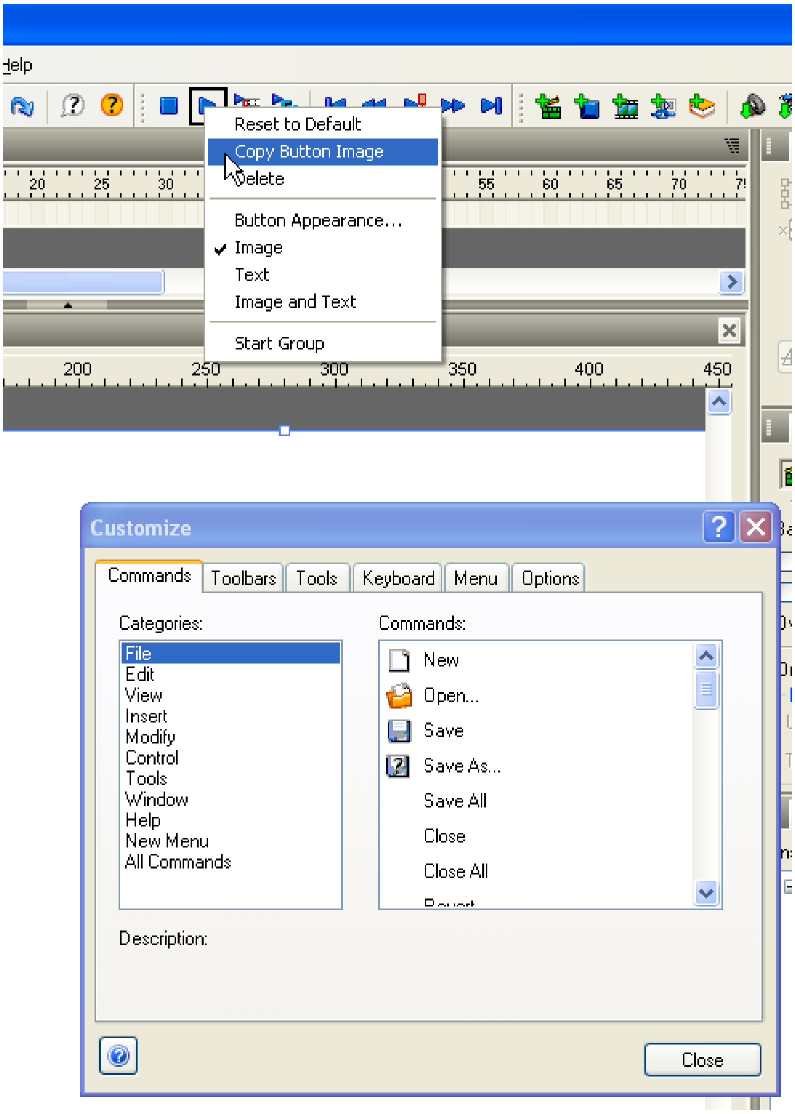
6.2.7.3.7 Button Appearance

The appearance of any toolbar buttons can be edited using the right click context menu when the Customize dialog is open.

Steps:
1. Select Tools | Customize
2. Right-click on (for example) the Play Movie button in the toolbar (see below).

The context menu options are:

- **Reset to Default**: Resets any changes to toolbar buttons to installed default
- **Copy Button Image**: Copies button image to the clipboard
- **Delete**: Deletes the select icon from the toolbar
- **Button Appearance...**: Opens a button appearance editor. See the section on 'Button Appearance Editor' below
- **Image**: Toolbar button shows as an image
- **Text**: Toolbar button shows as text
- **Image and Text**: Toolbar button shows as both image and text
- **Start Group**: Adds group separator to the toolbar



Button Appearance Editor

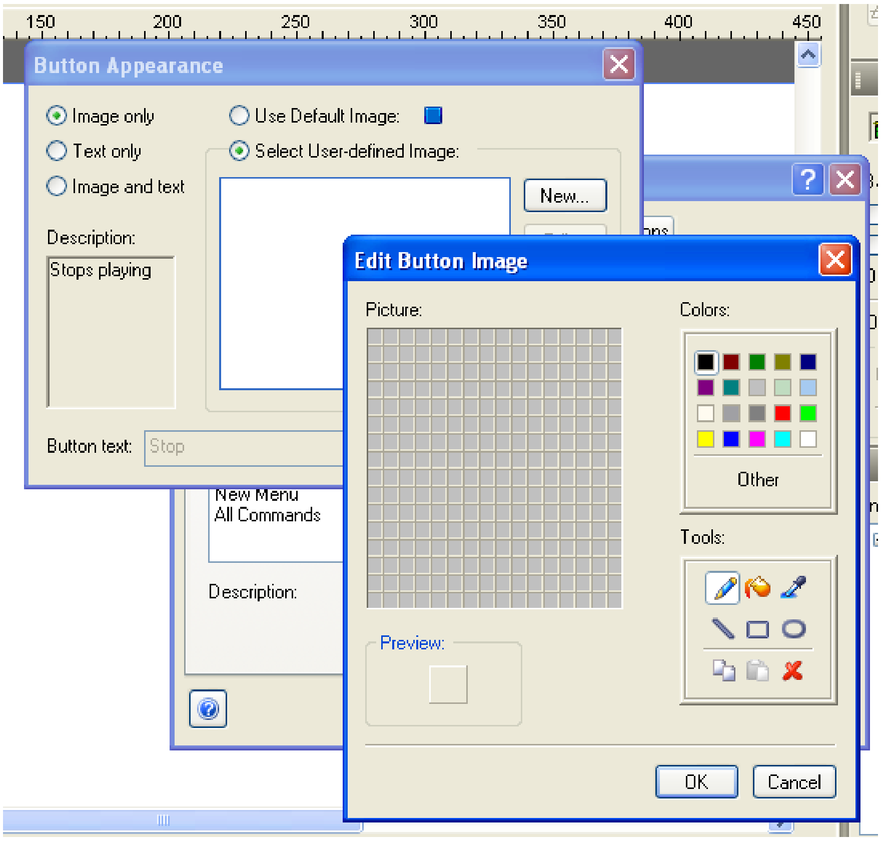
Select 'Button Appearance...' from context menu to display the Button Appearance dialog (as shown below). The Button Appearance dialog edits the display of the button. Options include:

- **Image Only**: Select either the default image or a user-defined image

- **Text Only**: Edit the Button Text field to change the displayed text
- **Image and text**: Edit the Button text field and select to use the default image or a user-defined image

User-defined Image

The 'Select User-defined image' option displays the 'Edit Button Image' dialog (as shown below). Use the pixel editor to create a new icon (16x16) pixel by pixel.



Button Appearance

● Image only
○ Text only
○ Image and text

○ Use Default Image:
● Select User-defined Image:

Description: Stops playing

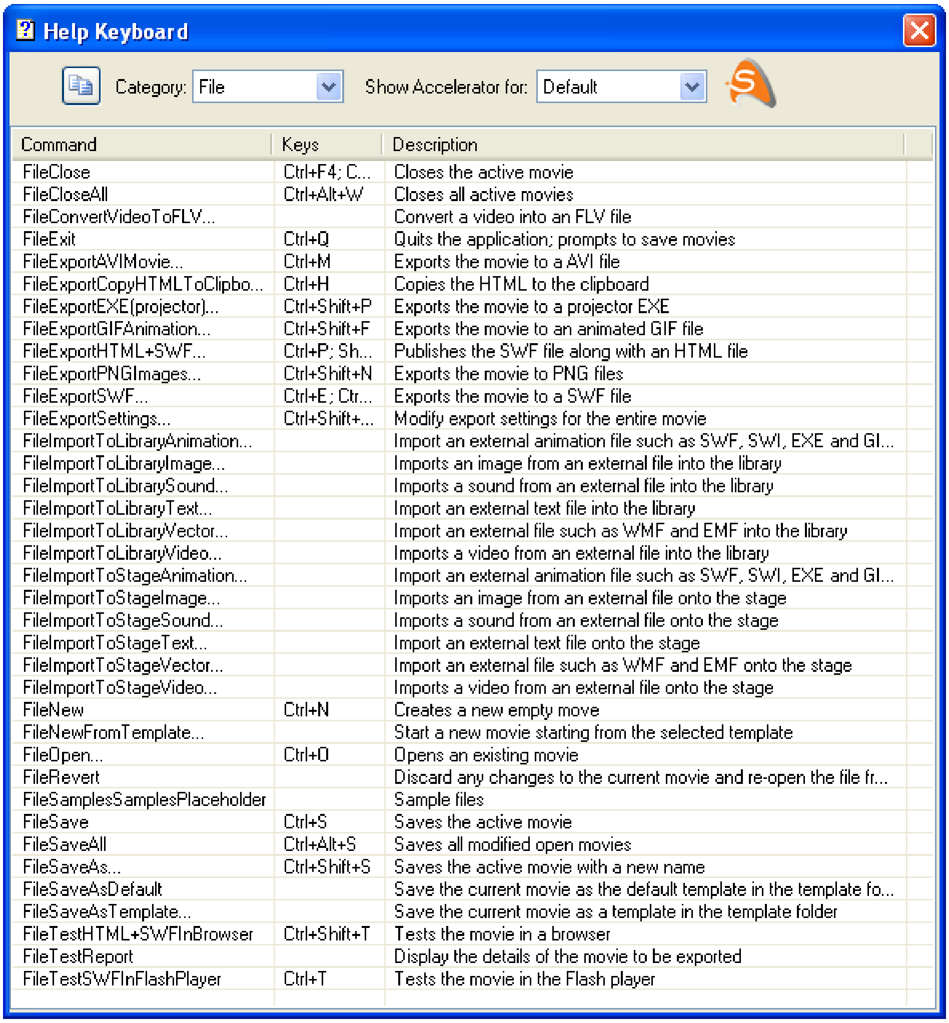
Button text: Stop

6.2.7.4 Keyboard Map

Tools | Keyboard Map

Displays a map of the current keyboard shortcuts and a brief description of functions.

You can modify your keyboard shortcuts at the Keyboard tab on the Customize dialog box.

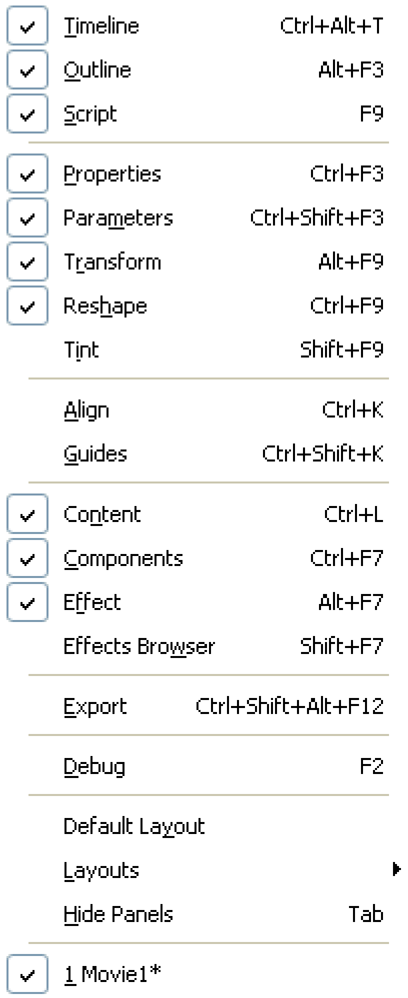


Category: File

Show Accelerator for: Default

Command	Keys	Description
FileClose	Ctrl+F4; C...	Closes the active movie
FileCloseAll	Ctrl+Alt+W	Closes all active movies
FileConvertVideoToFLV...		Convert a video into an FLV file
FileExit	Ctrl+Q	Quits the application; prompts to save movies
FileExportAVIMovie...	Ctrl+M	Exports the movie to a AVI file
FileExportCopyHTMLToClipbo...	Ctrl+H	Copies the HTML to the clipboard
FileExportEXE(projector)...	Ctrl+Shift+P	Exports the movie to a projector EXE
FileExportGIFAnimation...	Ctrl+Shift+F	Exports the movie to an animated GIF file
FileExportHTML+SWF...	Ctrl+P; Sh...	Publishes the SWF file along with an HTML file
FileExportPNGImages...	Ctrl+Shift+N	Exports the movie to PNG files
FileExportSWF...	Ctrl+E; Ctr...	Exports the movie to a SWF file
FileExportSettings...	Ctrl+Shift+...	Modify export settings for the entire movie
FileImportToLibraryAnimation...		Import an external animation file such as SWF, SWI, EXE and GI...
FileImportToLibraryImage...		Imports an image from an external file into the library
FileImportToLibrarySound...		Imports a sound from an external file into the library
FileImportToLibraryText...		Import an external text file into the library
FileImportToLibraryVector...		Import an external file such as WMF and EMF into the library
FileImportToLibraryVideo...		Imports a video from an external file into the library
FileImportToStageAnimation...		Import an external animation file such as SWF, SWI, EXE and GI...
FileImportToStageImage...		Imports an image from an external file onto the stage
FileImportToStageSound...		Imports a sound from an external file onto the stage
FileImportToStageText...		Import an external text file onto the stage
FileImportToStageVector...		Import an external file such as WMF and EMF onto the stage
FileImportToStageVideo...		Imports a video from an external file onto the stage
FileNew	Ctrl+N	Creates a new empty move
FileNewFromTemplate...		Start a new movie starting from the selected template
FileOpen...	Ctrl+O	Opens an existing movie
FileRevert		Discard any changes to the current movie and re-open the file fr...
FileSamplesSamplesPlaceholder		Sample files
FileSave	Ctrl+S	Saves the active movie
FileSaveAll	Ctrl+Alt+S	Saves all modified open movies
FileSaveAs...	Ctrl+Shift+S	Saves the active movie with a new name
FileSaveAsDefault		Save the current movie as the default template in the template fo...
FileSaveAsTemplate...		Save the current movie as a template in the template folder
FileTestHTML+SWFInBrowser	Ctrl+Shift+T	Tests the movie in a browser
FileTestReport		Display the details of the movie to be exported
FileTestSWFInFlashPlayer	Ctrl+T	Tests the movie in the Flash player

6.2.8 Window Menu



The Window Menu manages what is displayed in the application window, for example the panels, their layout (arrangement), and the document which is active. A Panel can be enabled or disabled via the checkbox.

Window | Timeline (Ctrl+ALT+T, ALT+Shift+F3)
Shows or hides the Timeline Panel.

Window | Outline (ALT+F3)
Shows or hides the Outline Panel.

Window | Script (F9)
Shows or hides the Script Panel.

Window | Properties (Ctrl+F3)
Shows or hides the Properties Panel.

Window | Parameters (Ctrl+Shift+F3)
Shows or hides the Component Parameters Panel.

Window | Transform (Alt+F9)

Shows or hides the Transform Panel.

Window | Reshape (Ctrl+F9)
Shows or hides the Reshape Panel.

Window | Tint (Shift+F9)
Shows or hides the Color Tint Panel.

Window | Align (Ctrl+K)
Shows or hides the Align Panel.

Window | Guides (Ctrl+Shift+K)
Shows or hides the Guides Panel.

Window | Content (Ctrl+L)
Shows or hides the Content Panel.

Window | Components (Ctrl+F7)
Shows or hides the Components Panel.

Window | Effect (Alt+F7)
Shows or hides the Effect Panel.

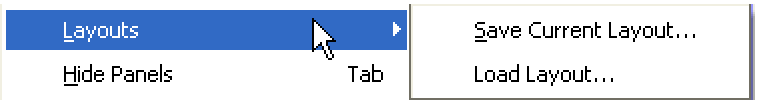
Window | Effects Browser (Shift+F7)
Shows or hides the Effects Browser Panel.

Window | Export (Ctrl+Shift+ALT+F12)
Shows or hides the Export Panel.

Window | Debug (F2)
Shows or hides the Debug Panel.

Window | Default Layout
Shows or hides the Default Layout.

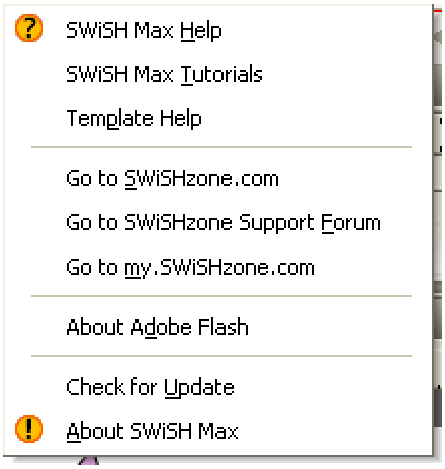
Window | Layouts
Shows the Layouts submenu which lets the user Save the current layout or manage layouts (load, delete etc.). In this example the user has saved their own custom layout as "My Special Layout".



Window | Hide [Show] Panels (Tab)
Toggles Hide and Show for all the Panels. When deselected the previous panels are restored to view.

The final list of movies lists the current documents open by the application.

6.2.9 Help Menu



SWiSH Max Help
SWiSH Max Tutorials
Template Help
Go to SWiSHzone.com
Go to SWiSHzone Support Forum
Go to my.SWiSHzone.com
About Adobe Flash
Check for Update
About SWiSH Max

Help | SWiSH Max Help
Opens the SWiSH Max Help file and lists help topics.

Help | SWiSH Max Tutorials
Opens the SWiSH Help file and shows a list of tutorials.

Help | Template Help or **Help | Movie Help**
This item is conditionally enabled if a help file (.chm) exists and is of the same name as the currently loaded .swi file. Clicking on this item if enabled will open the related help file. This facility allows authors to provide a compatible .chm file for the configuration of their movie.

Help | Go to SWiSHzone.com
Launches the browser and goes to the SWiSHzone.com website at http://www.swishzone.com.

Help | Go to SWiSHzone Support Forums
Launches the browser and goes to the SWiSHzone.com support forums at http://forums.swishzone.com/.

Help | Go to my.SWiSHzone.com
Launches the browser and goes to the SWiSHzone.com customer account at http://my.swishzone.com/.

Help | About Adobe Flash
Launches the browser and goes to the Flash Player web site.

Help | Check for Update
Launches the browser and goes to the SWiSHzone.com web site at http://www.swishzone.com to check if there is an updated build of SWiSH Max. See also Tools | Preferences | Check for Update.

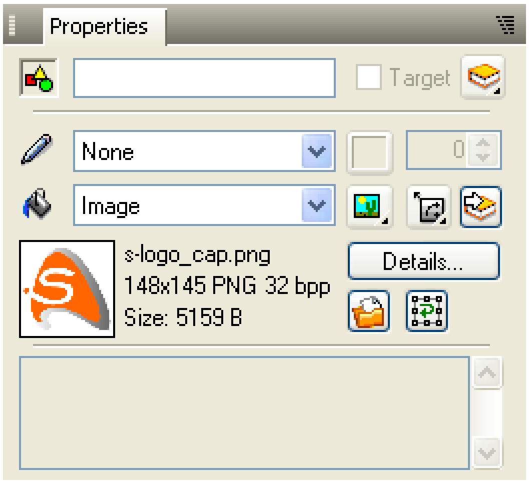
Help | About SWiSH Max
Displays application information, version number and copyright information.

6.3 Panels

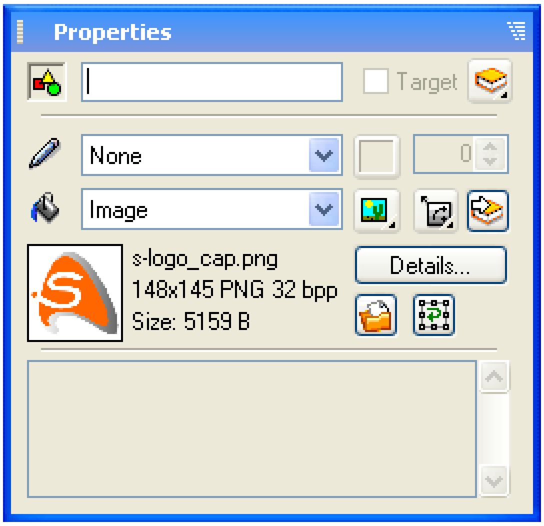
Panels are movable windows to control various options and settings.

Panels can be:
- floated over the 'SWiSH Max application' window

- docked around the edges of the 'SWiSH Max application' window
- in the center of the 'SWiSH Max application' window
- grouped together into a combined tabbed Panel.



Docked Properties Panel



Floating Properties Panel

There are a number of ways to rearrange Panels. Panels can be dragged around (i.e. left-click drag with the mouse) by the title bar or by the tab when grouped into a combined Panel. A panel can dragged to another edge to dock it, or dragged onto the title bar or tab bar of another Panel to form or join a combined Panel, or dragged elsewhere to float over the application window.

To undock a panel

- double click with left mouse button on the Panel label
- left-click and drag label away from the docked position

To dock a panel

- double click with left mouse button on the Panel label
- left-click and drag the panel grip icon (▮) to a docking position
- right-click with mouse on the label or left-click on panel menu icon (☰) and select an appropriate docking option. e.g. for a floating Properties panel the options are: Dock Properties Panel, Dock Properties Panel with Layout, Hide Properties Panel

Turn Panels on and off using the Panel Menu. Some Panels will appear automatically when needed. For example if an Action is added the Script Panel will appear, even if it had previously been turned off.

By default, the Layout Panel is in the center of the 'SWiSH Max application' window. The Script Panel is docked with the Layout Panel. The Timeline is docked across the top. On the right hand side there are three groups of docked panels: Transform and Reshape; Properties and Parameters; Outline, Content Components, Effect.

Panels can be restored to their application default layout using the 'Default Layout' option on the Panels Menu. Multiple layouts can be saved and a user defined default is set using Windows | Layouts | Manage Layouts.

The main application window Panel is the Layout Panel.

SWiSH Max Window menu lists the following Panels:

- Timeline (Ctrl+ALT+T, Alt+Shift+F3)
- Outline (ALT+F3)
- Script (F9)

- Properties (Ctrl+F3)
- Parameters (Ctrl+Shift+F3)
- Transform (Alt+F9)
- Reshape (Ctrl+F9)
- Tint (Shift+F9)

- Align (Ctrl+K)
- Guides (Ctrl+Shift+K)

- Content (Ctrl+L)
- Components (Ctrl+F7)
- Effect (Alt+F7)
- Effects Browser (Shift+F7)

- Export (Ctrl+Shift+ALT+F12)

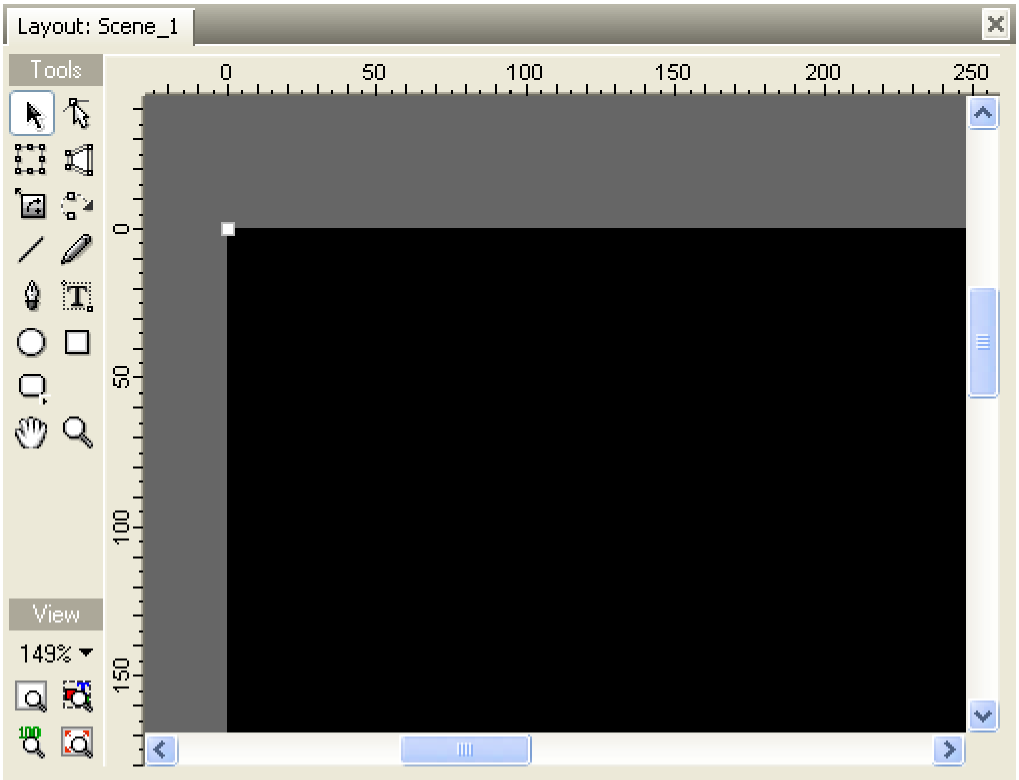
- Debug (F2)

The 'Properties' Panel shows the properties of a selected object. The content of this Panel change depending on what Object (if any) is selected. The different 'Properties' Panels available are:

- Button Object
- Group Object
- Shape Object
- Scene Object
- Movie Clip Object
- Text Object
- Sound Object
- Embedded Video Object
- External Media Object

6.3.1 Layout Panel

The 'Layout' Panel shows how Objects are arranged in the workspace. Arrange and edit Objects, as well as preview Movies, Scenes or Effects in this Panel.



The title on the 'Layout' Panel (shown above as "Scene_1") will be the name of the Scene or Movie Clip currently worked on.

The 'Layout' Panel has three components:

- the workspace and stage
- the Toolbox
- the View Options.

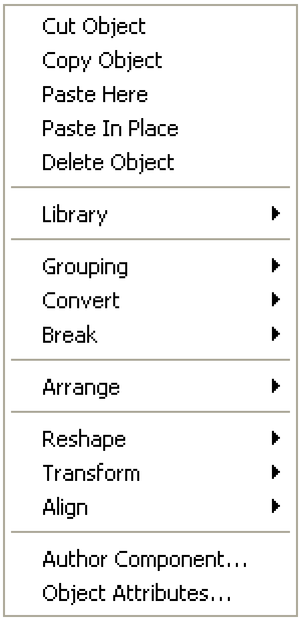
When Objects are selected and altered, different cursor and object anchors will appear depending on the Tool currently selected.

The Workspace and Stage
The main area of the 'Layout' Panel is the workspace. The area within the workspace that corresponds to the viewable area of your Movie is called the 'stage'.

Scroll bars automatically appear on the right and bottom of the workspace when the entire stage area does not fit within the 'Layout' Panel. In the picture above, the stage is too wide to fit in the 'Layout' Panel, so a scroll bar is displayed at the bottom. Pan the workspace by moving the scroll bars or by using the Pan tool.

A Ruler can be displayed at the top and left edges of the workspace. This helps view x and y positions in pixels. Use the Show Rulers option on the View Menu to turn the Rulers on or off. It is also possible to display a Grid over the workspace. Use the Show Grid option on the View Menu to turn the Grid display on or off.

A right mouse click within the 'Layout' Panel will display the following context menu:



This menu provides a convenient shortcut to many frequently performed operations.
The 'Paste Here' option will cause the most recently Cut or Copied object to be placed at the current mouse pointer position. This is different to the Past In Place option where the copied object is placed immediately above the most recently Cut or Copied object.

The Toolbox

The Toolbox is located along the top-left edge of the workspace and contains the Tools that operate on the stage/workspace. The selected Tool determines what SWiSH Max does when the mouse cursor is clicked and dragged around the workspace.

View Options

The Zoom controls are fixed to the bottom left of the workspace and are used to change the Scale of the stage/workspace. Increase the amount of Movie visible by zooming out, or see more detail by zooming in. The Zoom In or Zoom Out functions or the Zoom Tools are also on the View Menu.

Layout Panel

The 'Layout' Panel has three modes of operation:

- editing mode
- 'Preview Frame' mode
- play mode.

Editing Mode

Arrange and edit Objects for the current Scene using the editing mode.

The Main Menus and Toolbars can be used to directly edit selected objects on the stage, or the scene. A context Menu can be displayed by right-clicking on an Object or an unused area of the workspace. This Menu will permit various operations on the Scene or Object.

When using the Tools, select an Object in the 'Layout' Panel by clicking on it. To select multiple Objects use Shift+Click or Control+Click (Ctrl+Click). Objects can be moved by selecting (click) and dragging with the mouse cursor. This changes the reference position of the Object.

The Object selected is displayed with 'handles' around it. Handles are small squares, or circles, at the bounding edges of the Object. Change the Object or its Transform by dragging the handles that appear around it.

When a drawing Tool is active, clicking and dragging will perform whatever Action is appropriate for the selected drawing Tool.

Preview Frame Mode
Select 'Preview Frame' mode by selecting the Preview Frame option from the Control Menu or the Control Toolbar, by clicking on the Frame Ruler in the Timeline Panel, or by selecting the Motion Path Tool. Return to editing mode by turning off the 'Preview Frame' option, or by clicking on the Frame Ruler in the Timeline Panel.

When in 'Preview Frame' mode, the handles around the currently selected Object are red and a red play head appears on the Timeline Panel. Any Motion Path for the currently selected Object is indicated by a dotted line. Each dot on the Motion Path line corresponds to the position of the Object at a given Frame. The motion is made up of one of more Effects. The last Frame of each Effect is called a Keyframe, and is indicated by a larger blue dot in the Motion Path.

Most of the things done in editing mode can also be done in 'Preview Frame' mode. It is necessary to be in the 'Preview Frame' mode to transform an object at a keyframe (i.e. position, scale, angle, skew, color or alpha). Select the keyframe in the timeline and edit the motion settings for the current Effect, rather than from the original position the object was placed.

Play Mode
View the 'Play' mode by selecting Play Movie, Play Scene or Play Effect from the Control Menu or Control Toolbar. This previews the Movie inside the 'Layout' Panel as it would appear on a web page or in the Flash Player. Play Movie plays all the Scenes of the Movie in sequence. Play Scene plays only the current Scene. Play Effect starts at the first Frame of the first selected Effect and plays until the last Frame of the last selected Effect. If only one Effect is selected, it will play the Frames for that Effect only. Play Effect is also available in the 'Effect Settings' dialog box.

If the 'Stop playing at end of movie' checkbox is checked in the Movie Properties dialog the preview will repeat until stopped, otherwise it will play once and stop.

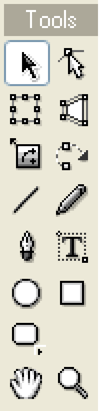
Return to editing or 'Preview Frame' mode by selecting Stop from the Control Menu or the Control Toolbar.

When in play mode, the currently selected Object does not show its handles. Right-clicking on the 'Layout' Panel shows a context Menu with information about the Flash Player, rather than the usual right-click context Menu. In this case, the 'Layout' Panel effectively become a 'Flash Player' window. Also, when in the Play Mode Objects can not be selected or dragged in the 'Layout' Panel.

6.3.1.1 The Toolbox

The Toolbox is fixed to the top-left of the Layout Panel and contains the tools that operate on the workspace. The selected tool determines what SWiSH Max does when you click and drag the mouse on the workspace. For example, when the 'Line' tool is selected, a click and drag will create a line object, but when the 'Rectangle' tool is selected, a rectangular object will be created.

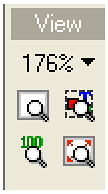
Selecting a tool locks it until another tool is selected. Pressing the control key overrides the current tool and sets the Selection tool.



For more information see the Tools section.

6.3.1.2 View Options

The View Options are fixed to the bottom-left corner of the Layout Panel. The view options allow you to quickly adjust the scale of the workspace and modify grid settings.



147%▾ **Zoom Menu**
Displays a pop-up Menu of common zoom factors (25%, 33%, 50%, 66%, 75%, 100%, 150%, 200%, 300%, 400%, 500%)

View | Fit Scene in Window (Ctrl+2)
To fit the entire Scene (stage area) into the Layout Panel.

View | Fit Objects in Window (Ctrl+3)
To fit all selected Objects into the Layout Panel. If no Object is selected, this function will fit all Objects in the Scene into the Layout Panel.

View at 100% (Ctrl+1)
Views the Movie at its actual size in the Layout Panel.

Adjust Zoom on Layout Change
Resizes the zoom view to be proportional to the size of the Layout Panel.

6.3.2 Timeline Panel

The 'Timeline' Panel contains time-based properties for the current Scene. The Scene is made up of a series of Frames, in the same way that a motion picture is made up of Frames. The Timeline is a visual representation of the Frames with the first Frame at the left and last Frame at the right

The top row shows the Frame Events and corresponding Actions for the Scene. These Actions are executed when the Movie reaches the Frame that the Action is located. Frame Actions always have a duration of one Frame, but you can execute more than one Action in a single Frame.

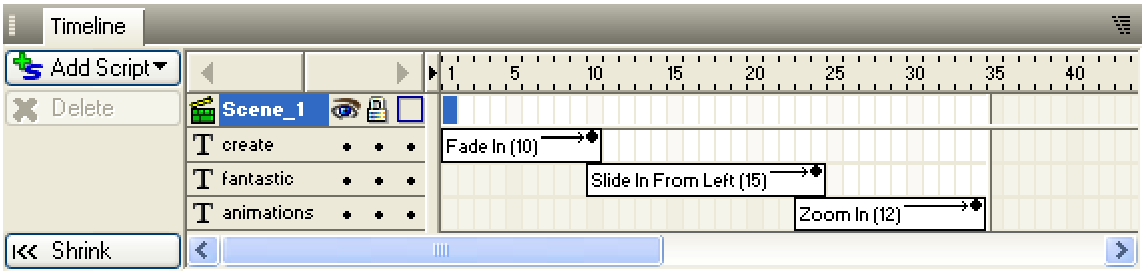
The rows below the Scene row represent the objects in the Scene. The rows are displayed in stacking order, with the object that is in front of all other objects displayed at the top (just below the scene row), and the object that is behind all other objects is displayed at the bottom.

Each object row shows the Effects that are applied to that object. Effects may have a duration of one or more Frames, but only one Effect can be applied to an object at any given Frame.

Seconds are marked in the timeline with black lines. The number or Frames shown between seconds is controlled by your Movie's Frame Rate.

Note:

- A Scene contains zero Frames until you add an Effect or Action
- When editing a Movie Clip, the 'Timeline' Panel displays the Timeline for the Movie Clip



Scene Navigation
Navigate through movies Scenes using the Scene Navigation arrows. These arrows appear only when there is more than one Scene in your Movie.

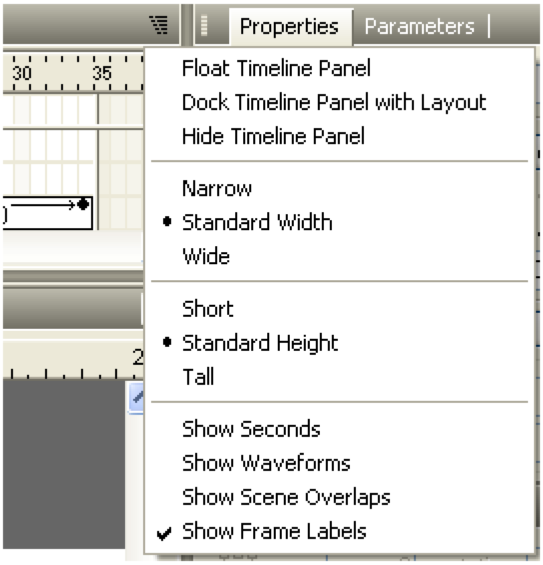
Scene_2 ▶ **Forward Arrow**
Moves to the next Scene in your Movie.

◀ Scene_1 **Backward Arrow**
Moves to the previous Scene in your Movie.

Add Effect ▾ Add Script ▾ **Add Effect / Add Script buttons**
Add a Script to a Scene Frame or add an Effect to an object using these buttons. Display a Menu of Effects or Scripts by clicking on the 'Add Effect' or 'Add Script' buttons, respectively.

Del Object Del Effect Del Script **Delete Object / Effect / Script buttons**
Deletes the currently selected object, Effect or Script. If no Script or Effect is selected the object will be deleted.

Panel Menu options
Modifies the timeline display options.



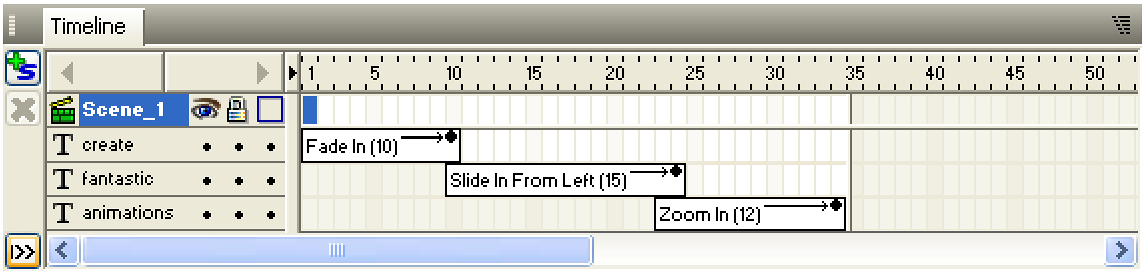
- Float Timeline Panel
- Dock Timeline Panel with Layout
- Hide Timeline Panel
- Narrow
- ● Standard Width
- Wide
- Short
- ● Standard Height
- Tall
- Show Seconds
- Show Waveforms
- Show Scene Overlaps
- ✓ Show Frame Labels

The available options are:

- **Narrow**: makes the timeline frame width narrow.
- **Standard width**: makes the timeline frame width the standard width.
- **Wide**: makes the timeline frame with wide.
- **Short**: makes the timeline row height short.
- **Standard Height**: makes the timeline row height the standard height.
- **Tall**: makes the timeline row height tall.
- **Show Seconds**: seconds are marked in the timeline with black lines.
- **Show Waveforms**: soundtrack waveforms are displayed. See the Insert Sound section.
- **Show Scene Overlaps**: displays overlap of scenes
- **Show Frame Labels**: displays the frame label set by setLabel() script

|<< Shrink **Minimize/Maximize**

This button toggles the Timeline buttons between maximized (text and icon displayed) and minimized (icon-only displayed) states. The screen capture below show the shrunk version of the capture above.



Timeline control

This is the most complex control in SWiSH Max, and is the key to combining and coordinating animations. The basic components of the Timeline control are listed below.

Frame Ruler

The Frame Ruler is located at the top of the Timeline and shows the Frame number. It also has markers for each one-second interval.

Play Head

The Play head also appears on the Frame Ruler when in Preview Frame mode.

Object Labels

Object Labels are used for selecting objects or the Scene or Movie Clip.

'Up' Button

An 'Up' button will appear above the object list when editing the Timeline for a Movie Clip. It selects the Movie Clip or Scene the next level up in the stacking order.

Close Symbol Editor

When editing a symbol the timeline will display a button to leave the Symbol Edit mode.

Close Movie Clip Editor

When editing a Movie clip using the Open in Layout button, this button will appear in the timeline. Use this button to close the editor.

Scene/Movie Clip Row

The Scene or Movie Clip row is always the top row in the Timeline, although it may not be visible when scrolled down the Timeline control. This is where Frame Events can be added or edited. The corresponding Frame Action is automatically executed when the Movie reaches the Frame it is placed in.

Object Rows

Objects rows appear under the Scene Row. They appear in stacking order, the same as in the Outline Panel . That is, objects at the top of the list will obscure objects at the bottom of the list in the Layout Panel. This is where Effects can be added, moved or edited.

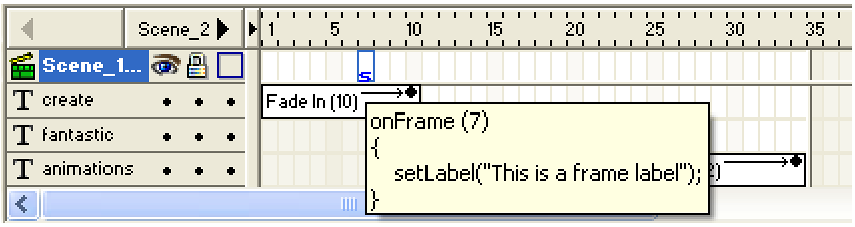
Current Frame

The current Frame is highlighted. In Preview Frame mode, the red Play head also indicates the current Frame.

Using Frame Actions

Adding Frame Actions

Add Frame Actions by either clicking on the 'Add Script' button (while the Scene/Movie Clip row is selected) or by right-clicking on an empty Frame in the Scene/Movie Clip row. Any of the available Actions can be placed as a Frame Action. After adding the Action, SWiSH Max displays a small icon in the Frame indicating the type of Action. When the mouse cursor is moved over the Action Frame, a Screen-Tip displays which Actions are executed at this Frame. The screen capture below show the script associated with the action.



```
onFrame (7)
{
  setLabel("This is a frame label");
}
```

Note: A Frame Action is automatically executed when the Movie reaches the Frame in which it is placed

Moving Frame Actions

Move Frame Actions by clicking and dragging the Frame to a new position. Select multiple Actions and drag them together.

Editing Frame Actions

Edit Frame Actions by clicking on the Frame. The Script Panel will show the selected Action. If the 'Script' Panel was not visible, then double-clicking on an Action Frame will display the 'Script' Panel.

Copying and Pasting Scripts

Copy Scripts by right-clicking on a Script and choosing Copy from the context Menu or by pressing the Ctrl+C key. Similarly paste a copied Script by right-clicking on an empty Frame in an object row and choosing Paste from the context Menu or by using selecting a Frame and pressing the Ctrl+V key.

Deleting Frame Actions

Delete Frame Actions by selecting the desired Action (frame) and then clicking the 'Delete Script' button.

Using Object Effects

Adding Effects

Add Effects either by clicking on the 'Add Effect' button (while an object row is selected) or by right-clicking on an empty frame in an object row. Select an Effect to add from the this Menu. After adding the Effect, SWiSH Max displays a rectangle with the Effect title and duration in the Timeline and also displays the Effect Properties Panel. When the mouse cursor is moved over the Action Frame, a Screen-Tip will display the name of the Effect if it is partially obscured.

Moving Effects

Move Effects by clicking and dragging the Effect to a new position on the Timeline. Multiple effects, on a single object row, can be selected and dragged together.

Resizing Effects

Most Effects can be lengthened and shortened by dragging the ends of the Effect in the Timeline.

Note: Some Effects (such as Place and Hide) always only a single Frame in length. Others, like Typewriter, have their length determined by the Effect settings and the object to which they are applied. In these cases, the Effect cannot be resized directly on the Timeline.

Editing Effect Properties

Edit the settings for an Effect by double-clicking on the Effect in the Timeline, or select the Effect and observe the Effect Properties Panel. The pages on the Effect Properties dialog box is different for each type of Effect.

Deleting Effects

Delete Effects either by selecting them and then pressing the 'Delete Effect' button, or by right-clicking on an Effect and choosing 'Delete Effect' from the context Menu.

Copying and Pasting Effects

Copy Effects by right-clicking on an Effect and choosing Copy from the context Menu or by pressing the Ctrl+C key. Similarly paste a copied Effect by right-clicking on an empty Frame in an object row and choosing Paste from the context Menu or by using selecting a Frame and pressing the Ctrl+V key.

Previewing Effects

Preview Effects by right-clicking on the Effect and selecting Play Effect from the context Menu, or by using the Control Menu or Toolbar or by clicking on the Play Effect button on the top of the Effect Properties Panel. Stop the preview by selecting the Stop option on the Control menu or Toolbar.

Inserting, Deleting and Playing Frames

Right-clicking a Frame in the Timeline displays the options for inserting, deleting or playing frames. To insert, delete or play multiple frames right-click and drag over a number of frames in the Timeline.

Right-click on an individual Frame only inserts or deletes that frame.

The available options are:

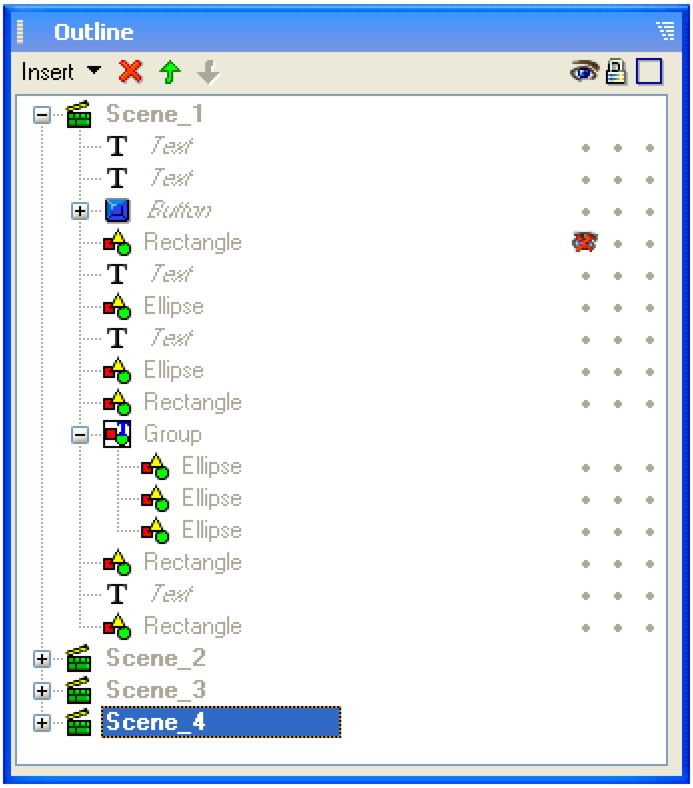
- **Insert Frame(s)**: Inserts the selected number of frames at the current frame
- **Delete Frame(s)**: Deletes the selected frames
- **Play Frame(s)**: Plays the selected frames
- **Insert Second(s)**: Inserts one second worth of frames (the same number of frames set as the frame rate on the Movie panel)
- **Delete Second(s)**: Deletes one second worth of frames (the same number of frames set as the frame rate on the Movie panel).
- **Inserting Keyframes**

This option splits a Move effect into two (split at the frame the keyframe is added). If 'Insert Keyframe' is used before any other effects, it will create a Move effect from Frame 1 up to the selected frame. If it is used after another effect, it will create a Move effect from the first frame after the last effect up to the selected frame.

Note: There are limitations and/or unexpected behavior if the move effect being split has a curved path or has easing. In those cases, the result will not be the same as the original Move effect. Select multiple objects and one frame to insert keyframes on all objects at once.

6.3.3 Outline Panel

The 'Outline' Panel shows an overview of the structure of the entire Movie.



The 'Outline' Panel has a row of buttons at the top. Below this is the 'Outline' tree. The order of Scenes in the tree determines the order in which the Scenes are played. The order of objects within a Scene or Movie Clip determines the stacking order. Objects higher up in the stacking order appear in front of objects further down, within the same Scene or Movie Clip.

Insert ▾ **Insert**
Insert Scenes and objects using the 'Insert' button. The 'Insert' button menu offers a list of Objects which can be inserted. See also the Insert Menu or the Insert Toolbar.

✖ **Delete**
Clicking the 'Delete' button will delete currently selected object(s) or Scene. Attempts to delete a Scene will demand confirmation of the action.

↑ **Move Up and** ↓ **Move Down**
'Move Up' and 'Move Down' buttons change the order of Scenes, or the stacking order of objects. Objects can also be reorder by dragging with the mouse, using the ***Ctrl+Up*** and ***Ctrl+Down*** keys, or using the Order options on the Modify Menu or the Standard Toolbar.

Outline Tree
The 'Outline' Tree show the Scenes and objects within the Movie.

Note:

- Scenes are on the root level of the tree
- objects are shown as children of Scenes
- Group, Movie Clip and Button Objects can have their own object children
- each item in the tree has an icon indicating the type of object
- all objects have edit state attributes which can be checked during editing: eye, padlock and outline view icons

Select Scenes or objects by clicking on them in the 'Outline' tree. Use ***Shift-Click*** and ***Ctrl-Click*** for multiple selections. Copy Scenes by using ***Ctrl-C*** and paste them with ***Ctrl-V***.

Edit the name of the selected Scene or object by clicking on its name or by pressing the ***F2*** key.
For text objects that do not have the 'Has Name' option checked, this will also change the text itself.

Change the playing order of Scenes or the stacking order of objects by dragging, up and down within the 'Outline' tree (click-Drag with the left mouse).

Expand (or contract) the branch for a Scene, Group or Movie Clip by clicking on the small plus sign (or minus sign) next to its name.

Double-clicking in parts of the Outline tree produces the following reaction:

- the outline tree object label displays the object's Properties Panel
- a Movie Clip label selects the object in the the Timeline Panel
- on any of the attribute icons (eye, lock, outline) displays the Object Attributes

To display a context Menu of options appropriate for an object or Scene right-click on the Object in the 'Outline' tree.

Hide objects in the Layout Panel while in edit or 'Preview Frame' mode by turning off the eye icon next to its name. Click on the eye icon to toggle between 'on' or 'off'. You can also use the corresponding commands on the Edit Menu.

6.3.3.1 Object Edit State

To assist with the selection of overlapping objects, each object can have its edit state set to Show/Hide (• / ✖), Unlocked/Locked (• / 🔒) and the Outline shown (• / □). The edit state is toggled between the default state (•) and enabled state by clicking on the Icon.

Visible 👁 (i.e. default state •)
The object is visible and can be selected for editing from the Layout Panel. This Object will obscure objects that are behind it and will prevent them from being edited.

Invisible ✖
The object is invisible and can not be selected for editing from the Layout Panel. Objects normally obscured by this object can be edited. The object still will be visible during movie playback.

Locked 🔒
The object is visible but can not be selected for editing from the layout panel. This state is useful to prevent individual objects from being involved in a select all statement. Default state (•) is unlocked.

Outline View □ **or** □
Displays the outline of the object. This can be useful for showing a 'wireframe'-like view of objects. Default state (•) is disabling the object outline.

6.3.4 Script Panel

The 'Script' Panel contains all the Events and Actions for the currently selected Scene or object.

The Script Panel has three different sections:

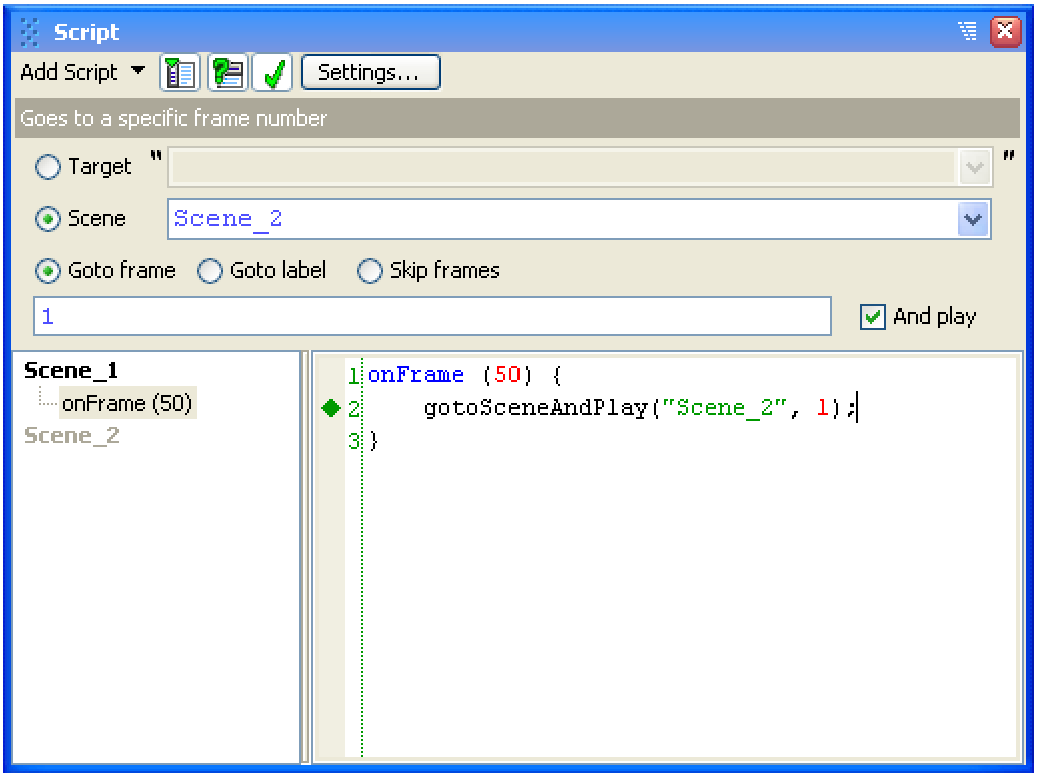
- **Script editor:** the main script text editor.
- **Script Outline Tree pane** (▤): the section on the left which contains a tree of script functions. Can be toggled on/off using the icon
- **Script Assistant pane** (▤): the section at the top which displays the parameters of selected script. Can be toggled on/off with icon

Add Script ▾ **Add Script**
The 'Add Script' button at the top of the panel displays the Script tree. The Script tree can be used to select a specific event or action. The Script Assistant Pane displays the 'Script Command Properties' of the currently selected event or action.

✓ **Highlight Script Errors**
The 'Highlight Script errors' button displays syntax errors in the script panel.

Settings... **Settings**
The 'Settings' button opens the Script Panel preferences which is also available from Tools | Preferences | Script.
The exact layout of the Script Panel will depend on the function/command that has been selected. In the example below, the Panel shows the properties of the currently selected Goto Frame Action.

Note: There is a draggable splitter between the Script Outline Pane and the Script editor to resize the panel as needed.



Goes to a specific frame number

○ Target " "

● Scene Scene_2

● Goto frame ○ Goto label ○ Skip frames

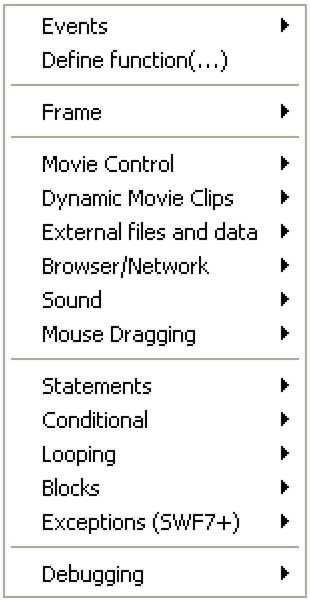
1 ☑ And play

Scene_1
onFrame (50)
Scene_2

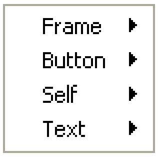
```
onFrame (50) {
    gotoSceneAndPlay("Scene_2", 1);
}
```

Add Script

This button adds a new Event Handling Function, Action or User Defined Function. Pressing the 'Add Script' Button displays the 'Script' tree as shown below:



The first section of the 'Script' tree contains the available Events as below:



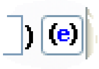
The 'Define function(...)' option is also displayed in this section, which is used to create User Defined Functions.

The second section of the 'Script' tree contains the available Actions.

Deleting Script (Events / Actions)
Delete Events and Actions by selecting them in the 'Script' tree and then clicking on the keyboard 'DEL' key. If an Event is deleted its associated Actions will also be deleted. Similarly, if an If Frame Loaded Action is deleted, its child Actions will also be deleted.

Script tree
This shows the Events and Actions for the currently selected Scene or object. Events are shown at the root of the tree, and Actions are shown as children of Events.

Script label bar *(Script Assistant Pane)*
This bar show the currently selected Event or Action and parameter.



Expressions *(Script Assistant Pane)*
Most actions and event functions accept parameters. In some cases string parameters are expected, eg."frame_1". The (e) button can be used to enter a formula that results in the creation of the appropriate string.

Once the (e) button is selected for a field:

- Expressions can be entered
- The field will be displayed in brackets
- The script entered will be display in a blue font

Right-clicking the field will display the Script Syntax menu
eg. If the variable fr contains 1, then the expression ("frame_" add fr) would evaluate to "frame_1" which would then be used as the action parameter.

6.3.5 Properties Panel

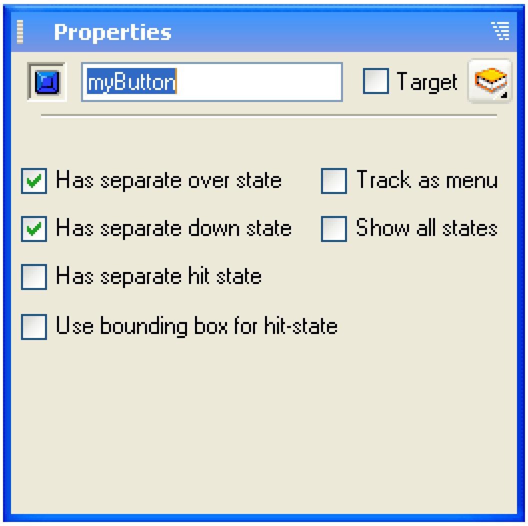
The 'Properties' Panel shows the properties of a selected object. The content of this Panel change depending on what Object (if any) you have selected.

The different 'Properties' Panels available are:

- Button Object Properties Panel
- Group Object Properties Panel
- Scene Object Properties Panel
- Shape Object Properties Panel
- Movie Clip Object Properties Panel
- Text Object Properties Panel.
- Sound Object Properties Panel
- External Media Object Properties Panel

6.3.5.1 Button Object Properties Panel

The Properties Panel for a Button Object shows the properties for the selected Button.



myButton □ Target

☑ Has separate over state □ Track as menu
☑ Has separate down state □ Show all states
□ Has separate hit state
□ Use bounding box for hit-state

See more information about buttons in the sections on the Insert Button and the Button Object.

Name *(shown above as "myButton")*
The name of the Button as also shown in the Outline Panel. If the object is set as a Target the name is the script reference.

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

Add to Library
Adds the object to the library or links to an existing Library Object. See more detail at Library.

Track as menu
Button tracking refers to how a button behaves as it tracks the movement of the mouse. A Button Object can track the mouse in one of two modes:

- as a push button
- as a Menu.

The default setting to track as a push button (i.e. Track as menu turned off).

If the button is tracking as a push button (i.e. 'Track as Menu' is not checked), while playing a Scene or Movie Clip, when the button is clicked all mouse movement Events are directed to the push button until the mouse button is released. This is refer to as 'capturing' the mouse. For example, if the button is clicked and the cursor dragged outside the button (without releasing) the button changes to the Over state, and the pointer remains a hand cursor.

If the button is tracking as a Menu (i.e. 'Track as menu' is checked), then the mouse is captured when the button is clicked. For example, if a Menu button is clicked and the cursor dragged outside the button (without releasing), the button changes to the Up state, and the pointer reverts to an arrow cursor.

Show all states
If set, all button states are displayed concurrently while editing.

Has Separate Over State
When selected adds the button's over state to the Outline Panel. Editing the button's over state modifies the object shown when the mouse is over the button. For example a different colored version of the original button object may be shown when the button is scrolled over.

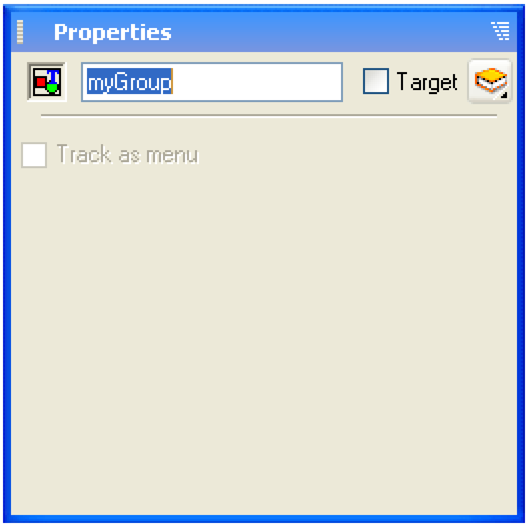
Has Separate Down State
When selected adds the button's over state to the Outline Panel. Editing the button's down state modifies the object shown when the mouse clicks on the button. For example a different colored version of the original button object may be shown when the button is selected.

Has Separate Hit State
When selected adds the buttons over hit to the Outline Panel. Editing the button's hit state modifies the bounds for selecting the button. For example a larger version of the original button object may be used as the buttons hit state which would make the clicks just outside of the button be counted as a button click.

Use bounding box for hit-state
When checked, uses the bounding box as the hit area, otherwise uses the actual shape or letters themselves.

6.3.5.2 Group Object Properties Panel

The 'Group' Panel shows properties for the current Group.



Name *(shown above as "myGroup")*
This is the name of the Group as also shown in the Outline Panel. If the object is set as a Target the name is the script reference.

Note: The Properties Panel for a Group Object will be shown when a button state is selected. In that situation the name of the button state can not be changed.

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

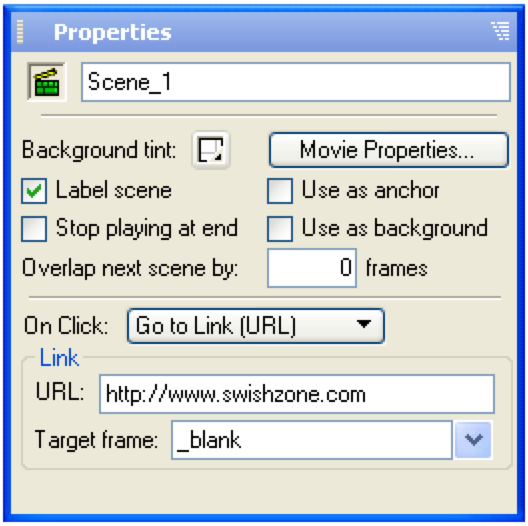
Add to Library

Adds the object to the library or links to an existing Library Object. See more detail at Library.

Track as menu
'Track as menu' sets how a Button Group behaves in response to the mouse selection. If checked, the mouse is captured when the group is clicked. This option is greyed out unless the shape has a mouse event attached to it, such as an on(rollOver) event. Refer to Button Object Properties Panel for more information.

6.3.5.3 Scene Object Properties Panel

The Properties Panel for a Scene shows properties for the selected Scene.



Properties

Scene_1

Background tint: Movie Properties...

☑ Label scene ☐ Use as anchor

☐ Stop playing at end ☐ Use as background

Overlap next scene by: 0 frames

On Click: Go to Link (URL)

Link

URL: http://www.swishzone.com

Target frame: _blank

See more information about Scenes in the section Insert Scene.

Name *(shown above as "Scene_1")*
This is the name of the Scene as also shown in the Outline Panel. It can also be used to identify the Scene in Goto Frame Actions and in script references.

Background tint and color picker ( )
Changes the background tint color for the current Scene using the using the Color Picker. If the alpha value is less than 100%, the Movie background color can be tinted. If the alpha value is 0% (transparent is the default setting) the Scene will have the same color as the Movie default.

'Movie Properties' button
This sets the common properties of the Movie.

Label Scene
Labels the first frame of the Scene with the Scene name.

Use as anchor
Specifies the Scenes to be used as web page anchors, allowing the Forward and Back buttons in a web browser to be used to navigate through the Movie.

Stop playing at end
Automatically stops the Movie at the end of the Scene.

Use as background
Sets the scene to be displayed behind all subsequent scenes.

Overlap next scene by [__] frames

Scenes can made to overlap by the specified number of frames permitting an overlapping transition between scenes. This option is not available if scene is used for the background (see above).

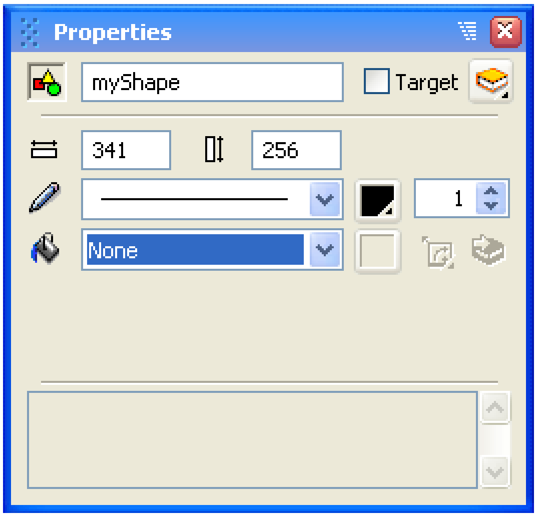
On Click
Sets the following Actions for mouse clicks during the scene:

- **Do nothing:** No Action
- **Go to Next Scene:** Advance to next Scene
- **Go to Link (URL):** Sets a URL for the entire Scene. If the Scene is clicked while the Movie is playing the browser will display this URL in the specified HTML Target Frame i.e. not the Movie frame but the HTML frame.

Note: If the Scene contains an event handler for On Release, this will override the On click Action.

6.3.5.4 Shape Object Properties Panel

The Properties Panel for a Shape Object shows the properties for the selected Shape.



Properties

myShape □ Target

341 256

1

None

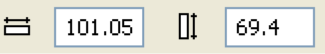
See more information about shapes in the sections on the Line Tool, Pencil Tool, Pen Tool, Ellipse Tool, Rectangle Tool, Autoshape Tool as well as the Shape Object.

Name *(shown above as "myShape")*
The name of the shape as also shown in the Outline Panel. If the object is set as a Target the name is the script reference.

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

Add to Library
Adds the object to the library or links to an existing Library Object. See more detail at Library.

The **Width** and **Height** of the object can be adjusted via these fields:

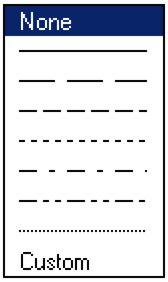


101.05 69.4

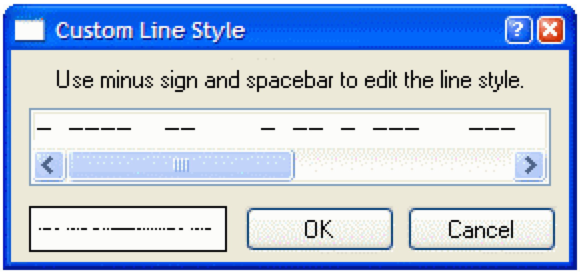
This sizes entered here resize the shape instead of scaling it. Scaling will be applied if the shape is resized via the Transform panel.

Line style

Select from a number of line styles for the line around the outside of the shape. The default style is solid. If a line around the outside of the shape if not needed, select 'None' from the Menu.



The custom option defines a line style using the 'Custom Line Style' dialog box. Enter a combination of minus signs and spaces. The window at the bottom of the dialog box offers a preview of the line style created. Press 'OK' to accept the custom line style.


Custom Line Style

Use minus sign and spacebar to edit the line style.

Line color

Set the color of the line around the outside of the shape (if any) using the Color Picker. The default color is black.

Line thickness

Set the thickness of the line around the outside of the shape (if any). The default thickness is 1.

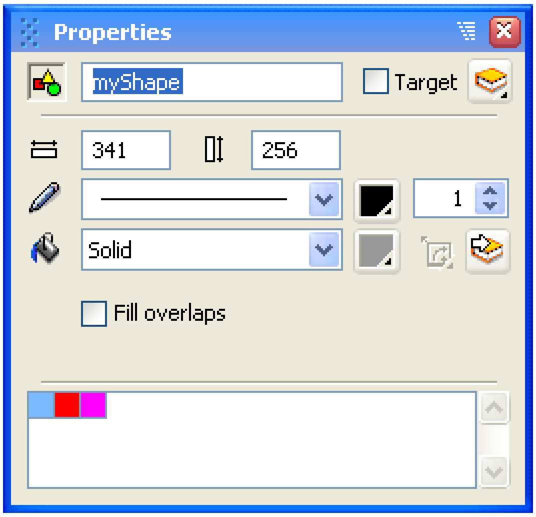
Fill style

Select the fill style of a shape from a number of styles. The default setting is solid, though if the interior is to be hollow, select 'None' from the Menu.

- **Solid**: uses a single color, which may have a degree of transparency (alpha value). Refer to Solid Shape Object Properties Panel for more details.
- **Gradient**: uses a smooth change in colors in either Lineal or Radial bands across the shape. Refer to Gradient Shape Object Properties Panel for more details.
- **Image**: uses a bitmap image to fill the shape. Refer to Image Shape Object Properties Panel for more details.

6.3.5.4.1 Shape Object - Solid Fill

This section discusses Properties Panel settings for the Shape Object with a Solid fill. Refer to Shape Object Properties Panel for details on settings common to all Shape Object Panels.



File Style - 'Solid'
Sets a solid fill for the shape.

Fill color
Set the color of the fill using the Color Picker. The default color is black.

Add to Library
Add the color to the library. See also Library. If there are existing Color resources, these can be linked to the current color fill. Select the Library item and click on the link icon ( ) to link the Objects fill to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object as a copy of the library Resource.

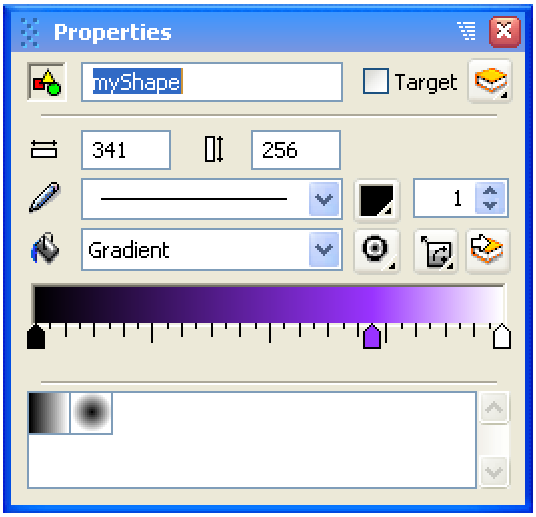
Note: If color settings match an existing library resources that resource will be used (as copy or link).

Fill overlaps
This function forces the shape to have no border and to be filled with a single solid color. Any areas where the shape has overlaps will be filled. This is different to the usual behavior of shapes, where overlapping areas are left hollow. You may find this setting useful when setting up masks for Movie Clips. The default setting is to have this function turned off.

Note: There is a bug in Flash, such that it will crash when trying to import a .swf file that has the 'Solid shape with overlap filled' option turned on for any shape.

6.3.5.4.2 Shape Object - Gradient Fill

This section discusses Properties Panel settings for the Shape Object with a Gradient fill. Refer to Shape Object Properties Panel for details on settings common to all Shape Object Panels.



File Style - 'Gradient'

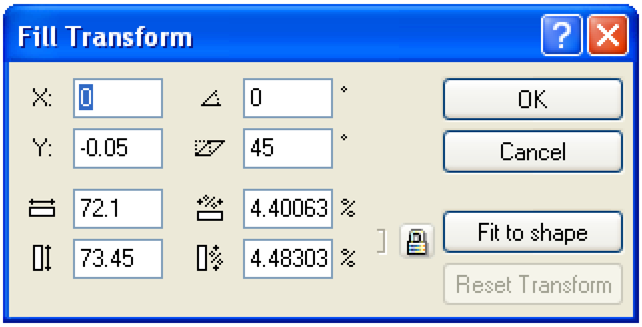
The Gradient Fill type can be:

- **Lineal gradient:** uses a smooth change in colors in straight bands across the shape. The Fill Control lets you change the colors in the gradient (see below).
- **Radial gradient:** uses a smooth change in colors in circular bands across the shape. The Fill Control lets you change the colors in the gradient (see below)

Fill Transform

Defines how the gradient fills the shape. Options are either 'Fit to Shape' which always fit the shape while reshaping, Edit Transform or 'Edit Fill Transform...' which presents a dialog for custom setup.

The 'Fill Transform' dialog box is represented below.



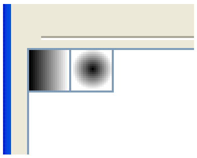
Fill Transform

X: 0
Y: -0.05
72.1
73.45
0 °
45 °
4.40063 %
4.48303 %

The controls are the same as for the Transform Panel. However, in this case, the Transform applies to the fill pattern within the shape, rather than the shape within the Scene. You can also visually edit the fill Transform using the Fill Transform tool.

Add to Library

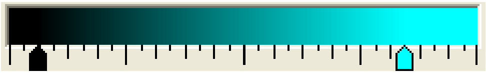
Add the Gradient to the library. See also Library. If there are existing Gradient resources (shown lower in the dialog, see below), these can be linked to the current gradient. Select the Library item and click on the link icon ( ) to link the Objects fill to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object as a copy of the library Resource.



Gradient Resources in the Library

Note: If gradient settings match an existing library resources that resource will be used (as copy or link).

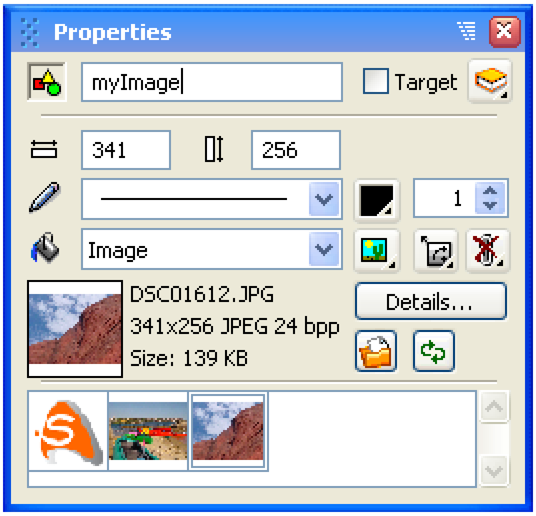
Gradient Slider



The gradient slider control shows you the gradient fill across the shape. For an explanation of the bounds of the gradient this represents refer to the section on the Fill Transform tool. Click to place new colors onto the gradient. Click on a triangular colored handle to change the corresponding color using the Color Picker. Drag the handles along the gradient to adjust their positions. Hold the ***shift key*** while dragging to snap the handle to the grid marks provided. Drag the triangular colored handles completely off the bar to delete them.

6.3.5.4.3 Shape Object - Image Fill

This section discusses Properties Panel settings for the Shape Object with an Image fill. Refer to Shape Object Properties Panel for details on settings common to all Shape Object Panels.



Properties

myImage □ Target

341 256

1

Image

DSC01612.JPG
341x256 JPEG 24 bpp
Size: 139 KB

File Style - 'Image'

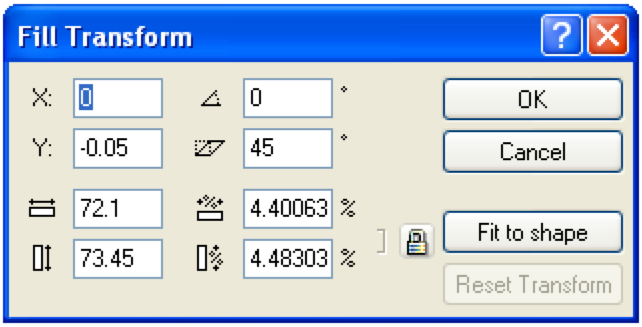
- **Single image:** uses a bitmap image to fill the shape. If the image is too small to fill the shape completely, then the outside pixels are stretched. Click on the Fill Control to select an image
- **Tiled image:** uses a bitmap image to fill the shape. If the image is too small to fill the shape completely, then the image is tiled. Click on the Fill Control to select an image

Fill Transform

Defines how the gradient fills the shape. Options are:

- **Fit to Shape** (*Single Image only*): always fit the shape while reshaping
- **Reset Transform:** resets the transform
- **Edit Fill Transform..:** presents a dialog for custom setup

The 'Fill Transform' dialog box is represented below.



Fill Transform

X: 0 | 0 °

Y: -0.05 | 45 °

72.1 | 4.40063 %

73.45 | 4.48303 %

The controls are the same as for the Transform Panel. However, in this case, the Transform applies to the fill pattern within the shape, rather than the shape within the Scene. To visually edit the fill Transform use the Fill Transform tool.

Add to Library

Add the Object to the library. See also Library. If there are existing Image resources (shown lower in the dialog), these can be linked to the current Object. Select the Library item and click on the link icon ( ) to link the Objects fill to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object as a copy of the library Resource.

This button opens a dialog to select an image.

Reload Image

Reloads the current content from an external file.

Details...

Image details

Edits the Image properties. See separate sections for a description of these options:

- **Export**: export settings for the image
- **Appearance**: edit the brightness, contrast, hue, saturation, luminance and gamma of the image
- **Effects**: apply a range of effects (e.g. blur, emboss) to the image
- **Crop**: change the displayed area of the image.

6.3.5.4.3.1 Export Image Properties

This section discusses Image Properties dialog settings for the Image Shape Object.

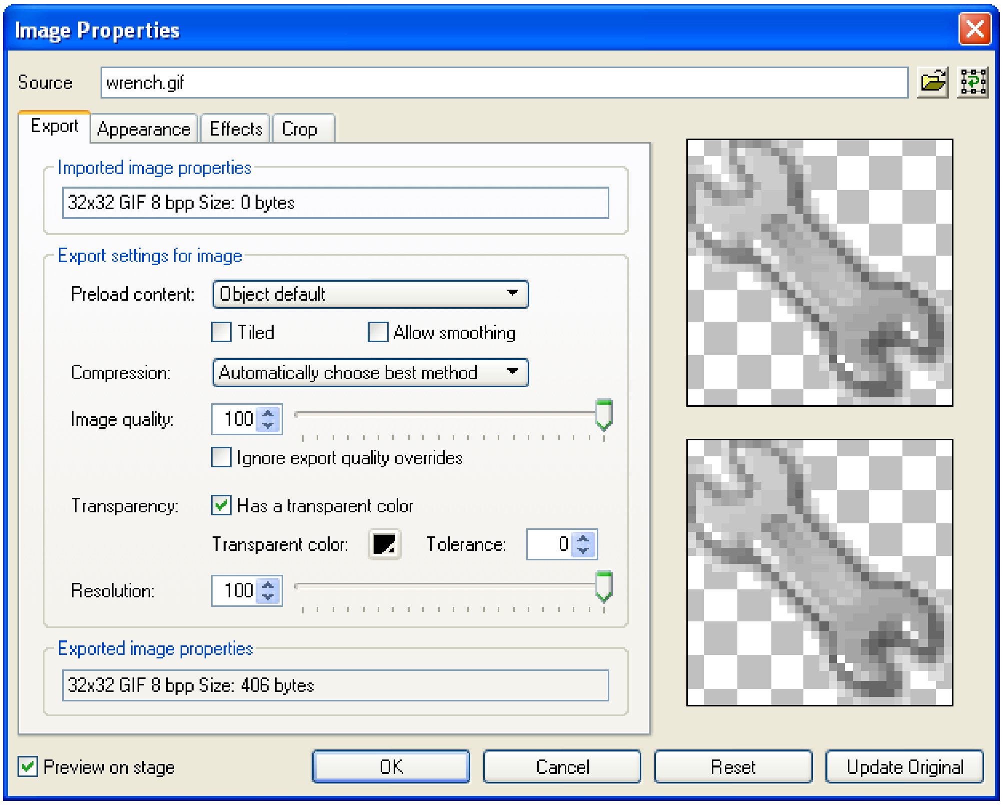


Image Properties

Source: wrench.gif

Imported image properties: 32x32 GIF 8 bpp Size: 0 bytes

Export settings for image

Preload content: Object default

□ Tiled □ Allow smoothing

Compression: Automatically choose best method

Image quality: 100

□ Ignore export quality overrides

Transparency: ☑ Has a transparent color

Transparent color: ■ Tolerance: 0

Resolution: 100

Exported image properties: 32x32 GIF 8 bpp Size: 406 bytes

☑ Preview on stage

Source
Path and filename of the source image file. Shown here as 'wrench.gif'.

Open image file
Opens a file open dialog to select and replace the current image file.

Reload image
Reloads the current image file from the original file. This is used when the file is edited outside of SWiSH Max.

Preview on stage
Changes to the image are shown in the image displayed in the Layout Panel (stage) of SWiSH Max.

Reset
Clears all changes to the image from any of the tabs settings (export, appearance, effects or crop).

Update Original
Applies changes to the original file. This changes the original files irreversibly.

Imported Image Properties Export / Image Properties
States the dimensions (in pixels), compression (bits per pixel i.e. bpp) and size of the image file imported and exported. A preview of the original and exported image is also provided. The export preview shows the visible changes selected in the Image Properties tab.

Export settings for Image

Preload content

The 'Preload content' option defines where the image definition appears in the .swf file. Using appropriate pre-loading of images will avoid delays or jerkiness while the Movie is playing. The Preload content options are as follows:

- **Disabled**: definitions are written out at the Frame where the object is first placed. If the definition is large because there is a large object or block of text, or if the connection speed is slow, this could introduce a short delay or jerk in the playback
- **Before Scene**: definitions are grouped together at the start of the Scene. This may mean a short delay before the Scene starts playing, but such delays are generally less noticeable than those that occur while the Scene is playing
- **Before Movie**: definitions are grouped together before the first Frame of the Movie. This can mean a delay before the Movie begins to play
- **At Preload Frame**: adds a Preload Content Action as a Scene Event. This acts as a marker for where definitions are to appear. Any objects before the first Preload Content action are treated as though Preload content was disabled
- **Object default**: the value for this setting is taken from the default defined for the Object (or Scene), and is not overridden here. This is the default and unless the Scene and Movie settings are changed the effective value will be the same as the At Preload Action setting

Tiled

Uses a bitmap image to fill the shape. If the image is too small to fill the shape completely, then the image is tiled. See the same setting in Image Shape Object.

Allow Smoothing

If Allow Smoothing is check an image that has been resized will be smoothed. Smoothing can blur images so in some cases this may not be advisable.

Compression

The compression options used for the exported image:

- **Automatically choose best method (default)**: SWiSH Max sets the mode that gives the smallest compressed file size
- **As imported**: don't use any compression. The image exported is as close to the imported image as possible

Note: Use the existing compression of your imported image, or have SWiSH Max recompress the image itself to the quality level you select. Some images must be recompressed because they use progressive JPEG encoding, which will crash the Flash Player if not recompressed, or because the original file is not JPEG compressed.

- **Force 8 bpp if smaller / Force 16 bpp if smaller**:
- **Smaller file size**: optimizes compression to create the smallest file size
- **Force 8 bpp lossless (GIF) / Force 16 bpp lossless (PNG) / Force 32 bpp lossless (PNG)**

Uses 'lossless' ZIP compression, which means compression does not change the image quality or alter the image content. This is the same compression
used for .zip files. The options offers different bits per pixel (bpp).

'Force 32 bpp lossless' will always keep the image as is. It keeps the RGB values for a color in 1 byte each. 'Force 16 bpp lossless' rounds the RGB values up to the nearest multiple of 8 (except for 0 values, that stay zero), so the combined RGB values can fit into two bytes (16-bits). That means the colors in the image may change, if they are not already the correct colors.

Note: 16 bpp lossless mode does not have an alpha value, and so cannot be used for images with transparency.

'Force 8 bpp lossless' keeps the colors as in the 32 bpp mode, but limits total number of different colors to 256. If there are more than 256 colors it will map less used colors to the nearest more used color consequently the colors in the image will change if there are more than 256 distinct colors in the image.
The 'Automatically choose best method' option sets the mode that gives the smallest compressed file size. However, if the 16 bpp or 8 bpp modes would result in a change to the colors in the image, then SWiSH Max will not consider those modes.
The 'Force 8 bpp if smaller' compression option will force SWiSH Max to consider 8 bpp mode, even if it means the colors will change.
The 'Force 16 bpp if smaller' compression option will force SWiSH Max to consider 16 bpp mode, even if it means the colors will change.

The 'Smallest file size' compression setting is the combination of the two modes above, and will consider all three modes, regardless of changes to color, when finding the smallest compressed file size

Force 24/32 bpp lossy (JPEG)
JPEG compression is 'lossy' compression, which means image quality is reduced during compression. You can control the compression indirectly using the Image Quality setting. This is the same compression that used by JPEG files.

Image quality slider
The higher the value, the better the image quality, the larger the files size and the less compression. The quality is not directly related to the compression ratio. If you imported a JPEG image and specify a quality lower than the original quality, or if you change any of the "Adjust Appearance" options, or if the image is progressive JPEG, then SWiSH Max will automatically recompress, otherwise it will use the imported JPEG data.

Ignore export quality overrides
Selecting the option 'Ignore export quality overrides' ignores the settings on the Object's Export Panel and will use only the Image Properties settings for the image.

Transparency

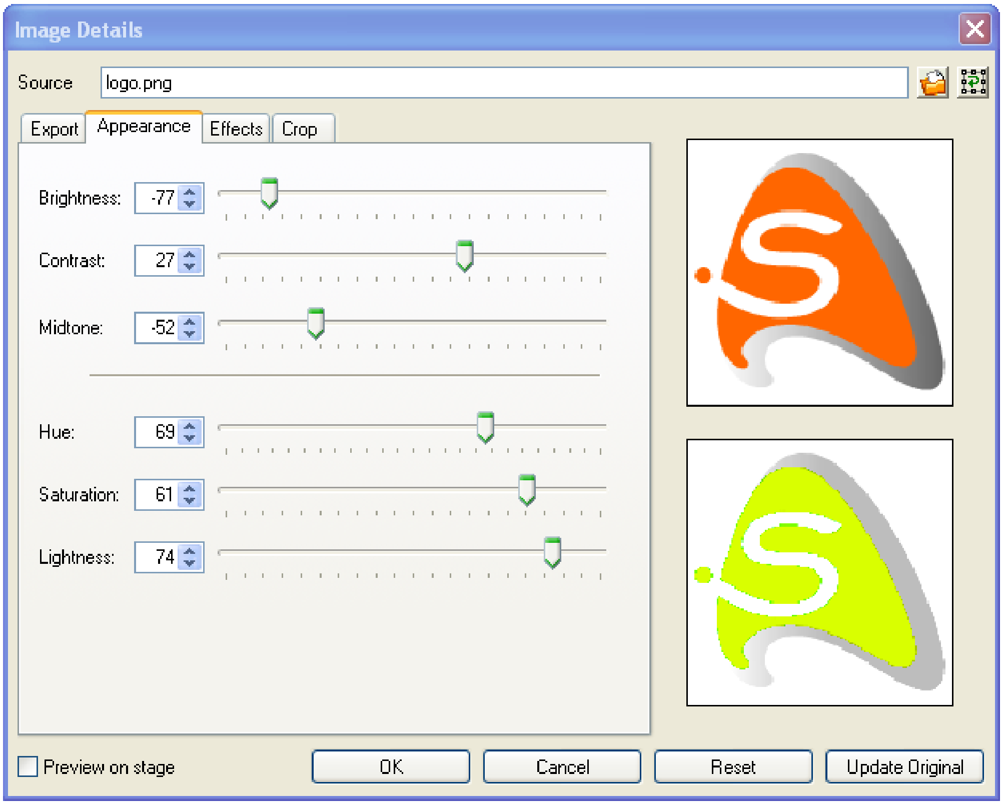
Has a transparent color
The 'Transparency' checkbox is only available for opaque images.

Transparent color (■) / Tolerance
When the 'Transparency' setting is checked, a specific color (chosen from the Color Picker) from the image can be made transparent. The 'Tolerance' setting controls how similar to the specified color the pixel must be before it is changed to transparent. The valid tolerance setting is between 0 and 255. At low settings, the colors must be close. At higher settings the tolerance is greater, so more pixels are made transparent

Resolution
The Resolution setting adjusts the images pixel resolution to a percentage of the original resolution.

6.3.5.4.3.2 Appearance Image Properties

This section discusses Image Properties dialog settings for the Image Shape Object.



Source: logo.png

Brightness: -77

Contrast: 27

Midtone: -52

Hue: 69

Saturation: 61

Lightness: 74

□ Preview on stage

Source, Open, Reload, Preview on stage, Reset and Update Original are described in the 'Export Image Properties' section.

Brightness (-100% to 100%)
The Brightness control brightens or darkens the image. Increasing Brightness makes the blacks brighter but doesn't change the whites whilst decreasing brightness makes the whites darker, but doesn't change the blacks.

Contrast (-100% to 100%)
The Contrast control adjusts the difference between light and dark in an image. For example, a contrast of -100 will make the image all grey. A contrast of +100 will make it all black, white, or basic colors (red, yellow, green, cyan, blue, pink).

Midtone (-100% to 100%)
The midtone is the grey tone in the image which lies midway between the highlight and the shadow tones. Increasing the midtone level makes all colors brighter, increases the contrast for darker colors and decreases contrast for lighter colors. The opposite applies for for decreasing the midtone.

Hue (-180 degrees to 180 degrees)
The Hue control adjusts the picture colors across the scale. For example a hue shift of 128 will turn red into green, green into blue, and blue into red. Changing the hue can add a tint to the whole image.

Saturation (-100% to 100%)
The Saturation control adjusts the color depth of the image. For example a saturation of -100 will make the

image grey scale.

Lightness (-100% to 100%)
The Lightness adjusts mid-tone brightness. For example a larger positive value will lighten the darker mid-tones. A negative value will darken the lighter mid-tones.

Note: SWiSH Max uses the HSV model for color space. LIghtness is 'value' in this model.

The following sections discuss Brightness, Contrast and Midtone controls in more detail with examples.

Brightness

When increased, light colors stay light, but dark colors get lighter by the percentage of white specified. Midtones get proportionally brighter. As a result, the contrast decreases, and overall brightness increases.

For example when brightness is at +50%, Black #000000 becomes Grey #808080, Dark Grey #404040 becomes Mid Dark Grey #A0A0A0, Grey #808080 becomes Light Grey #C0C0C0, Light Grey #C0C0C0 becomes Lighter Grey #E0E0E0, White #FFFFFF is unchanged.

When decreased, dark colors stay dark, but light colors get darker by the percentage of white specified. Midtones get proportionally darker. As a result, the contrast decreases, and overall brightness decreases.

For example when brightness is at -50%, Black #000000 in unchanged, Dark Grey #404040 becomes Darker Grey #202020, Grey #808080 becomes Dark Grey #404040, Light Grey #C0C0C0 becomes Mid Dark Grey #606060, White #FFFFFF becomes Grey #808080.

Contrast

When increased, light colors get lighter and dark colors get darker by the percentage specified. Midtones are relatively unchanged. As a result, the overall brightness is relatively unchanged.

For example when contrast at +50%, Black #000000 is unchanged, Dark Grey #404040 becomes Black #000000, Grey #808080 is unchanged, Light Grey #808080 becomes White #FFFFFF, White #FFFFFF is unchanged.

When decreased, light colors get darker and dark colors get lighter by the percentage specified. Midtones are relatively unchanged. As a result, the overall brightness is relatively unchanged.

For example when contrast is at -50%, Black #000000 becomes Dark Grey #404040, Dark Grey #404040 becomes Mid Dark Grey #606060, Grey #808080 is unchanged, Light Grey #C0C0C0 becomes Mid Light Grey #A0A0A0, White #FFFFFF becomes Light Grey #C0C0C0.

Midtone

When increased, midtones get lighter by the percentage of white specified. Darker and lighter colors are relatively unchanged. As a result, the contrast for lighter colors decreases, the contrast for darker colors increases, and overall brightness increases.

For example when midtone is at +50%, Black #000000 is unchanged, Dark Grey #404040 becomes Mid Dark Grey #606060, Grey #808080 becomes Light Grey #C0C0C0, Light Grey #808080 becomes Lighter Grey #E0E0E0, White #FFFFFF is unchanged.

When decreased, midtones get darker by the percentage of white specified. Darker and lighter colors are relatively unchanged. As a result, the contrast for lighter colors increases, the contrast for darker colors decreases, and overall brightness decreases.

For example when contrast is at -50%, Black #000000 is unchanged, Dark Grey #404040 becomes Darker Grey #202020, Grey #808080 becomes Dark Grey #404040, Light Grey #C0C0C0 becomes Mid Light Grey #A0A0A0, White #FFFFFF is unchanged

6.3.5.4.3.3 Effects Image Properties

This section discusses Image Properties dialog settings for the Image Shape Object.

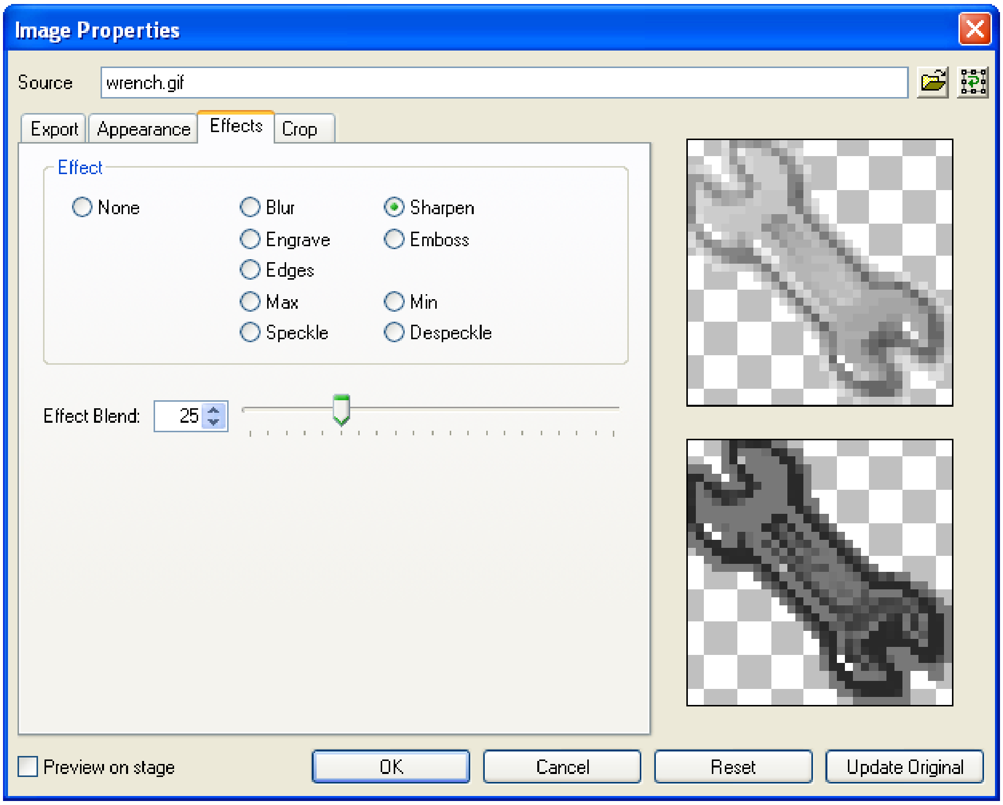


Image Properties

Source: wrench.gif

Effect: ● Sharpen

Effect Blend: 25

□ Preview on stage

Source, Open, Reload, Preview on stage, Reset and Update Original are described in the 'Export Image Properties' section.

Effect
Applies the selected effect to the % degree set by the Effect Blend slider (or numeric stepper entry). For example selecting Blur and Effect Blend to 50 sets a 50% blur to the image.

None: Image is exported unchanged.
Effects: Blur, Sharpen, Engrave, Emboss, Edges, Max, Min, Speckle, Despeckle.

6.3.5.4.3.4 Crop Image Properties

This section discusses Image Properties dialog settings for the Image Shape Object.

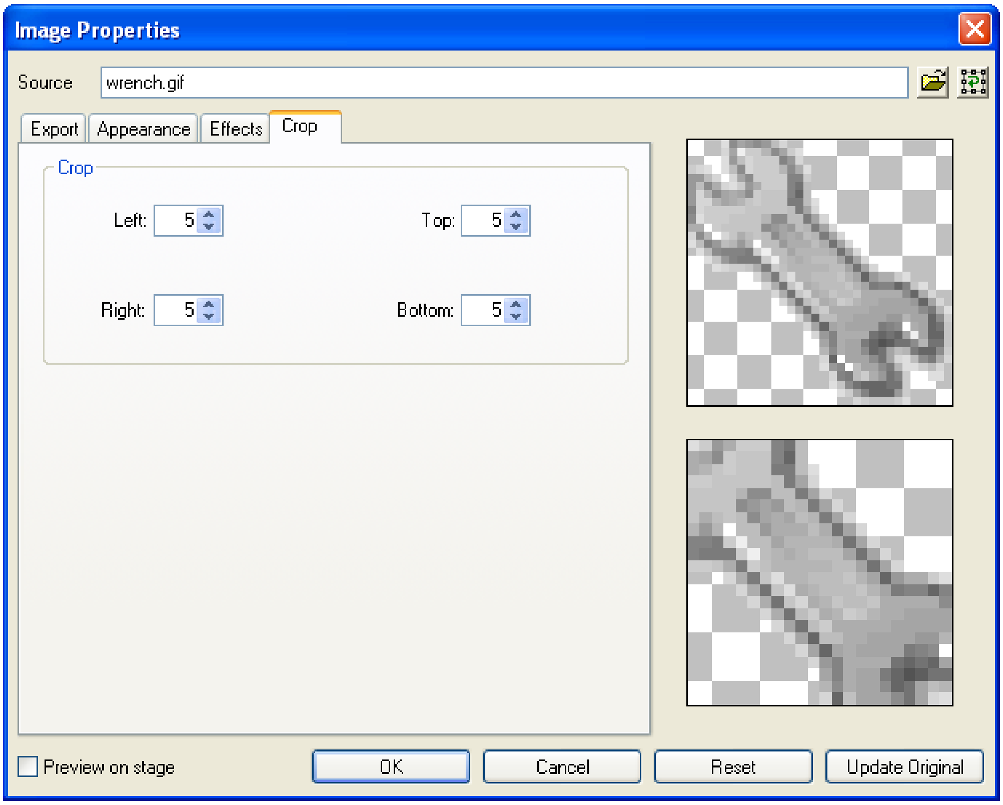


Image Properties

Source: wrench.gif

Export | Appearance | Effects | Crop

Crop

Left: 5

Top: 5

Right: 5

Bottom: 5

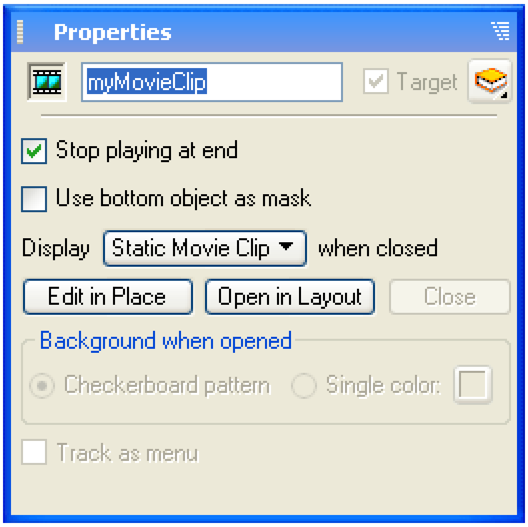
□ Preview on stage

Source, Open, Reload, Preview on stage, Reset and Update Original are described in the 'Export Image Properties' section.

Crop (Left / Right / Top / Bottom)
Crops the image. Only the cropped section is exported. The Crop offset is set in Pixels.

6.3.5.5 Movie Clip Object Properties Panel

The Properties Panel for a Movie Clip Object shows the properties for the selected Movie Clip.



Properties

myMovieClip ☑ Target

☑ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip when closed

Background when opened

● Checkerboard pattern ○ Single color: □

□ Track as menu

See more information about Movie Clips in the section on Movie Clip Object.

Name *(shown above as "myMovieClip")*
The name of the Movie Clip appears in the Outline Panel. It can also be used to identify the Movie Clip in various scripting commands such as isNearTarget(), nextFrameAndPlay() etc.

Target
The Target option is grey (and unselectable) because it is implied that a Movie Clip is always a target. Target is selectable for Text Objects and Shape Objects which need not be targets. The object to be treated as a script object.

Add to Library
Adds the object to the library or links to an existing Library Object. See more detail at Library.

Stop playing at end
Automatically stops the Movie Clip Movie at the end. By default the Movie Clip will play continuously as long as it is placed. If this option is checked it will play once and stop at the last Frame.

Use bottom Object as mask
When checked, the object in the bottom-most layer of the Movie Clip becomes the mask shape. The rest of the Movie Clip content will only be visible where the mask shape is filled. Any Movie Clip content outside the mask shape is invisible. The default setting is to have the masking option turned off.

Note: The masking shape must be a single shape or text object. The masking shape cannot be a Scripting Object. Masking will not work if these conditions are not met. If Scripting is added to the mask object, it will convert that object to a Scripting Object and the mask will no longer work.

The Movie Clip has the following 'closed' state display options:

- **Static Movie Clip**: Displays all objects in the Movie Clip even objects which are not shown at the 1st frame
- **Movie Clip Icon**: the Movie Clip icon is displayed for the Movie Clip instead of its contents, unless the Movie is playing.
- **First frame**: only the child objects shown in the 1st frame of the Movie Clip are displayed, unless the Movie Clip is expanded or the movie is played.

Edit in place

This option will open the timeline of this Movie Clip to edit the objects in place and actions placed inside of it. This is also the same as clicking the (+) icon next to the Movie Clip on the 'Outline' panel. Movie Clips open for editing in place will have a border shown around it.

Open in Layout
When pressed, this option will open the Movie Clip for editing with it's own background. This is different than editing in place in that the objects outside the Movie Clip are not visible while editing. Use the Close Movie Clip button in the Timeline panel to close the editor.

Close
Closes editing modes 'Edit in place' or 'Open in layout'.

Background *(only available whilst in 'Open in Layout' edit mode)*
Sets the background image of the Movie Clip to appear when you expand it for editing using 'Edit in Layout'. The background highlights the difference with editing the contents of a Movie Clip, rather than the content of a main Movie Scene. The settings do *not* affect the appearance of the Movie Clip when it is not being edited. The default setting for this function is a single color. Options are:

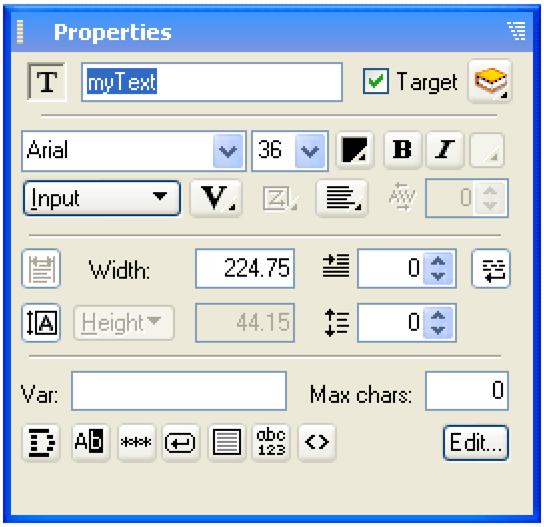
- **Checkerboard pattern**
- **Single color (■)**

Change the background color for the Movie Clip using the Color Picker. If the alpha value is less than 100%, it is possible to tint the Movie background color. If the alpha value is 0% (the default), the Movie Clip will have the same color as the Movie default.

Track as menu item
Sets whether or not the Movie Clip can receive mouse release events. This option is typically used when using Movie Clips as menu items. See Button Object Panel for more details on this function.

6.3.5.6 Text Object Properties Panel

The Properties Panel for a Text Object shows the properties for the selected Text.



Properties

T myText ☑ Target

Arial 36 **B** *I*

Input

Width: 224.75 0

Height 44.15 0

Var: Max chars: 0

Edit...

See more information about text in the sections on the Text Tool, Insert Text and the Text Object.

Note: It is now possible to enter text with different Font, Size and Color into the same text object. Consequently, the selections for Font, Size, Color etc only apply to the currently selected text.

Name *(shown here as "myText")*
The name of the Text as also shown in the Outline Panel. If the object is set as a Target the name is the script reference. If no name is specified, the outline panel will display "*Text*".

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

Add to Library
Adds the object to the library or links to an existing Library Object. See more detail at Library.

Font Selection *(shown above as "Arial")*
Selects the font for the currently selected text within the Text Object. Only TrueType and Postscript fonts can be used with SWiSH Max. The pull-down Menu lists all the TrueType and Postscript fonts installed on the computer. When the pull-down list is activated, a window appears next to the current selection showing samples of the font.

Font Size *(shown above as "36")*
Selects the size of the font (in points) for the currently selected text within the Text Object. Select the size from the drop-down list of preset sizes, or type the size into the available space.

Text Color button
Selects the color of the currently selected text within the Text Object using the Color Picker. Select both color and transparency value (alpha). The default text color is black.

Bold and **Italic Buttons**
Lets you set the style of the currently selected text to plain, bold, italic or a combination of both. The default settings are for plain text.

Insert Character
Select a character from the font map. This menu shows all characters available to the selected font.

Text Object Type
A Text Object can be Static, Dynamic, or Input. See more details in the section on Text Objects. The options available in the Text Object Properties Panel will vary depending on which text object type is chosen and which font typeface is selected.

Typeface Options
Select a type face for displaying the text. The available list box options are below with the exception of 'Use Device Fonts' which is included here for completeness but it in a separate button:

- **Use Vector Fonts:** Displays text in normal vector format.
- **Vector Font (pixel aligned):** This option forces text to be top-left aligned to a whole pixel. Turn it on when using a pixel font (such as miniml) to ensure the text is properly aligned. This means it is not necessary to manually set the text to whole pixels or to change the text alignment.
- **Pixel Fonts (sharp):** This option will convert any vector font into a pixel-based font. Each part of a pixel font is aligned to exact pixels on the monitor which gives it a crisp appearance even at smaller sizes as opposed to vector fonts which can appear blurred at small sizes.
- **Pixel Fonts (smooth):** Same as sharp pixel fonts however this option will add anti-aliasing on the corners and diagonal lines making it look more smooth than sharp pixel fonts.
- **Use Device Fonts** *(whole object only)*: Specifies that the Flash Player uses device fonts to display

the text. Using device fonts means no fonts will be embedded into the Movie, instead the Flash Player will use the font on the local machine that is the closest match. Using device fonts can decrease the Movie size and increase readability for text under 10 points *(see the note below for the limitations of using Device Fonts).*

Note:

- Device font can only apply to the whole object and not per character. Other typefaces apply per character.
- Due to a Flash Player limitation, Device Fonts cannot be scaled, rotated, masked, or have an alpha value of less than 100% (opaque). When receiving this error message, make sure to choose Vector Fonts instead of Device Fonts (change the "D" icon to "V" or "P"), and do not use the fonts: _sans; _serif; or, _typewriter. Check to make sure the fonts are at 100% scale on the Transform panel (both vertical and horizontal scale). Do not apply any complex effects to Device Fonts (eg, explode, snake, vortex, etc). See Help / Frequently Asked Questions for more information.

Text Flow

Lets you select the direction of the text flow. The flow can only be defined for Static Text. The available directions are:

- from left to right, top to bottom. This is the default setting.
- from top to bottom, right to left
- from right to left, top to bottom
- from top to bottom, left to right

Justification

Lets you set the text justification for the text object. The available justification are:

- **Left:** Aligns the text to the left side.
- **Center:** Aligns all of the text to the center.
- **Right:** Aligns the text to the right side.
- **Full:** Aligns all lines of text except lines that have been wrapped to the right and left sides. It will add spaces between words to spread them out *(this option is only available for Static Text).*
- **Full (all lines):** This aligns all lines to both the left and right sides - It will add spaces between individual words to spread them out *(this option is only available for Static text).*
- **Full (all except bottom line):** Aligns all lines except the bottom line to both the right and left side *(this option is only available for Static text).*
- These justification options are the same as above; however, these options are only available when the text flow is changed to 'Top to Bottom' *(full justification options are only available for Static Text).*

Auto Kerning / Kerning numeric entry % 0

When selected the Auto Kerning option uses the kerning settings in the font in use. Kerning between individual characters can be altered by selecting those characters and changing the % kerning. The Kerning control adjusts the horizontal distance between characters in a word. A negative number will move the characters closer together, while a positive number will move the characters further apart. A value of zero (the default setting) resets the kerning to the standard distances defined by the font.

Enable Margins and Indent *(always enabled for Dynamic and Input Text)*
Displays the margin and indent controls. See the section on Text Tool for a description of the cursors. This option is fixed as enabled for Dynamic and Input Text. When selected you can set the following options:

Width: 237.9

Width
Sets the sizes of the right margin for the Text Object.

5

Indent
Sets the amount to indent the first line of the Text Object.

Wrap Text at Word Breaks *(not available for Static Text)*
Wraps lines of text at the end of the text field, otherwise text is only wrapped at carriage returns.

Auto-Size Height *(not available for Static Text)*
When selected defines the height of the text field to be the size of the text. When unselected you can define the height of the text field using the Height/Lines selection.

Height ▾ 44

Height/Lines
When Height is selected you can define the height of the text field to be a set number of pixels. When Lines is selected you can define the height of the text field to be a set number of lines.

21

Leading
The Leading control adjusts the vertical distance between lines in a multi-line Text Object. A negative number will move the lines closer together, while a positive number will move the lines further apart. A value of zero (the default setting) resets the leading to the standard distance defined by the font.

Var:

Variable *(not available for Static Text)*
Variable whose value appears in the Text Object. This is only available for named and targeted Text Objects. If the dynamic/input text field is a target, then setting this option uses a variable to access the text field contents, otherwise use the ".text" property of the text field. If this option is set the variable specified MUST have a name different to the name of the text field itself. If the dynamic/input text field is NOT a target, then it is automatically associated with a variable of the same name (eg. for a simple text field called 'myfield' use the variable also called 'myfield' to access the text field contents via the scripting language, see Text Field Object for more information on scripting).

Max chars: 20

Maximum characters *(only available for Input Text)*
The limit on how many characters can be typed into the Input Text Object.

Use Device Fonts
See Font typeface descriptions above.

Text is Selectable
The user can select the text in the text field.

Password Text
The * character is displayed for each text character, useful for password input fields.

Multi-line Text

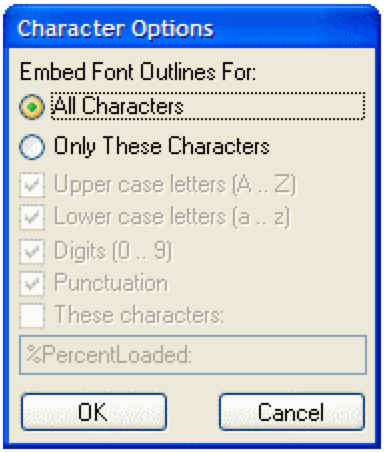
Allows users to use the enter key for newlines when entering text in an input text field.

Black Border with White Background

Shows the text field with a black border and a white background.

Character Options

When using Dynamic or Input text, this will open the character embedding options.



Character Options

Embed Font Outlines For:

● All Characters

○ Only These Characters

☑ Upper case letters (A .. Z)

☑ Lower case letters (a .. z)

☑ Digits (0 .. 9)

☑ Punctuation

□ These characters:

%PercentLoaded:

Defines which characters will be embedded in the Movie:

- **All characters** (ASCII): All character embedded.
- **Only these characters**: A defined set of characters will be embedded.
  - **Upper case letters (A..Z)**: Embed all upper case letters
  - **Lower case letters (a..z)**: Embed all lower case letters
  - **Digits (0..9)**: Embed all digits
  - **Punctuation**: Embed all punctuation characters
  - **These characters**: The inputted set of characters will be embedded (Note: No spaces or commas are required between the character list)

Render Text as HTML

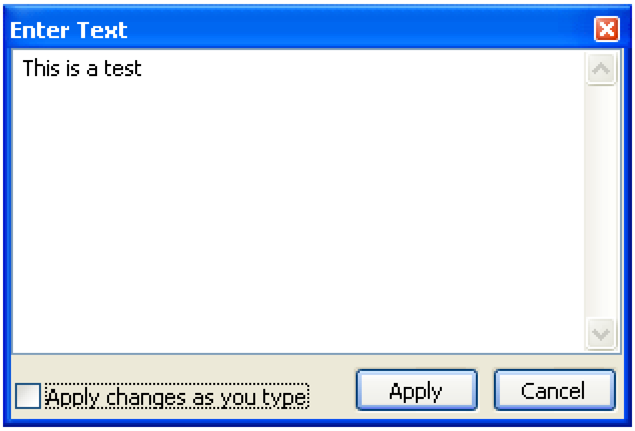
Preserves HTML text formatting options such as size and hyperlinks for input and dynamic text fields. HTML text formatting can only be seen externally from SWiSH Max, for example using File | Test | SWF in Player. The HTML tags <A>, <B>, <FONT COLOR>, <FONT FACE>, <FONT SIZE>, <I>, <P>, and <U>. The HTML attributes supported are LEFTMARGIN, RIGHTMARGIN, ALIGN, INDENT, and LEADING.

Edit... Edit Text dialog

Opens a separate dialog for the entry of text.

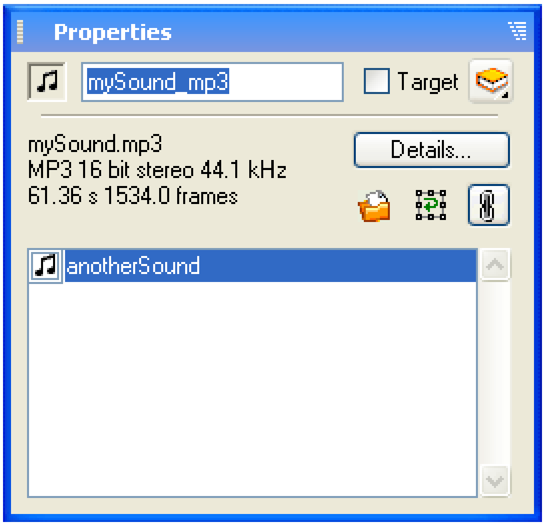
Apply changes as you type

Applies changes to text in the dialog directly to the Text Object on the stage (Layout Panel). Formatting of text in this dialog is not possible. If this checkbox is not checked, pressing the Apply button updates the text on the stage.



6.3.5.7 Sound Object Properties Panel

The Properties Panel for a Sound Object shows the properties for the selected Sound Object.



mySound.mp3
MP3 16 bit stereo 44.1 kHz
61.36 s 1534.0 frames

Name *(shown above as "mySound_mp3")*
The name of the sound as also shown in the Outline Panel. The Name is based on the filename. If the object is set as a Target the name is the script reference.

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

Add to Library
Add the object to the library or links to an existing library object. See more detail at Library.

Description
This will display the frequency, bitrate, and length (in total seconds and frames) for the selected audio file.

'Details...' button
Opens a properties dialog for the sound.

Load Sound
This button opens a dialog to select a sound.

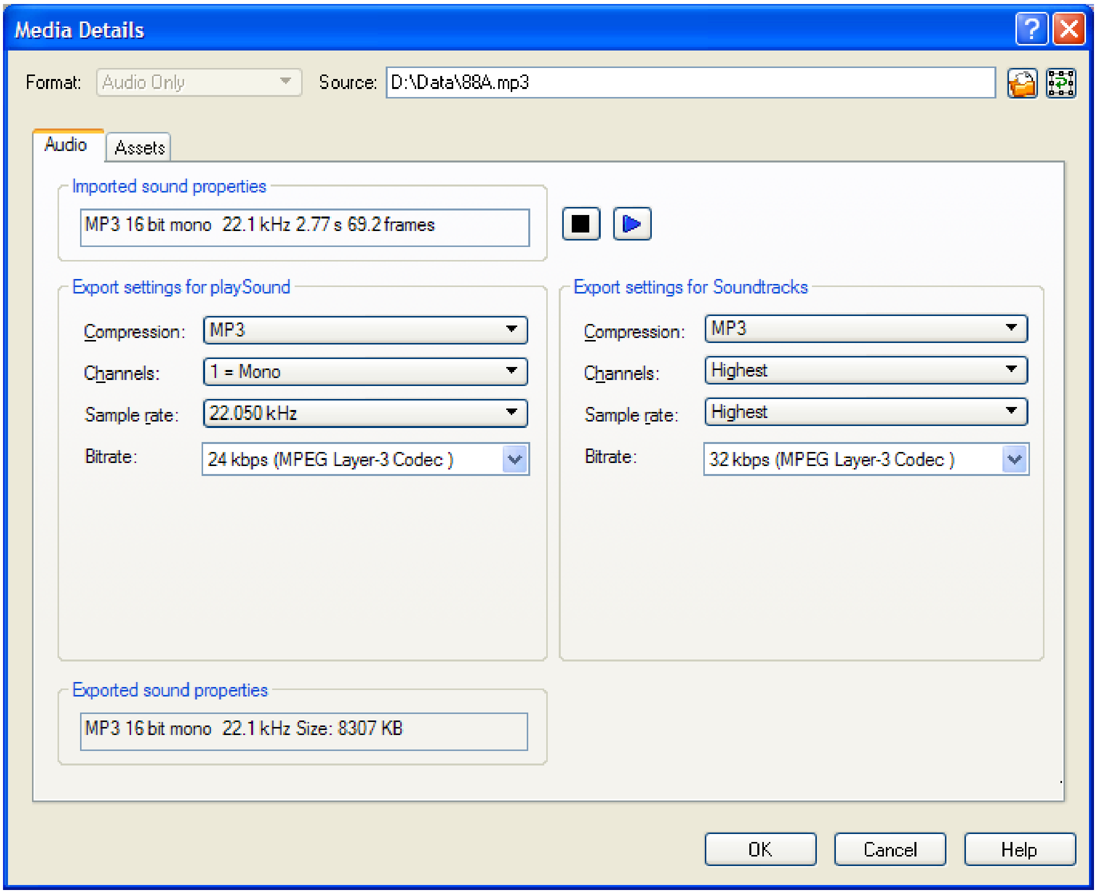
Reload Sound
Reloads the current content from an external file.

Add to Library
Add the Object to the library. See also Library. If there are existing Sound resources (shown lower in the dialog), these can be linked to the existing Object. Select the Library item and click on the link icon ( ) to link the Object to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object as a copy of the library Resource.

6.3.5.7.1 Audio Properties

This section discusses the Media Details dialog settings for the Sound Object.

The Media Details dialog has an Assets tab which allows the sound to be treated as a shared asset and the Audio tab.



Format: Audio Only
Source: D:\Data\88A.mp3

Imported sound properties
MP3 16 bit mono 22.1 kHz 2.77 s 69.2 frames

Export settings for playSound
Compression: MP3
Channels: 1 = Mono
Sample rate: 22.050 kHz
Bitrate: 24 kbps (MPEG Layer-3 Codec)

Export settings for Soundtracks
Compression: MP3
Channels: Highest
Sample rate: Highest
Bitrate: 32 kbps (MPEG Layer-3 Codec)

Exported sound properties
MP3 16 bit mono 22.1 kHz Size: 8307 KB

Audio tab options include:

Source *(shown above as "mySound.mp3")*
Path and Filename for the source sound.

Load Sound
This button opens a dialog to select a sound.

Reload Sound
Reloads the current content from an external file.

Imported Sound Properties
States the format, channels, frequency, duration and frames of the sound file imported.

▶ ⏸ **Play / Pause (Toggle) Sound**
Plays the imported sound.

■ **Stop Sound**
Stops the sound playing.

Export Settings for playSound / Export Settings for Soundtrack
Export setting which apply to the sound if it is used as either a playSound() event or Soundtrack. See a description of Soundtracks and Event sounds at 'Import Sound...'. The Soundtrack setting applies to all soundtracks and can also be set at 'File | Export Settings | Soundtrack'.

- **Compression**: Available sound compression options (see details on ADPCM and MP3 below):
  - **None**
  - **ADPCM**:
  - **MP3**: Preferred.

- **Channels**: Set the channels the sound uses:
  - **Lowest** *(only available for Soundtrack):* Chooses the lowest setting among all soundtracks within the Movie.
  - **1 = Mono**:
  - **2 = Stereo**:
  - **Highest** *(only available for Soundtrack):* Chooses the highest setting among all soundtracks within the Movie.

- **Sample rate**: Set the output sample rate:
  - **Lowest** *(only available for Soundtrack):* Chooses the lowest setting among all soundtracks within the Movie.
  - **5.5125 kHz**:
  - **11.025 kHz**:
  - **22.050 kHz**:
  - **44.100 kHz**: *(see comment below)*
  - **Highest** *(only available for Soundtrack):* Chooses the highest setting among all soundtracks within the Movie.

- **Bitrate** *(available for MP3 only)*: Available bitrates depend on the MP3 Codec installed.

- **Samples Size** *(available for ADPCM only)*: Choose the sample size from 2 to 5 bits/sample.

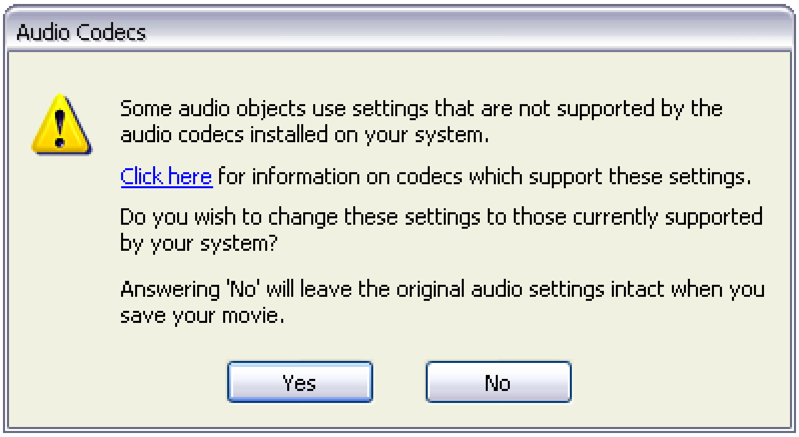
Note on Compression types ADPCM vs. MP3:

ADPCM (Adaptive Differential Pulse Code Modulation) is a family of audio compression and decompression algorithms. It is a simple but efficient compression scheme that avoids any licensing or patent issues that arise with more sophisticated sound compression schemes, and helps to keep player implementations small. For SWF files of version 4 or later, MP3 Compression is a preferable format. MP3 produces substantially better sound for a given bitrate. ADPCM uses a modified Differential Pulse Code Modulation (DPCM) sampling technique where the encoding of a each sample is derived by calculating a 'difference' (DPCM) value, and applying to this a complex formula which includes the previous quantization value. The result is a compressed code, which can recreate almost the same subjective audio quality.

MP3 is a sophisticated and complex audio compression algorithm supported in SWF 4 and later. It produces superior audio quality at better compression ratios than ADPCM. Generally speaking, MP3 refers to MPEG1 Layer 3; however, SWF supports later versions of MPEG (V2 and 2.5) that were designed to support lower bitrates. Flash Player supports both CBR (constant bitrate) and VBR (variable bitrate) MP3 encoding. *Source*.

.

Note on 44.1kHz encoding
SWiSH Max use the Windows built-in ACM driver to encode MP3. However this codec only supports audio less than 44.1kHz. For encoding at 44.1kHz a different MP3 encoder is required. If a suitable codec is not found on your PC, SWiSH Max will prompt with a link to SWiSHzone.com offering a recommendation for a suitable codec.



Audio Codecs

Some audio objects use settings that are not supported by the audio codecs installed on your system.

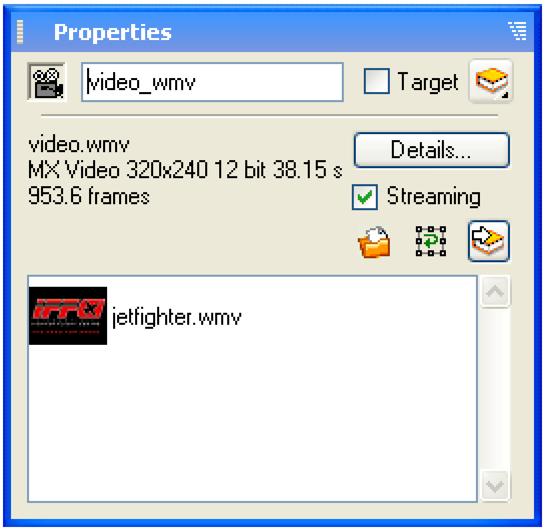
Click here for information on codecs which support these settings.

Do you wish to change these settings to those currently supported by your system?

Answering 'No' will leave the original audio settings intact when you save your movie.

6.3.5.8 Embedded Video Properties Panel

The Properties Panel for an Embedded Video Object shows the properties for the selected Embedded Video.



Properties

video_wmv □ Target

video.wmv
MX Video 320x240 12 bit 38.15 s
953.6 frames

Details...

☑ Streaming

jetfighter.wmv

Name *(shown above as "video_wmv")*
The name of the embedded video as also shown in the Outline Panel. The Name is based on the filename. If the object is set as a Target the name is the script reference.

Target
If set, allows the object to be treated as a script object. The target option is only displayed if the object is named.

Add to Library
Add the object to the library or links to an existing library object. See more detail at Library.

'Details...' button
Opens a properties dialog for the media.

Streaming
When the Streaming option ***is check*** the video frames are placed consecutively (one video frame for each timeline frame) in the main timeline starting from the frame set by the 'Preload content' option in 'File | Export Settings | SWF'. However SWiSH Max also ensures all video frames are placed BEFORE the frame where the Movie Clip is placed so the last frame may contain more than one video frame. The result is the Movie doesn't have to slow down waiting for the entire Movie Clip (including the embedded video) to load.

When the Streaming option ***is not checked***, all video frames will be placed into one frame set by the 'Preload content' option in 'File | Export Settings | SWF'.

Load Video
This button opens a dialog to select a video.

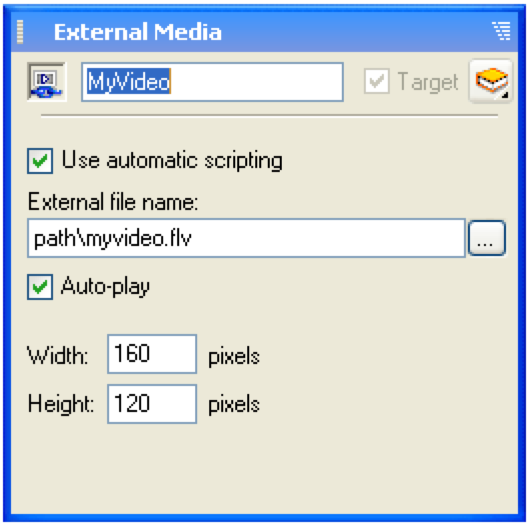
Reload Video
Reloads the current content from an external file.

Add to Library
Add the Object to the library. See also Library. If there are existing video resources (shown lower in the dialog), these can be linked to the existing Object. Select the Library item and click on the link icon ( ) to

link the Object to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object as a copy of the library Resource.

6.3.5.9 External Media Properties Panel

The Properties Panel for an External Media Object shows the properties for the selected External Media.



External Media

MyVideo ☑ Target

☑ Use automatic scripting

External file name:

path\myvideo.flv ...

☑ Auto-play

Width: 160 pixels

Height: 120 pixels

Note: At the moment only Flash Video (.flv) video can loaded.

Name *(shown above as "MyVideo")*
The name of the External Media as also shown in the Outline Panel. The Name is based on the filename. If the object is set as a Target the name is the script reference.

Target
The Target option is grey (and unselectable) because it is implied that Extermal Media is always a target.

Add to Library
Add the object to the library or links to an existing library object. See more detail at Library.

Use automatic scripting
Set the "Use automatic scripting" checkbox so that the external media container uses an automatically generated container class that has methods for controlling the playback and loading of the external file. The scripting aspects of this container class are described in the External Media Class section of the scripting reference.

If "Use automatic scripting" is ***not checked***, a basic empty video object is supplied, and the author must include all script for creating and attaching the a network connection and stream to the external video container so it can load or playing the external file. For example, if there is a scene which has an external media container named "my_ExternalMedia" that ***does not*** use "Use automatic scripting", then script could be added (as shown below) to the scene to play the external file "myvideo.flv".

```
onFrame (1) {
   var my_nc : NetConnection = new NetConnection();
   my_nc.connect(null);
   var my_ns : NetStream = new NetStream(my_nc);
   my_ExternalMedia.attachVideo(my_ns);
   my_ns.play("myvideo.flv");
}
```

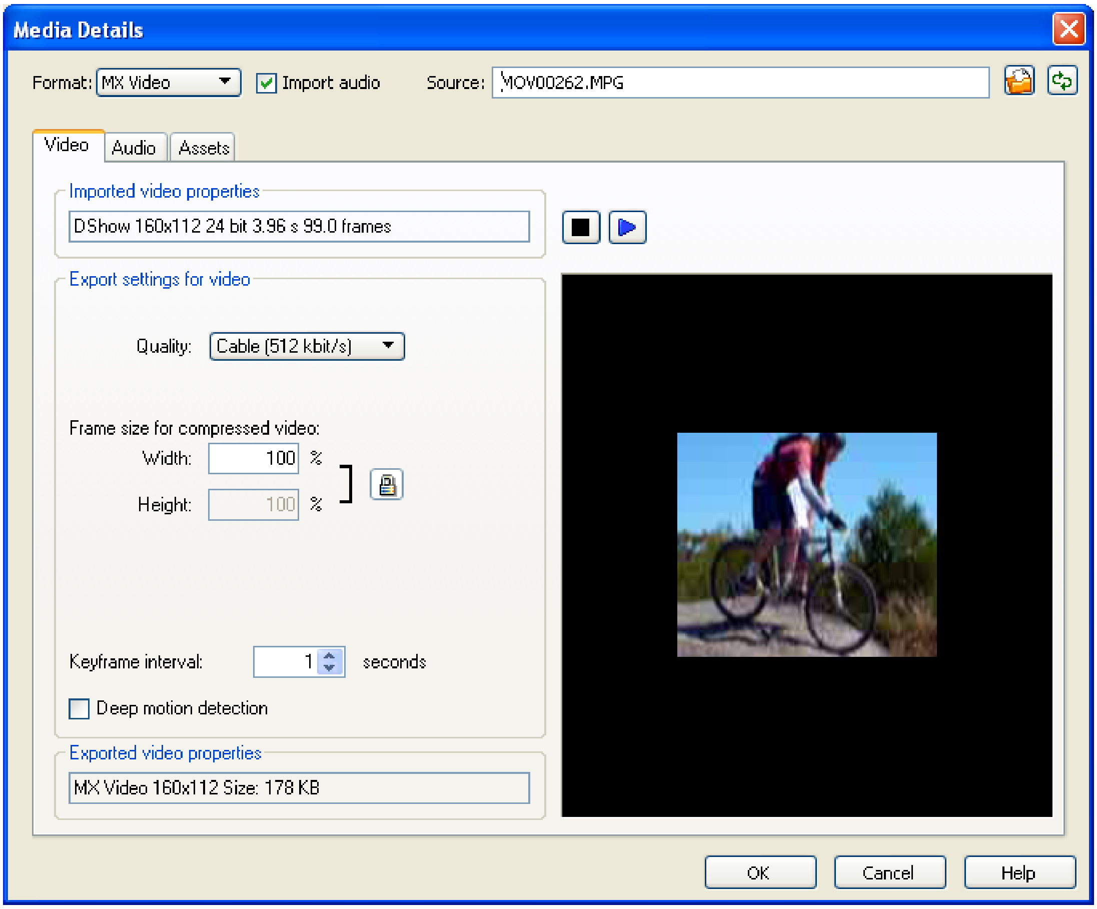
Auto-play
Set the "auto-play" checkbox so the external media container automatically loads and plays the external file. Otherwise, the external media container will load, but not play, the external file.

External File Name
Specify the name of the external file. This can either be a simple name, or an absolute or relative path to the file.

Width/Height
Set the width and height of the external media container in pixels.

6.3.5.10 Video Properties Tab

This tab is available when Importing a video and when adjusting the properties of an embedded video via the Embedded Video Properties Panel.



Format: MX Video ☑ Import audio Source: MOV00262.MPG

Imported video properties
DShow 160x112 24 bit 3.96 s 99.0 frames

Export settings for video
Quality: Cable (512 kbit/s)
Frame size for compressed video:
Width: 100 %
Height: 100 %
Keyframe interval: 1 seconds
□ Deep motion detection

Exported video properties
MX Video 160x112 Size: 178 KB

Format
This is the format that the imported video will be exported as.

Source
The path to the source video

Imported video properties

Lists the type of compression, original size, length and number of frames.

Export settings for video

The following settings define the quality of the exported video.

Quality:
Allows video compression and quality to be altered depending on the available bitrate.

Width, Height
The output frame size. The padlock button allows the aspect ratio to locked (Height altered according to width to maintain aspect ratio) or unlocked (width and height can be altered independently).

Keyframe interval
This defines how often a keyframe is inserted into the exported video.
Key frames are frames that do not depend on previous frame content (this a specific video conversion term). Their encoding is of a better quality than other frames, but they are much bigger in encoded size. They make the movie easier to seek into. You can adjust the interval in seconds between Key Frames.

Deep motion detection
This is only available with MX Video. This option activates a slower but efficient algorithm to compensate movie content motion. It is recommended to use it when source movie content is shaky (most of camcorder captures are) or when content has some rapid actions, like sport. The resulting converted movie file will be smaller or of a better quality if constant bitrate control is selected.

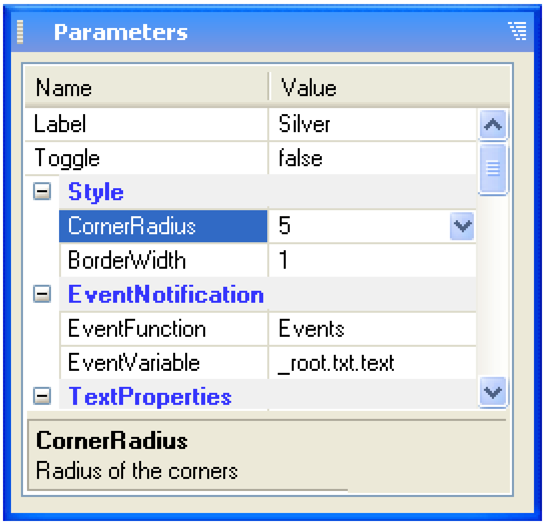
Exported video properties

Lists the export type of compression, revised size length etc.

The audio tab shown allows the associated audio properties to be viewed / altered.
The available parameters will depend on if the video is embedded or external.

6.3.6 Parameters Panel

The Parameters Panel allows the user to adjust the parameters for the selected component. The available parameters depend on the component that is selected.



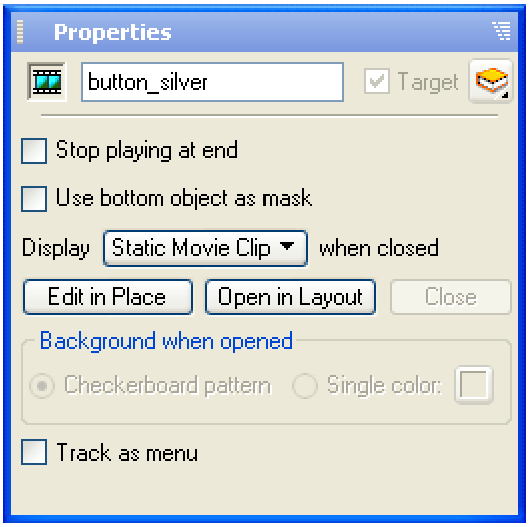
Name	Value
Label	Silver
Toggle	false
⊟ **Style**	
CornerRadius	5
BorderWidth	1
⊟ **EventNotification**	
EventFunction	Events
EventVariable	_root.txt.text
⊟ **TextProperties**	

CornerRadius
Radius of the corners

The columns lists the available parameters that can be altered. The **Name** column shows the current value of the parameter and allows alteration of that parameter. Depending on the Parameter and the Component, the value may be altered via either a **edit**, **list**, **combo**, **spin**, **color** or **group** control.

Underneath the Property / Name table, additional prompting is supplied for the currently selected parameter. In the case of the image above, the selected parameter 'CornerRadius' is used to define the "Radius of the corners". This field accepts a numeric value which will change the radius of curvature of the button corners.

Whenever a Component is selected the Properties for the Component will also be visible in another panel.



button_silver ☑ Target

□ Stop playing at end

□ Use bottom object as mask

Display Static Movie Clip when closed

Background when opened

● Checkerboard pattern ○ Single color: □

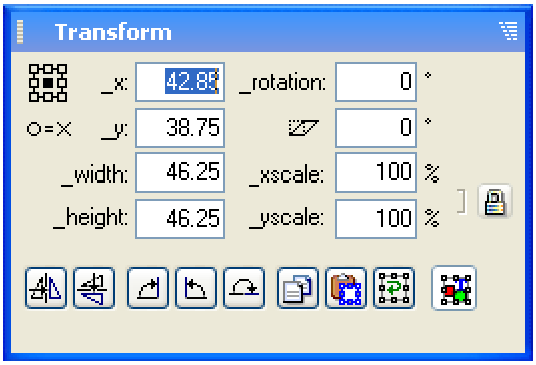
□ Track as menu

Refer to the section on Properties Panel for more details but in this example the top section of the panel shows the type of component, (in this case it is a Movie Clip) and the component's name (currently "button_silver"). As Components are targets, the name should be chosen so that it does not clash with any names currently in use in your Movie.

6.3.7 Transform Panel

The 'Transform' Panel changes the current object's position, size, scale and the angle of rotation and skew of the X and Y axes.

Also see Tools | Transform Tool.



_x: 42.85 _rotation: 0 °
_y: 38.75 0 °
_width: 46.25 _xscale: 100 %
_height: 46.25 _yscale: 100 %

Note: whilst in Preview Frame mode some of the Transform edit boxes are shown in red type if a Key Frame is selected. The red type indicates the options that will modify only that keyframe and not the object as a whole.

▦ / O=X 9 Point icon and the 'Reference = Transformation' point lock

The 9 point icon points represents the handles in the object selected. Click on a point to move the Transformation point to that handle. If the 'Transform = Reference' icon button is turned on, then the Transformation and Reference points will move together. Turning on the 'Transform = Reference' icon button will also move the Transformation point to the Reference point. See more details in the section 'To move the Transformation or Reference points'.

_x and _y position
These boxes change the x and y coordinates of the object's Reference point. Note that the reference point can be moved around the object. Entry into these boxes will alter the x and y position of the Object.

_width and _height
These boxes change the width and height of the object. Entry into these boxes will alter the width and height of the Object.

_rotation
This box changes the rotation of the selected Object. See more Tools section for information about Skew.

skew
This box changes the skew of the selected Object. See more Tools section for information about Rotate

_xscale and _yscale
These boxes are the scale factors applied to the width and height of the object as a percentage. Press the 'lock' icon to fix the aspect ratio of width to height.

Flip Horizontal
Flips the selected Object horizontally.

Flip Vertical
Flips the selected Object vertically.

Rotate 90
Rotates the selected Object by 90 degrees.

Rotate -90
Rotates the selected Object by -90 (270) degrees.

Rotate 180
Rotates the selected Object by 180 degrees.

Copy
Copies all the Transform settings of an Object.

Paste
Pastes all the transformed settings of a copied Object onto to the current Object. The Paste option has no affect if a Transform Copy hasn't already been made.

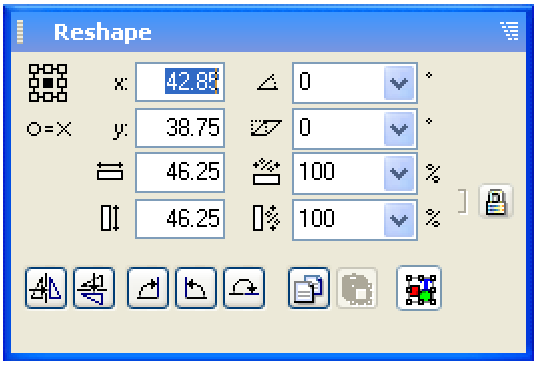
Reset
Resets the Transform to default settings, except for position.

Reshape / Transform as Group
'Reshape / Transform as Group' toggles the mode to change the selected, or grouped, objects as either a group or independently.

6.3.8 Reshape Panel

The 'Reshape' Panel changes the current object's position, size and the angle of rotation and skew of the X and Y axes.

Also see Tools | Reshape Tool.



/ 9 Point icon and the 'Reference = Transformation' point lock

The 9 point icon points represents the handles in the object selected. Click on a point to move the Transformation point to that handle. If the 'Transform = Reference' icon button is turned on, then the Transformation and Reference points will move together. Turning on the 'Transform = Reference' icon button will also move the Transformation point to the Reference point. See more details in the section 'To move the Transformation or Reference points'.

x and y position

These boxes change the x and y coordinates of the object's Reference point. Note that the reference point can be moved around the object. Entry into these boxes will alter the x and y position of the Object.

width and height

These boxes change the width and height of the object. Entry into these boxes will alter the width and height of the Object.

rotation

This box changes the rotation of the selected Object. See more Tools section for information about Skew.
Note: The object is rotated by the specified angle and the internal reference angle is reset to 0. This behaviour is different to rotation applied via the Transform Panel.

skew

This box changes the skew of the selected Object. See more Tools section for information about Rotate.
Note: The object is skewed by the specified angle and the internal reference angle is reset to 0. This behaviour is different to skew applied via the Transform Panel.

width resize and heights resize

These boxes are the resize factors applied to the width and height of the object as a percentage. Press the 'lock' icon to fix the aspect ratio of width to height.

Flip Horizontal

Flips the selected Object horizontally.

Flip Vertical

Flips the selected Object vertically.

Rotate 90
Rotates the selected Object by 90 degrees.

Rotate -90
Rotates the selected Object by -90 (270) degrees.

Rotate 180
Rotates the selected Object by 180 degrees.

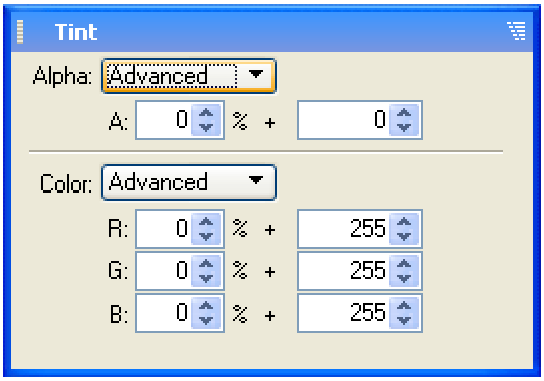
Copy
Copies all the Transform settings of an Object.

Paste
Pastes all the reshape settings of a copied Object onto to the current Object. The Paste option has no affect if a Reshape Copy hasn't already been made.

Reshape / Transform as Group
'Reshape / Transform as Group' toggles the mode to change the selected, or grouped, objects as either a group or independently.

6.3.9 Tint Panel

The 'Tint' Panel adjusts the color Transform for a given object. This is especially useful in 'Preview Frame' mode to edit the color Transform for the currently selected Effect.



Tint

Alpha: Advanced

A: 0 % + 0

Color: Advanced

R: 0 % + 255

G: 0 % + 255

B: 0 % + 255

There are two parts to the 'Tint' Panel:

- the 'Alpha' control
- the 'Color' control.

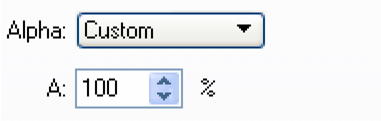
The options available depend on the selections from the pull-down lists.

Note: Whilst in the Preview Frame mode some of the Tint edit boxes are shown in red type if a Key Frame is selected. The red type indicates the options that will modify only that keyframe and not the object as a whole.

Alpha control

There are four selections for the 'Alpha' control:

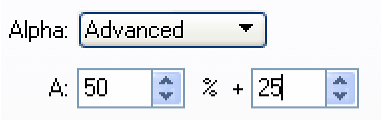
1. **Unchanged**: the alpha (transparency) value for the selected object is unchanged
2. **Transparent**: sets the alpha value to 0%, making the object completely transparent
3. **Custom**: specify the alpha value as a percentage of the current value.

   Alpha: Custom

   A: 100 %

   A value of 0% will make the object transparent. A value of 100% will leave the alpha unchanged. A value higher than 100% will make any transparent areas of the object more opaque.
4. **Advanced**: specify the new alpha value by taking a percentage of the current alpha value and adding a value to the result.

   Alpha: Advanced

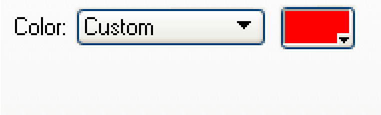
   A: 50 % + 25

   For example, setting the percentage to 0% and adding 100% will make the object 100% Opaque

Color control

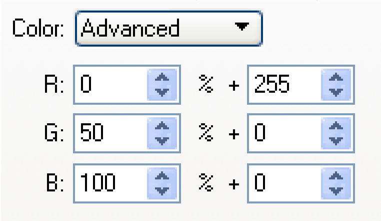
There are five selections for 'Color' control:

1. **Unchanged**: the color of the selected object is unchanged
2. **Black**: the object appears completely black (even if the object had multiple colors before)
3. **White**: the object appears completely white (even if the object had multiple colors before)
4. **Custom**: specify the color for the object using the Color Picker.

   Color: Custom 🟥

   The selected color is mixed with the original color of the object. The alpha value of the selected color gives the proportion of the selected color to mix. If the alpha value of the selected color is 0%, then the color of the object is unchanged; if the alpha is 100%, then the color of the object is completely replaced by the selected color; if, for example, a color with an alpha of 30% is selected, then the new color will be a mix of 30% of the selected color with 70% of the original color. Select the color by clicking on the color button to the right of the 'Color' Panel
5. **Advanced**: specify the red, green and blue components individually.

   Color: Advanced

   R: 0 % + 255

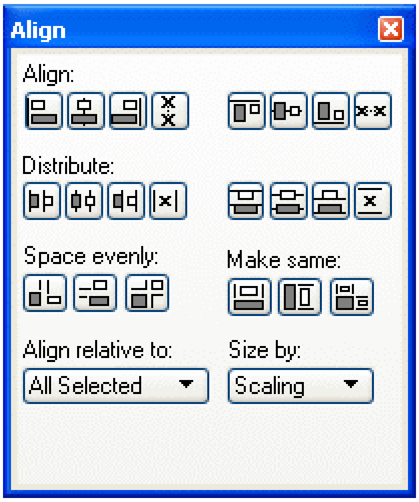
   G: 50 % + 0

   B: 100 % + 0

   For each, specify the new value by taking a percentage of the current value and adding a value to the result. For example, set the red component to 255, reduce the green component to 50%, and leave the blue component unchanged

6.3.10 Align Panel

The 'Align' Panel allows multiple objects to be aligned, distributed, spaced and resized.



Align relative to: All Selected

Size by: Scaling

The button images indicate the type of alignment, distribution, spacing or resizing to be performed.

Align

Align Left
Align selected objects to the left.

Align Horizontal Center
Align selected objects to the center horizontally.

Align Right
Align selected objects to the right.

Align Horizontal Anchor
Align selected objects horizontally by anchor point.

Align Top
Align selected objects to the top.

Align Vertical Center
Align selected objects to the center vertically.

Align Vertical Bottom
Align selected objects to the bottom.

Align Vertical Anchor
Align selected objects vertically by anchor point.

Distribute

Distribute Left

Distribute selected objects horizontally by left side.

Distribute Horizontal Center
Distribute selected objects horizontally by center point.

Distribute Right
Distribute selected objects horizontally by right side.

Distribute Horizontal Anchor
Distribute selected objects horizontally by anchor point.

Distribute Top
Distribute selected objects vertically by top side.

Distribute Vertical Center
Distribute selected objects vertically by center point.

Distribute Bottom
Distribute selected objects vertically by bottom side.

Distribute Vertical Anchor
Distribute selected objects vertically by anchor point.

Space Evenly

Space Horizontally
Space selected objects evenly horizontally.

Space Vertically
Space selected objects evenly vertically.

Space Both
Space selected objects evenly both horizontally and vertically.

Make Same

Same Width
Make selected objects the same width.

Same Height
Make selected objects the same height.

Same Size
Make selected objects the same height and width.

Align relative to ...
Options for the Align to list:

- **All Selected**: Makes all the alignment options align relative to all the selected objects
- **Last Selected**: Makes all the alignment options align relative to all the last selected object

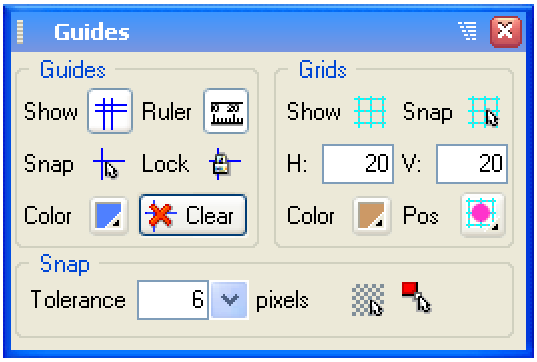
- **Parent**: Makes all the alignment options align relative to selected the objects parent. This is useful for positioning / aligning objects within a Movie Clip
- **Stage**: Makes all the alignment options align relative to the stage

Size by ...
Options for the size by list:

- **Scaling**: Scale selected objects when changing size
- **Resizing**: Resize selected objects when changing size

6.3.11 Guides Panel

The 'Guides' Panel alters settings associated with the placement Grids, Guides and Rulers.



H: 20 V: 20

Tolerance 6 pixels

Guides

Show Guides
Enables / disables the display of guide lines.

Snap to Guides
Enables / disables the snapping to guide lines.

Color
Selects the color for the Guide lines when displayed using the Color Picker. There is no alpha value available for this color selection.

Show/Hide Rulers
Shows (or hides) Rulers along top and left sides of the stage. The default setting is to have Rulers hidden.

Lock Guides
If set, prevents existing guide lines from being moved.

Clear All Guides
Clears all existing guide lines.

See the Guides section for more information on placing and moving guide lines.

Grids

Show/Hide Grids
Shows or hides a Grid over the stage area. The color of the Grid can be specified along with the Grid spacing. The default setting is to have no Grid.

Snap to Grid

Snaps object handles and locations to the nearest Grid line. This helps set a consistent size for, and positions of, objects. The snapping occurs regardless of whether the Grid is visible or not. The default setting is to have Snap to Grid turned off.

H: V:

Defines the Horizontal and Vertical spacing of the Grid in pixels.

Color

Select the color for the Grid lines when displayed using the Color Picker. There is no alpha value available for this color selection. The default color is grey.

Pos

Defines if the guides appear above or behind the stage objects.

Spacing - Horizontal and Vertical

Specifies the Grid size as pixels. The Grid can be used for display and for snapping object handles to the Grid. The default setting is 20x20 pixels.

Snap

Tolerance

The number of pixels from a snap location that'll will still cause an object to snap.

Snap to Pixel

Snaps object handles and locations to the nearest Pixel. A Pixel Grid will also be displayed when the movie is viewed at >=500% zoom.

Snap to Object Handles

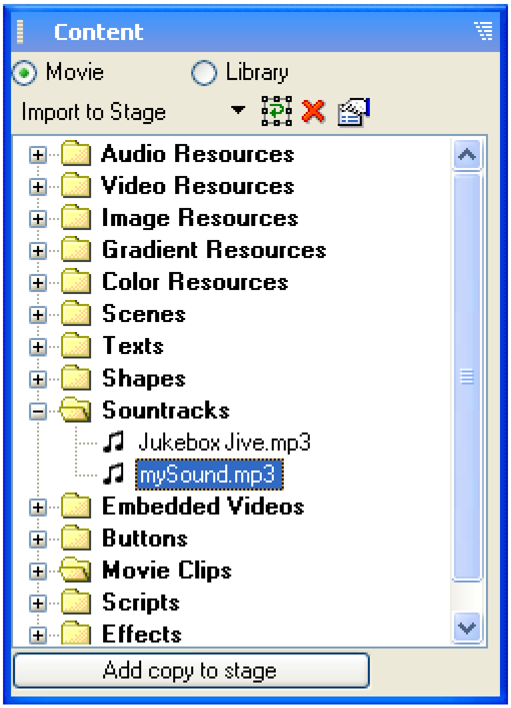
Snaps object handles so they align with neighboring objects. The default setting is to have Snap to Object Handles turned off.

6.3.12 Content Panel

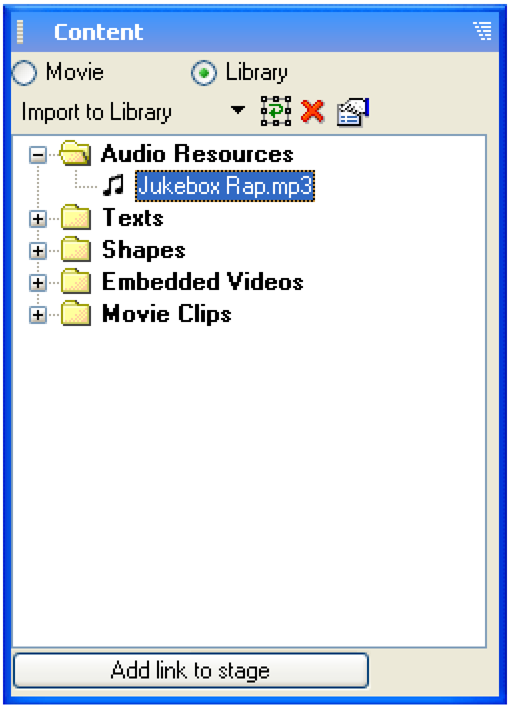
The Content Panel shows all the content available for use in, or to, your Movie. The Content is divided in the Movie view showing the Objects and Resources used in the current Movie, and the Library view showing the Symbols and Library Resources.

Any content placed on the stage will be listed in the Content | Movie view.

See the Library section for an explanation on managing Library content.



Content Panel - Movie Objects and Movie Resources



Content Panel - Library Symbols and Library Resources

Import to Stage / Import to Library
Imports indicated content directly to the stage (Layout Panel) or copies the content to the Library. These are the same as the File menu options: File | Import to Stage and File | Import to Library.

'Import to Stage' creates an appropriate Object for the content and places it on the Stage (Layout Panel).

'Import to Library' adds the content to the Library as a Resource or Symbol. If subsequently added to the stage, an appropriate Object is created which is either an Instance that links to the imported Symbol, or an Object that uses a link to the imported Resource.

See more information on imported content at these sections:

- Import Sound
- Import Video
- Import Image
- Import Animation
- Import Vector
- Import Text

Reload
Reloads the current content from an external file.

Delete
Deletes the selected content from the Movie.

Properties
Access Properties used for the content throughout the Movie.

Add Copy to Stage
Adds the content to the current Movie Scene.

Add Link to Stage
Add Library Item to the Stage. If a Resources is linked an appropriate Object is created and the Resource linked to the Object. If a Symbol is selected an Instance to the Symbol is created on the Stage.

Note: You can drag and drop the Content items (Movie and Library) from the Content Panel onto the Stage (Layout Panel) or onto the Outline Panel, which will add the item to the currently selected Scene in the Movie.

6.3.12.1 Library

The Library is used to accumulate and share common resources throughout the Movie. The Library can contain Symbols and Resources. Resources can be shared between a number of different Objects. The contents of the Library are called 'Library Items'.

Some Definitions

The **Library** is a collection of Symbols and Resources.
A **Library Item** is general description of any Symbol or Resources in the library.
A **Symbol** is an Object which is stored in the library (eg. Shape Object).
A **Resource** is any information stored in the library which is ***not*** a Symbol (eg. an image, color, gradient).
An **Instance** is a shared use of a library Symbol.
A **Linked Resource** is a shared use of a library resource.

The following topics discuss the Library and how to manage Library items:
Adding to the Library
Using Library Items
Instances and Symbols
Editing Symbols

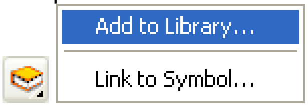
6.3.12.1.1 Adding to the Library

There are two ways to add to the Library:

- **Add to Library and create a link**: Adds the content to the Library as a Symbol then an Object is created (on the stage) which is an Instance of the imported Symbol (i.e. the object is linked to the Symbol).
- **Add a copy to Library**: adds the content to the Library as a Resource or Symbol. If subsequently added to the stage, an appropriate Object is created which is either an Instance that links to the imported Symbol, or an Object that uses a link to the imported Resource.

One or both of the options above are available at:

- **File | Import to Library ...** *('Add a copy to Library' only)*
- **Content Panel | Library | Import to Library** *('Add a copy to Library' only)*
- **'Add to Library' button** *(available on Properties Panels)*
  This option offers a submenu with the "Add to Library..." option



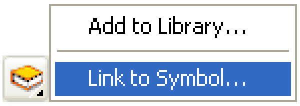
6.3.12.1.2 Using Library Items

When a Library item is added to the stage an appropriate Object is created which is either an Instance that links to the imported Symbol, or an Object that uses a link to the imported Resource.

Library Items can be placed in the following ways:

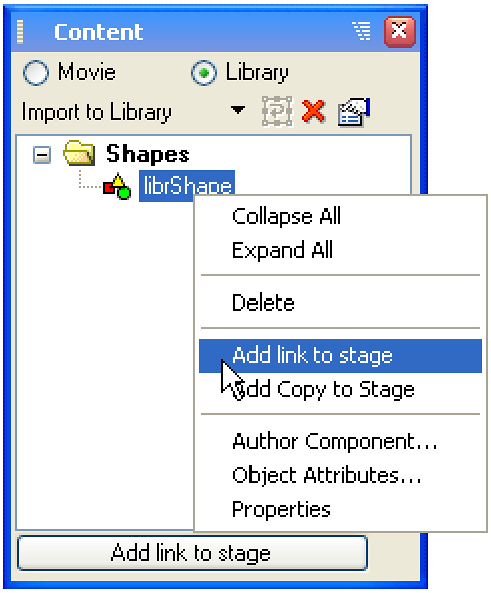
- **Insert | Library Symbol...**: Refer to the section Library Symbol...
- **Drag-drop from Content Panel** Library to the Layout Panel (stage), or the Outline Panel.
- **Library Resource Picker** (in appropriate Properties Panel, see below)

- **'Add to Library' button | Link to Symbol** (in appropriate Properties Panel)
  'Add to Library' option adds the object to the library or links to an existing Library Object. If there are existing Resources these can be linked to the current Object. Select the Library item (in the Library Resources Picker, see below) and click on the link icon ( ) to link the Object to the library resource. If already linked click on the unlink icon ( ) to break the link and leave the Object with a copy of the library Resource.

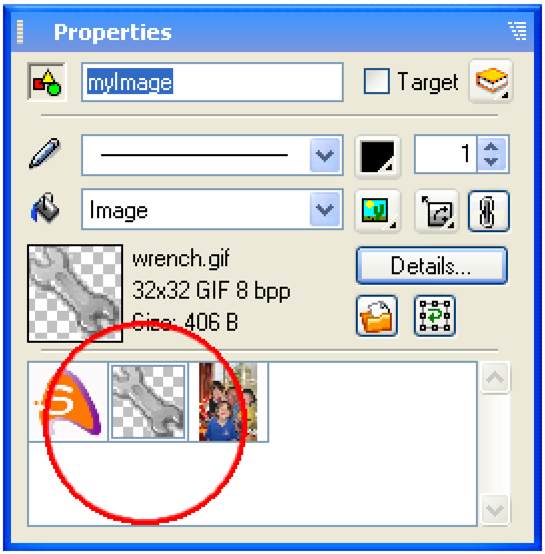
  Selecting 'Link to Symbol' opens the 'Insert from Library' dialog to select an appropriate Symbol or Resource.

- **Content Panel | Library option 'Add link to Stage'** 'Add Link to Stage' option is available from the button on the Panel and in the right-click context menu.



The context menu allows you to Collapse or Expand the entire content tree, delete a selected item, Add a link or Copy of the item to the stage, Author the component, modify Object Attributes or Alter the object Properties.

Library Resources Picker
Some Object's Properties Panel contain a 'Library Resource Picker'. Shown in the example below the Library Resources are images.



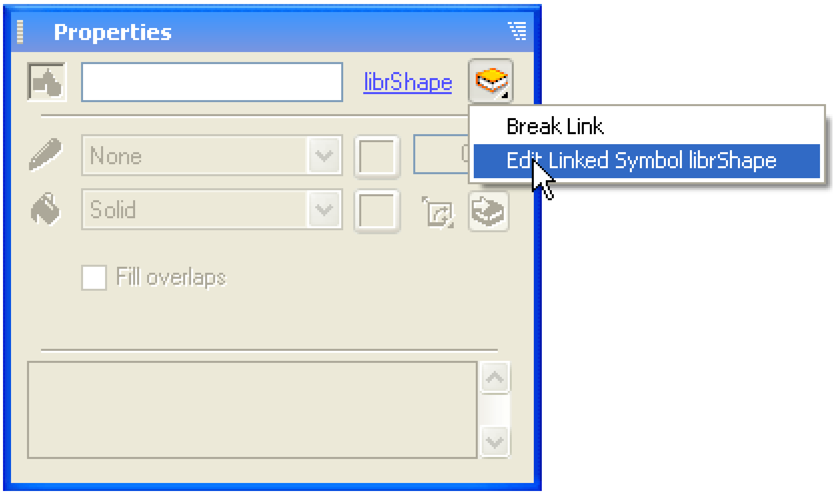
Shape Object Panel showing the Library Resources Picker and the images available to be linked for the image fill.

The Library Resource Picker contains library resources appropriate for use within the selected Object. The Library Resource Picker does not contain library items of the same type (as the Object) but resources which can be used in the selected Object. For example gradients for gradient fills, colors for use in the Color Picker and for solid fills in the Shape Properties Panel.

6.3.12.1.3 Instances and Symbols

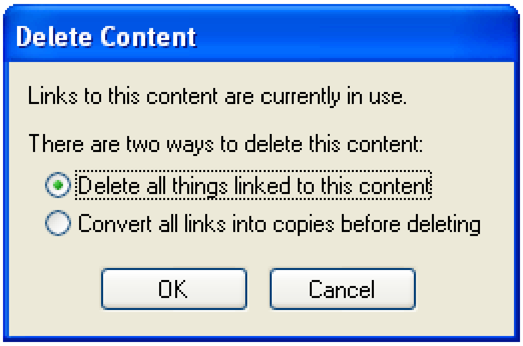
An Instance of a Symbol can be used through out the Movie. Updates to the Symbol will apply to all Instances in use. Information is stored only in the Symbol itself in the Library. Since all Instances get their information from the master symbol, SWiSH Max only needs to a repaint etc. to have the other Instances update, there is no need to copy information around. A disadvantage is that if the master somehow becomes deleted, the instance no longer has any master to reference, so take care not to delete master Symbols, indeed SWiSH Max will warn if against deleting Symbols in use.

If an Instance is placed and is selected, the Properties Panel displays the appropriate information for the type of object the Instance is an instance of, and the appropriate editing or drawing tools for that type of object are made to work. e.g. If the Instance is of a Shape Object, then the Shape Object Properties Panel is displayed and the shape editing tools are enabled. When editing an Instance, all changes to non-instance-specific information are made to the corresponding object in the content. Any other instances of that object are updated to reflect the change. When the SWI file is saved the Library with all Symbols and Resources is saved with the file.



Instance of Shape Object (Solid Fill) Library Symbol 'librShape' showing Library options: Break Link and Edit Symbol

Deleting a Library Item will prompt the user with the following dialog.



Delete Content

Links to this content are currently in use.

There are two ways to delete this content:

● Delete all things linked to this content

○ Convert all links into copies before deleting

- **Delete all things linked to this content**: Deletes all Objects with linked resources or Instances of Symbols.
- **Convert all links into copies before deleting**: Deletes the Library item but leaves the Objects with linked resources or the Instances as copies of the original Library Item.

6.3.12.1.4 Editing Symbols

Library Items can be edited or configured by double-clicking on the item in the Content Panel Library view or right-click and select the context menu items 'Details...' for Resources, or 'Properties..' for Symbols.

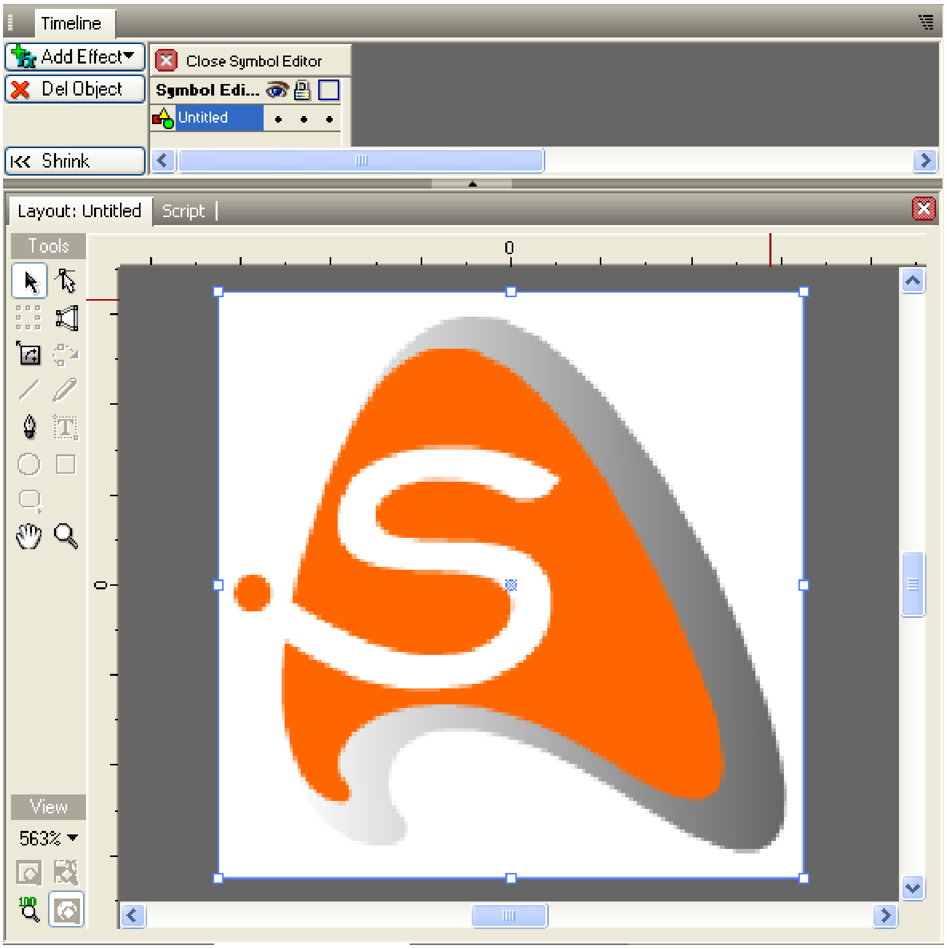
When Symbols are edited the Layout Panel view changes to edit the Symbol alone. The Symbol can be edited as per normal. The Tools available depend on the object edited. For the example image shape below Tools not available are greyed out.

To leave the Symbol editor select the 'Close Symbol Editor' button in the Timeline (see example below). Editing a symbol in the Symbol Editor actually edits a copy of the symbol so any changes made to the Symbol are not copied back to the Library until the Symbol Editor is closed.

A single shape Symbol can be edited but shapes can not be added. It is possible to add to a group or Movie Clip Symbol.

A shape Symbol can not be moved in the Symbol editor, though it's vertices can be changed. Note the reference point of the shape is fixed in the Symbol editor view. When editing a Movie Clip Symbol which has a shape inside it, then that shape can be moved within the Movie Clip.

Tip: To hole punch an existing shape Symbol convert it to a Movie Clip Symbol, apply the changes and then ungroup back to a shape.



6.3.12.1.5 Assets

Background
As part of the Adobe Flash file format (SWF) "definition tags" are used to define the content of the Flash (SWF) movie i.e. the shapes, text, bitmaps, sounds, videos etc. that are used in the movie. Each definition tag assigns a unique ID called a character to the content it defines. The Flash player then stores the character is a repository called the dictionary. Normally objects, and definition tags are not available to other SWFs during runtime. SWiSH Max allows the author to set an Object as an asset and enable it for export, or import. This Object is then said to be an "asset".

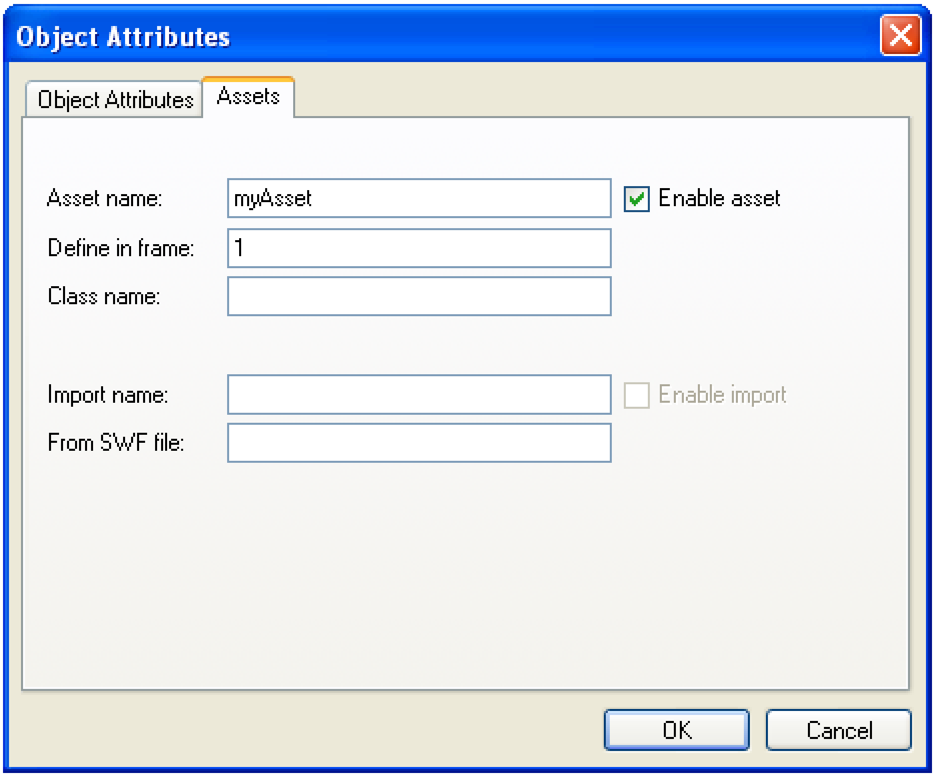
An Asset can be used for:
- scripting. Movie Clips etc. can be dynamically created using the asset name.
- shared assets. Named assets can be used from one SWF file in other SWF files.

The Object Attribute panel for Objects (stage and library) includes an assets tab with options to:
- **enable asset**: enables the object to be an assets. the asset MUST have a name.
- **Define in frame**: sets the export frame for the asset. For example, when using a preloader it is

preferrable to define all the assets AFTER the preloader is complete.

- **Class name**: Classes can be created using prototypes (see Custom Classes). The 'Class Name' is used to associate a class with the asset. The class can extend the behavior of the built-in object. Whenever an instance of the asset is created, the properties and behaviors of the instance are defined by the class assigned to it.

- **enable import**: imports an asset from another SWF (during runtime). Options include:
- **Import name**: Enter the name of the asset to be imported
- **From SWF file**: Enter the absolute or relative path of the SWF containing the asset (as named above). Relative path e.g. "subfolder\assets.swf", "..\..\subfolder\assets.swf" or same folder "assets.swf". Absolute path e.g. "http:\\www.mysite.com\files\assets.swf". where mysite.com is the same domain.



Object Attributes

Asset name: myAsset ☑ Enable asset

Define in frame: 1

Class name:

Import name: ☐ Enable import

From SWF file:

An asset can be imported from other SWF and also (re-)named (for export). When both options are set the definition is stripped form the SWF, and instead an import tag is included at authoring time as a placeholder for the object/symbol/resource defined in the SWI.

Note: to import an asset from another SWF

- that SWF also ***MUST*** to have Assets defined
- that SWF ***MUST*** be present at runtime

If a SWF is not available to import the asset the Flash player will simply pass over the problem without explicitly reporting the error.

Objects Supporting Asset naming

Some objects can not be used as assets.

Can be an Asset:

- Any Object other than Embedded video or scenes.

Can ***NOT*** be an Asset:

- Embedded Video
- Scenes
- Soundtracks
- Gradients

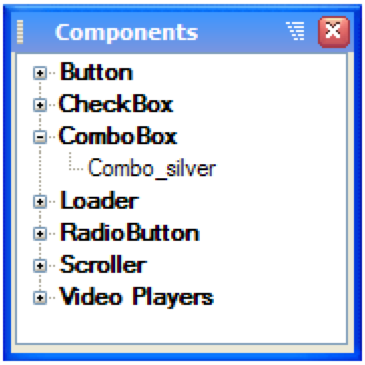
Example

```
onSelfEvent(load)
{
        attachMovie("library_video", "myvideo", 1);
}
```

The example links an instance of a symbol with asset name "library_video" from the library and attaches it to the movieclip. The instance has the name "myvideo".

6.3.13 Components Panel

The Components Panel allows the user to add a specific components to their current Movie.



In the default configuration, the Components Panel will appear in the bottom right hand corner along with the **Outline**, **Content** and **Effect** panels.

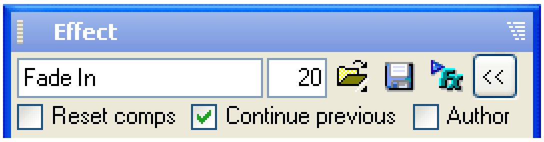
Components are grouped according to broad categories. The categories can be opened or closed with the **+** / **-** symbol. When opened, the individual Components are displayed.

To use a component, use the mouse to click-select and drag it from the the Parameters Panel onto the appropriate position in the Layout Panel.

When a Component is selected the Parameters Panel will become visible. The Parameters Panel is used to adjust the settings for the component.

6.3.14 Effect Settings Panel

The 'Effect Settings' Panel is displayed when you create an Effect. It can also be displayed by double-clicking on an Effect in the Timeline, or right-clicking on an Effect. Below is part of the 'Effect Settings' panel, which is common to all Effects.



Fade In | 20

☐ Reset comps ☑ Continue previous ☐ Author

You can save and reuse Effect Settings. See Saving Effects for more information.

Name (e.g. "Fade In")
Specifies the name of the Effect. This name appears on the Timeline.

Duration (e.g. "20")
Specifies the duration of the Effect as a number of Frames. For some Effects, the duration is either preset (e.g. the Remove Effect is always a single Frame), or calculated depending on the Effect Settings (e.g. the Typewriter Effect). In these cases, you cannot directly edit the duration.

Load
Loads predefined Effect Settings from the corresponding Effect folder on disk.

Save
Saves the current Effect Settings to the corresponding Effect folder on disk. See Saving Effects for more information.

Preview Effect
Previews the current Effect.

■ **Stop** *(toggle state of 'Preview Effect' above)*
Stops previewing the current Effect.

>> << **More/Less Options**
Selecting this option lets you define whether to show the 'Start At', 'Motion' and 'Easing' tabs and the 'Reset comp.' and 'Continue from/previous' effect options. By default Less Options is selected, meaning these tabs and options will not be shown.

Reset comp. (Reset components to original Transforms)
If the previous Effect was a Complex Effect that transformed the component positions, then ticking this checkbox will reset the components back to their original positions before the next Effect starts. Otherwise, the components are left in whatever state they were in at the end of the previous Effect. To demonstrate the effect apply the core effects 'Explode' and then '3D Spin' to some text. Set the 'Reset Comp' setting of 3D Spin effect and the position after the previous effect is reset to the original placement.

Continue from previous (Continue from previous effect)
This option tells SWiSH Max to apply the Effect starting with the Object as it was at the end of the previous Effect. If you want to specify a different starting condition for the Effect (or want to reset the elements of the Object to their initial positions), turn this option off. SWiSH Max will then add the Start At tab that lets you change the starting Transforms for the Effect.

Author
This option allows you to access the Effects Authoring options. These additional options allow you to build

your own SWiSH Max Effects. See Effects Authoring for more details.

Note: The Author option is only available in the Effect Settings Panel if the 'Allow effect authoring' checkbox is checked in the 'Tools | Preferences | Effects' dialog box.

There are a number of Effect sheet tabs that are common to two or more Effects. These are as follows:

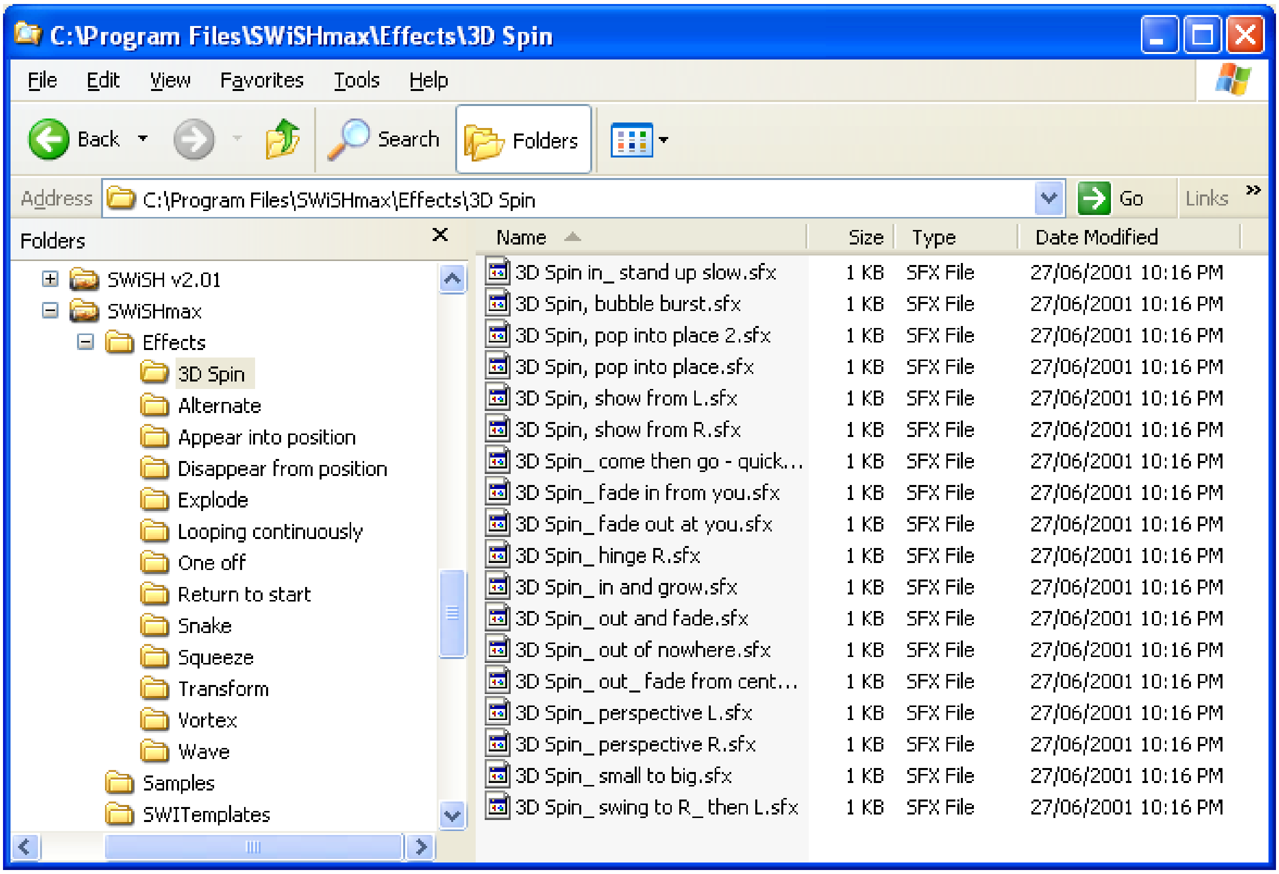
- Motion
- Easing
- Start At
- Cascade
- Camera
- Transforms
- Audio
- Video.

The other tabs are described in the corresponding Effects.

6.3.14.1 Saving Effects

Effect Settings are written to the Effects sub-folder of the SWiSH Max folder on your hard disk.

The Effects folder contains multiple sub-folders for each main Effect category. Within each category folder, multiple Effect Settings files (.sfx extension) can be stored. This provides convenient categorization and grouping of Effects.



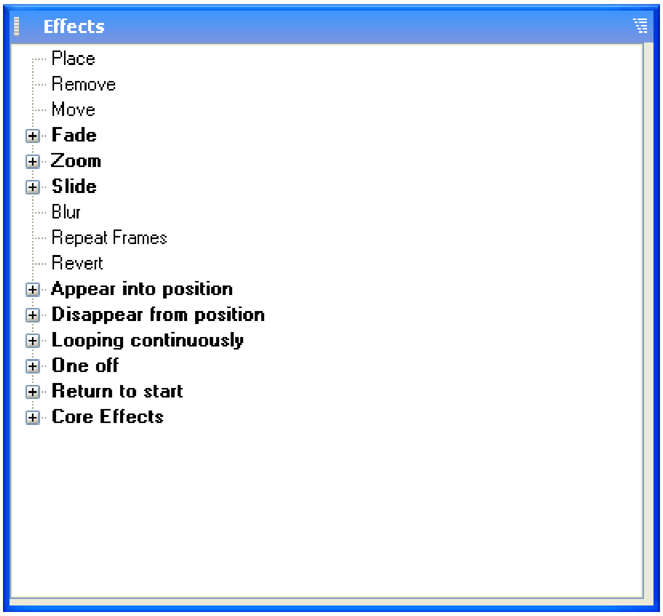
Name	Size	Type	Date Modified
3D Spin in_ stand up slow.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin, bubble burst.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin, pop into place 2.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin, pop into place.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin, show from L.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin, show from R.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ come then go - quick...	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ fade in from you.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ fade out at you.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ hinge R.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ in and grow.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ out and fade.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ out of nowhere.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ out_ fade from cent...	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ perspective L.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ perspective R.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ small to big.sfx	1 KB	SFX File	27/06/2001 10:16 PM
3D Spin_ swing to R_ then L.sfx	1 KB	SFX File	27/06/2001 10:16 PM

When creating and authoring Effects, you may decide to create your own categories. Simply create a folder within the Effects folder and save your Effects to that folder.

As the Effect Settings are stored within the .sfx file, you can exchange .sfx files with your friends to allow them to use the Effects you have created.

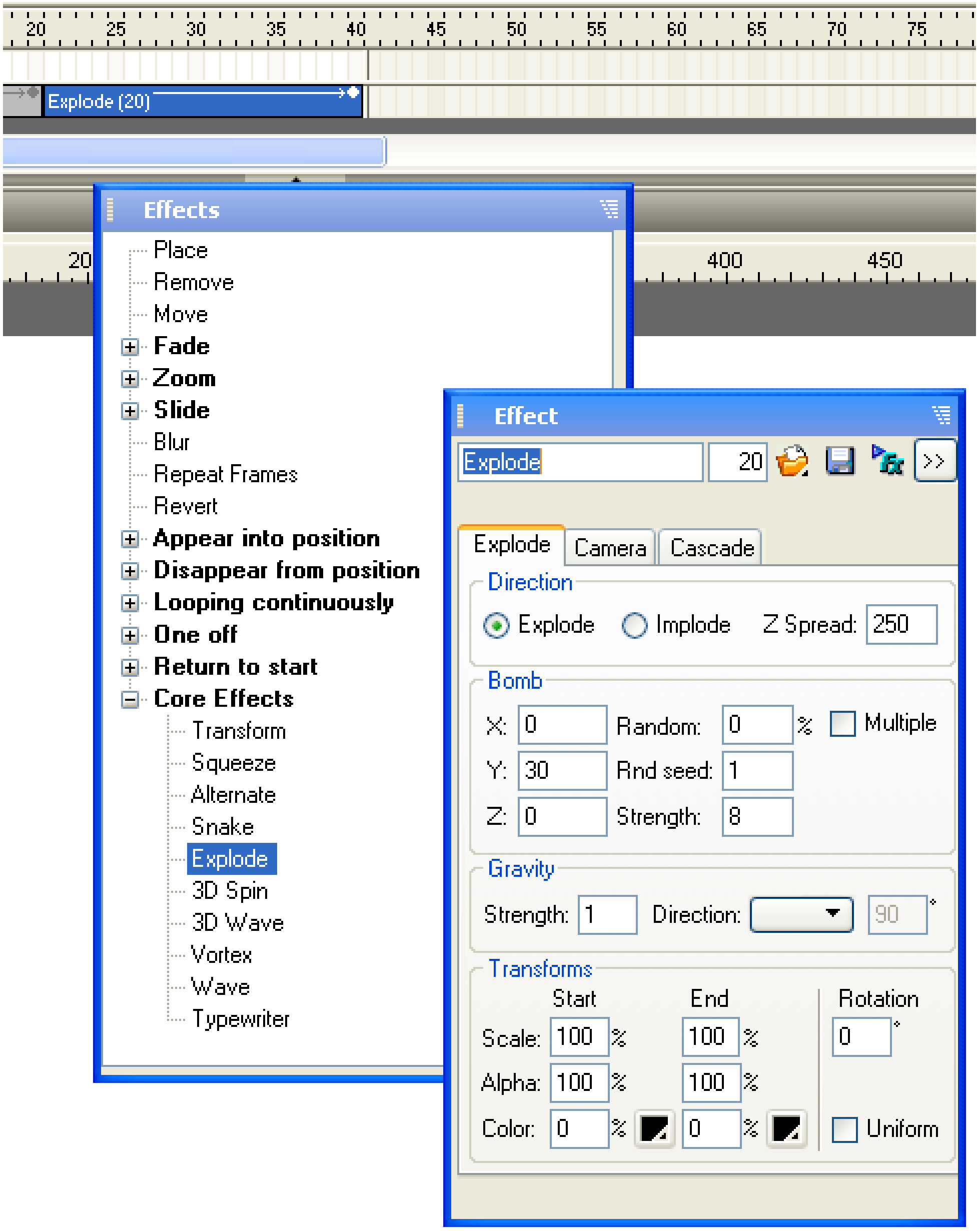
6.3.15 Effects Browser Panel

The 'Effects' Panel replaces any existing effect with the new effect selected.



Click on any frame in the timeline and click on an Effect (in the Effect Browse) to insert or replace (an existing Effect) or Alt-Click to insert an Effect.

When an object is selected, play the movie to preview the animation and select other Effects from the Effect Browser to get a live update. Use 'Edit | Undo' (Ctrl+Z) to restore the original effect and settings after the preview.



Explode 20

Direction

● Explode ○ Implode Z Spread: 250

Bomb

X: 0 Random: 0 % □ Multiple

Y: 30 Rnd seed: 1

Z: 0 Strength: 8

Gravity

Strength: 1 Direction: 90 °

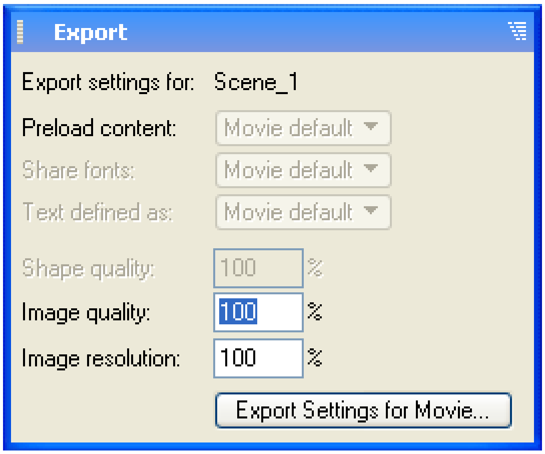
Transforms

	Start		End		Rotation
Scale:	100 %		100 %		0 °
Alpha:	100 %		100 %		
Color:	0 %	◪	0 %	◪	□ Uniform

6.3.16 Export Panel

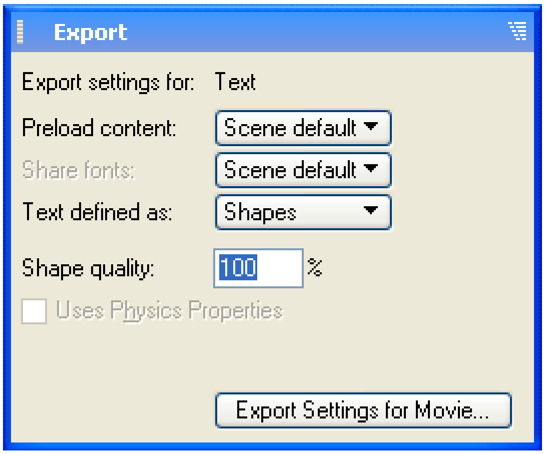
The 'Export' Panel sets the export options for objects in the Movie when generating the .swf file for the File | Test | SWF in Flash Player, File | Export | SWF... and File | Export | HTML+SWF... commands. The 'Export' Panel displays the export settings for selected items whether that be for a scene, object or a selection of objects. The options available will vary depending on what is selected.

The following sections describe all possible options available, some options may not apply, depending on the selection.



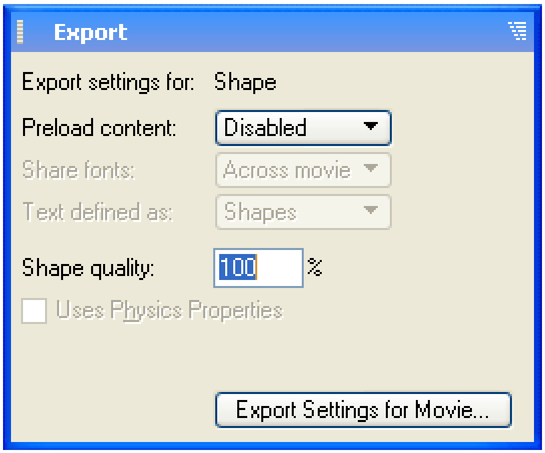
Export settings for: Scene_1
Preload content: Movie default
Share fonts: Movie default
Text defined as: Movie default
Shape quality: 100 %
Image quality: 100 %
Image resolution: 100 %

Export Setting Options for Scene



Export settings for: Text
Preload content: Scene default
Share fonts: Scene default
Text defined as: Shapes
Shape quality: 100 %
□ Uses Physics Properties

Export Setting Options for Text



Export settings for: Shape
Preload content: Disabled
Share fonts: Across movie
Text defined as: Shapes
Shape quality: 100 %
□ Uses Physics Properties

Export Setting Options for Shape

Note: The default options are "Movie default". These are defined in the 'Export Settings' dialog box SWF section which can be accessed by the 'Export Settings for Movie..' button.

Preload content

This option determines where object definitions are written to the .swf file. By grouping definitions, delays or jerkiness can be avoided while the Movie is playing. The Preload content options are as follows:

- **Disabled**: definitions are written out at the Frame where the object is first placed. If the definition is large because there is a large object or block of text, or if the connection speed is slow, this could introduce a short delay or jerk in the playback
- **Before Scene**: definitions are grouped together at the start of the Scene. This may mean a short delay before the Scene starts playing, but such delays are generally less noticeable than those that occur while the Scene is playing
- **Before Movie**: definitions are grouped together before the first Frame of the Movie. This can mean a delay before the Movie begins to play
- **At Preload Action**: adds a Preload Content Action as a Scene Event. This acts as a marker for where definitions are to appear. Any objects before the first Preload Content action are treated as though Preload content was disabled
- **Scene default**: the value for this setting is taken from the default defined for the Scene, and is not overridden here. This is the default and unless the Scene and Movie settings are changed the effective value will be the same as the At Preload Action setting
- **Movie default**: the value for this setting is taken from the default defined in the 'Export Settings' dialog box which can be accessed by the 'Export Settings for Movie..' button.

Share Fonts

Font definitions in a .swf file define the shapes for characters. To save space, only the actual characters used appear in font definitions. This setting lets you specify where font definitions are written out and what they contain. Unless you particularly want to override the default setting for the Scene, you should leave the setting as 'Scene Default'. The possible values for this setting are:

- **Disabled**: a separate font definition is written out for each Text Object for the characters used in that text object only
- Across Scene: font definitions are written out at the start of the Scene that include all the characters used in the Scene. This saves space by combining multiple fonts
- **Across Movie**: font definitions are written out at the start of the first Scene found that has any Text Objects with Share fonts set to Across Movie. These definitions include all the characters used in the Movie
- **Scene default**: The value for this setting is taken from the default defined for the Scene, and is not overridden here. This is the default and unless you change Scene and Movie settings the effective value will be the same as the Across Movie setting.
- **Movie default**: the value for this setting is taken from the default defined in the 'Export Settings' dialog box which can be accessed by the 'Export Settings for Movie..' button.

However, in all cases, if a font has already been defined that includes all the necessary characters, the existing font definition is used. In general, an effective value for the Across Movie setting will result in the smallest .swf size. However, it may mean that there is a delay at the start of the Movie when fonts are defined. This can be a problem if you are using a preloader. In this case, it is common to use Across Scene as the default for the preloader Scene, and Across Movie as the default for subsequent Scenes.

Text Defined As

This option lets you export the characters of the Text Object as individual shapes rather than as fonts and text. This should make no difference visually. However, if you later import the .swf file, you cannot edit any text that has Text as shapes enabled. You may also need to use this option if your text does not appear in the correct font when importing it into another application, such as Flash. The possible values for this setting are:

- **Text**: text is not exported as shapes, but rather is exported as font definitions and text records. If Complex Effects are used, then SWiSH Max will automatically define the text as individual objects for each character, so the characters can be animated individually
- **Shape**: text is converted into shapes. SWiSH Max generates separate shape definitions for each combination of font size or color.
- **Scene default**: the value for this setting is taken from the default defined for the Scene, and is not overridden here. This is the default and unless you change Scene and Movie settings the effective value will be the same as the Disabled setting.
- **Movie default**: the value for this setting is taken from the default defined in the 'Export Settings' dialog box which can be accessed by the 'Export Settings for Movie..' button.

If you need to define text as shape, the most efficient setting will depend on the variety of letters, fonts, font sizes and colors that you use. You should experiment with the settings for each individual Movie to get the smallest possible .swf size.

Uses Physics Properties

Lets you control the motion of the Object, including spin, velocity, acceleration and friction using special physics properties. You also need to turn on the physics support in the 'Script' section of the 'Export Panel' and the Object must be set as a scriptable Target.

Shape Quality

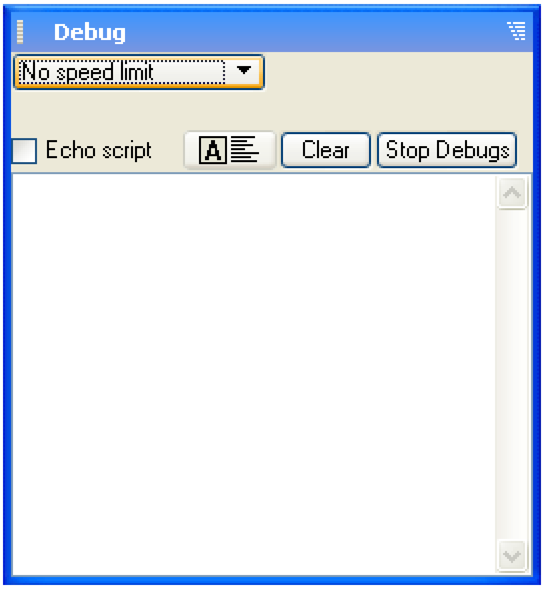
Specifies the shape quality up to 100%. Reducing a shapes quality will shift the vertices and control points in a shape to allow for smaller file size. Reducing quality will reduce the detail in the shape and change curvature of curved edges. Setting a shape quality of 0% will reduce the shape quality of all affected shapes to 0%, so they will become a single simple shape. Setting a shape quality of 100% would not change the quality for the shapes, and would use the settings for each individual shape. This sets the default value for the Shape Quality setting for the Scene. However, individual Objects can override these defaults.

Image Quality
Only applicable to JPEG compressed images that specify image quality, specifies image quality up to 100%. 0% would reduce the quality of the affected images to 0% resulting in a small file size but very poor quality. 100% would not change the quality for the images, and would use the settings for each individual image. This sets the default value for the Image Quality setting for the Scene. However, individual Objects can override these defaults.

Image Resolution
Specifies an images resolution up to 100%. Reducing an images resolution will reduces the images file size but add a mosaic effect to it. For an 100x100 pixel image, resetting the resolution to 50% will make it export as a 50x50 pixel image scaled by 200% to maintain the original image size. This sets the default value for the Image Resolution setting for the Scene. However, individual Objects can override these defaults.

6.3.17 Debug Panel

The 'Debug' Panel is used to help you see what is happening in your Movie's scripts. When you Play your Movie in SWiSH Max the 'Debug' Panel can display the script being executed and the value of your variables.



'No speed limit' list box
This option sets the speed of playback of your Movie. Speed options simulate internet connection rates. This is useful to demonstrate how fast your Movie will play or load when viewed online.

Speeds include:

- Custom Speed (set the rate in kb/s). Default is 512 kb/s.
- Dial-Up (56.5 kbit/s)
- ISDN (64 kbit/s)
- Cable (512 kbit/s)
- ADSL (1500 kbit/s)
- T1 (3Mbit/s)

Note: The speed limiting will only apply to content previewed entirely within SWiSH Max and can not apply to externally loaded content (e.g. using loadMovie()).

Echo Script
Outputs all script executed by your Movie to the 'Debug' Panel.

Auto Display button
When set the Debug Panel will be displayed when trace output is generated irrespective of whether the Debug Panel is visible or not.

Clear
Clears all current output from the 'Debug' Panel.

Stop/Start Debug
Stops or Starts the debug output being displayed in the 'Debug' Panel.

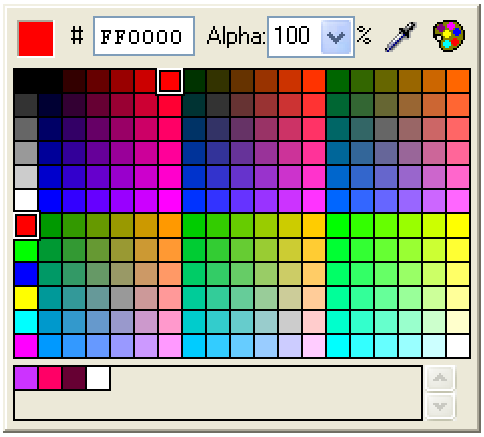
Note about SWiSH Max error checking:

SWiSH Max will detect uninitialised variables, missing variables, missing functions and missing methods if export is set to SWF6 or later. If export is set for SWF5 or earlier errors may still exist but will not be reported.

SWiSH Max will detect two runtime errors: missing variables (as above for SWF6+), and if a script section is too long to be compiled (all SWF versions). See Error Messages for a description of the error messages, their meaning and how to fix the problem.

6.4 Color Picker

The Color Picker pops up whenever the 'Color fill' button is clicked to select a color.



FF0000 Alpha: 100 %

RGB and Alpha Values
Directly enter hex values for the Red, Green, or Blue components of the color. If an Alpha value (transparency) is appropriate for the selection, enter this as a percentage, with 0% being completely transparent and 100% being completely opaque. These edit boxes also show the RGB and alpha values for the currently selected color.

The Alpha % is indicated by the cross hatching and color in the color swatch.

■ Solid color (Alpha = 100%)

▨ Opaque color (Alpha = 50%)

▢ Transparent color (Alpha = 0%)

Note:

- The Alpha value will initially be set to the last used value
- Transparent colors (Alpha = 0%) cannot be seen

Eyedropper

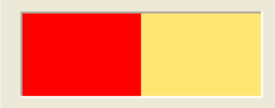
Click on this button to select a color from elsewhere on the screen. Move the eyedropper cursor over the color and click. To de-select a color, click on the 'Eyedropper' button again to deselect this function.

More Colors

Displays an enhanced Windows Color Picker, which gives a wider selection of colors and the option of displaying all colors, web safe colors, standard colors and colors put in the library. Each of the Color Picker views includes a Swatch, Eyedropper, MRU, 'Add to Library' and displays the colors as RGB, HSB, # and the alpha.

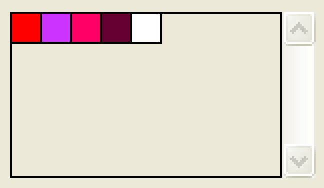
Swatch

This box shows the old color on the left hand side and the new color selection on the right.



MRU Colors

These buttons represent a selection of the Most Recently Used (MRU) colors. The colors in these boxes are updated every time a color selection is made.



Add to Library

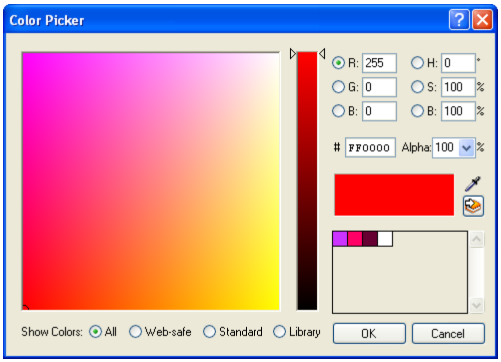
Adds the color to the library for use in other parts of the Movie.

Color Picker views

The Color Picker has the following views:

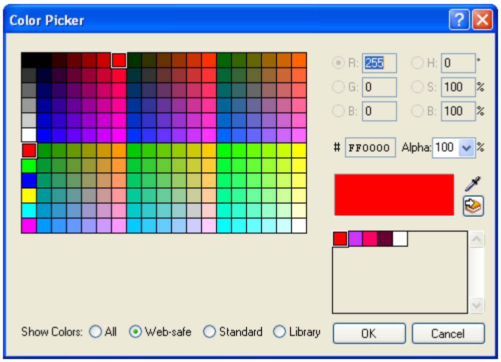
- **All**: This view displays all colors. Adjust the color by either RGB (red, green, blue) component, or by HSB (hue, saturation, brightness/lightness). The item defined by the radio button is fixed, all of the other items can be selected via the palette.
- **Web-Safe**: Select Web Safe colors. Web Safe, or Browser Safe colors consist of 216 colors that display non-dithered, consistent color on any computer or device with a 256 color (8-bit) display. Although 8-bit color can display 256 colors only 216 are displayed consistently. Web / Browser safe colors were defined in 1996 when the majority of computers had 8-bit cards.
- **Standard**: There are 140 Standard HTML colors names recognized by web browsers. To match a Movie's colors with HTML content it may be preferrable to refer to the color name directly.

- **Library**: This view displays any colors in the content library.



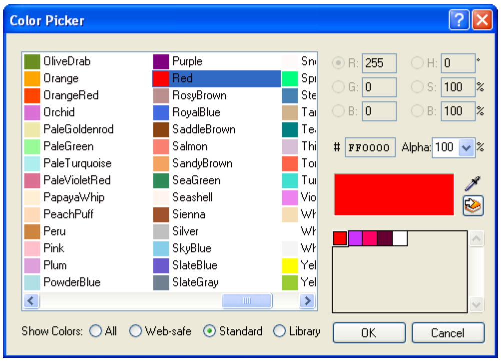
R: 255 H: 0°
G: 0 S: 100 %
B: 0 B: 100 %
FF0000 Alpha: 100 %
Show Colors: ● All ○ Web-safe ○ Standard ○ Library

Color Picker - 'All' view



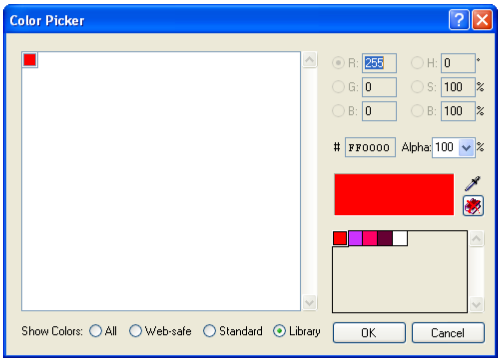
R: 255 H: 0°
G: 0 S: 100 %
B: 0 B: 100 %
FF0000 Alpha: 100 %
Show Colors: ○ All ● Web-safe ○ Standard ○ Library

Color Picker - 'Web-Safe' view



R: 255 H: 0°
G: 0 S: 100 %
B: 0 B: 100 %
FF0000 Alpha: 100 %
Show Colors: ○ All ○ Web-safe ● Standard ○ Library

Color Picker - 'Standard' view



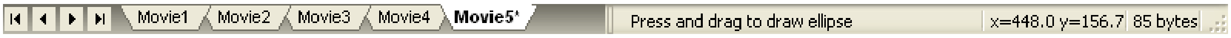
R: 255 H: 0°
G: 0 S: 100 %
B: 0 B: 100 %
FF0000 Alpha: 100 %
Show Colors: ○ All ○ Web-safe ○ Standard ● Library

Color Picker - 'Library' view

6.5 Status Bar

The Status Bar is at the bottom of the 'SWiSH Max application' window and is used to display help for the selected command or tool.

The Status Bar also displays the file size of the exported .swf file and the current x and y coordinates of the cursor in the Layout Panel. The size is calculated whenever a .swf file is exported, or the Test | SWF in Flash Player, HTML + SWF in Browser or Test Report commands are used. Until there is an export, the indicator says 'Unknown Size'.



Press and drag to draw ellipse
x=448.0 y=156.7
85 bytes

6.6 Toolbars

The Toolbars provide you with quick access to Menu commands and settings. Each Toolbar has a number of icons on it. These same icons appear with the corresponding Menu item.

Toolbars can be:

- floated over the 'SWiSH Max application' window
- docked around the edges of the 'SWiSH Max application' window.

When floated, each Toolbar has a ☒ 'Close' button, which hides the Toolbar.

There are a number of ways to rearrange Toolbars.

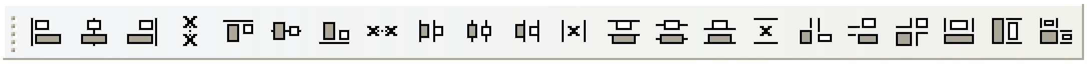
You can move a Toolbar around by dragging it by its title bar or gripper. You can drag it to another edge to dock it, or drag it elsewhere to have it float over the 'application' window. You can force a Toolbar to float by holding the Control key while dragging it. You can change the alignment of the Toolbar from horizontal to vertical, or vice versa, by holding the Shift key while dragging. You can double-click on the gripper of a docked Toolbar to float it, or double-click on the title of a floating Toolbar to dock it.

You can customize the appearance and content of Toolbars using the Toolbar Submenu and the Customize command on the View Menu.

The SWiSH Max Toolbars are:

- Alignment
- Control
- Export
- Grouping
- Guides
- Insert
- Library
- Menu Bar
- Standard
- Text
- Transform/Reshape
- View

6.6.1 Alignment



Also see Alignment operations.

Align to Left
Align selected Objects to the left.

Align to Center (Horizontally)
Align selected Objects to the center horizontally.

Align to Right
Align selected Objects to the right.

Align to Anchor (Horizontally)
Align selected Objects horizontally by anchor point.

Align to Top
Align selected Objects to the top.

Align to Center (Vertically)
Align selected Objects to the center vertically.

Align to Bottom
Align selected Objects to the bottom.

Align to Anchor (Vertically)
Align selected Objects vertically by anchor point.

Distribute to Left
Distribute selected Objects horizontally by left side.

Distribute by Center (Horizontally)
Distribute selected Objects horizontally by center point.

Distribute by Right
Distribute selected Objects horizontally by right side.

Distribute to Anchor (Horizontally)
Distribute selected Objects horizontally by anchor point.

Distribute by Top
Distribute selected Objects vertically by top side.

Align to Left (Ctrl+Alt+1)
Align selected Objects to the left.

Align to Center (Horizontally) (Ctrl+Alt+2)
Align selected Objects to the center horizontally.

Align to Right (Ctrl+Alt+3)
Align selected Objects to the right.

Align to Anchor (Horizontally)
Align selected Objects horizontally by anchor point.

Align to Top (Ctrl+Alt+4)
Align selected Objects to the top.

Align to Center (Vertically) (Ctrl+Alt+5)
Align selected Objects to the center vertically.

Align to Bottom (Ctrl+Alt+6)
Align selected Objects to the bottom.

Align to Anchor (Vertically)
Align selected Objects vertically by anchor point.

Distribute by Left
Distribute selected Objects horizontally by left side.

Distribute to Center (Horizontally)
Distribute selected Objects horizontally by center point.

Distribute to Right
Distribute selected Objects horizontally by right side.

Distribute to Anchor (Horizontally)
Distribute selected Objects horizontally by anchor point.

Distribute by Top
Distribute selected Objects vertically by top side.

Distribute to Center (Vertically)
Distribute selected Objects vertically by center point.

Distribute by Bottom
Distribute selected Objects vertically by bottom side.

Distribute by Anchor (Vertically)
Distribute selected Objects vertically by anchor point.

Space Evenly - Horizontally (Ctrl+Alt+7)
Space selected Objects evenly horizontally.

Space Evenly - Vertically (Ctrl+Alt+8)
Space selected Objects evenly vertically.

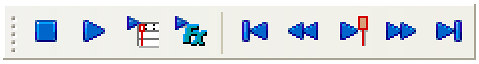
Space Evenly - Both
Space selected Objects evenly both horizontally and vertically.

Make Same Width (Ctrl+Shift+Alt+7)
Make selected Objects the same width.

Make Same Height (Ctrl+Shift+Alt+9)
Make selected Objects the same height.

Make Same - Both
Make selected Objects the same height and width.

6.6.2 Control



Stop
Stops playing.

Play
Plays the Movie.

Play Timeline
Plays the current Scene or Movie Clip only.

Play Effect
Plays the current Effect only.

Rewind
Rewinds to the first Frame in Preview Frame mode.

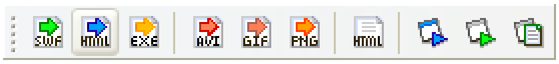
Step Back
Steps back to the previous Frame in Preview Frame mode.

Preview Frame
Switches to Preview Frame mode to preview the current Frame in the Layout Panel.

Step Forward
Steps forward to the next Frame in Preview Frame mode.

Go to End
Jumps to the last Frame in Preview Frame mode.

6.6.3 Export



Export to SWF (Ctrl+E)
Exports the current SWiSH Max Movie to a .swf file.

Export to HTML (Ctrl+P)
Exports the current SWiSH Max Movie to a .swf file and an .htm file

Export to EXE (projector) (Ctrl+Shift+P)
Exports the current Movie to a .exe file, known as a Projector. A Projector does not require the Flash Player to be installed to play the Movie.

Export to AVI (Ctrl+D)
Exports the current SWiSH Max Movie to a .avi file.

Test In Browser
Creates temporary .htm and .swf files, launches the default browser and displays the HTML page containing the Movie in the browser.

Test In Player
Creates a temporary .swf file, launches the Flash Player and plays the Movie in the Flash Player.

Report
Produces a report of the exported file.

For Example:

```
Filename: Movie.swf
Version: SWF7
File length: 32 bytes (40 bytes filesize)
```

```
Frame size: 560 x 410 pixels
Frame rate: 25.00 frames/sec
Total number of frames: 1 frames

- Entire Movie ----Tags---------------
+       Header:               21 bytes
+   ShowFrames:      1         2 bytes
+      BgColor:      1         5 bytes
+      PSPaths:      1         2 bytes
--------------------------------------
        Total:      3         30 bytes


- Preload Before Movie ---------------
+      BgColor:      1         5 bytes
+      PSPaths:      1         2 bytes
--------------------------------------
         Total:      2         7 bytes

- Scene #1 - Scene_1 -----------------
+   ShowFrames:      1         2 bytes
--------------------------------------
         Total:      1         2 bytes

- End --------------------------------
         Total:      0         0 bytes


- Frame Lengths ----------------------

- Scene #1 - Scene_1 -----------------
     -Frame-   -Bytes-   -Total-   -%-
           0         9         9  100%
--------------------------------------
```

6.6.4 Insert

Insert Scene
Inserts a new Scene into the Movie.

Insert Button
Inserts a button into the current Scene/Movie Clip/Group.

Insert Movie Clip
Inserts a Movie Clip into the current Scene/Movie Clip/Group.

Insert External Media
Inserts an external media container which can be used to load and play an external file in the current Scene/Movie Clip/Group.

Insert Library Symbol
Insert a symbol from the Library into the current Scene/Movie Clip/Group.

Import Sound...
Allows you to insert an audio file as a Sound Object to be used as a streaming Soundtrack.

Import Video...

Imports an external video file and converts it to Flash Video as either embedded video or an external FLV file.

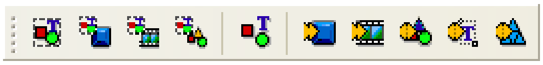
Import Image
Inserts an image from an external file into the current Scene/Movie Clip/Group.

Import Animation...
Imports an animation as a sequence of shapes.

Import Vector...
Imports vector graphics as a vector shape.

Import Text
Inserts a Text Object into the current Scene/Movie Clip/Group.

6.6.5 Grouping



Group as Group (Ctrl+G)
Groups the selected Objects together.

Group as Button
Groups the selected Objects together as a Button.

Group as Movie Clip
Groups the selected Objects together as a Movie Clip.

Group as Shape
Groups the selected texts and shapes together as a single Shape.

Ungroup (Ctrl+U, Ctrl+Shift+G)
Splits a Group/Movie Clip/merged-shape into separate Objects.

Convert to Button
Converts the selected Object into a Button.

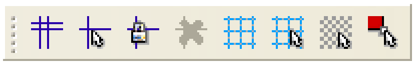
Convert to Movie Clip
Converts the selected Object into a Movie Clip.

Break into Shapes
Converts a complex shape into a Group Object containing separate individual Shape Objects for each simple component of the original complex shape.

Break into Letters
Converts a Text Object into a Group Object containing separate individual Text Objects for each letter of the text.

Break into Pieces
Converts the selected Object into a Group Object of smaller shapes.

6.6.6 Guides



Show Guides **(Ctrl+;)**
Displays the Guides.

Snap to Guides **(Ctrl+Shift+;)**
Snaps the cursor to Guides.

Locks Guides **(Ctrl+Alt+;)**
Locks Guides to the current positions.

Clear Guides
Clears all Guides.

Show Grid **(Ctrl+')**
Toggles display of Grid.

Snap to Grid **(Ctrl+Shift+')**
Snaps the cursor to the Grid.

Snap to Pixel
Snaps the cursor to the nearest pixel.

Snap to Object Handles **(Ctrl+Shift+/)**
Snaps the cursor to the object handles.

6.6.7 Library

Import Sound to Library
Adds a sound resource to the Library

Import Video to Library
Adds a video resource to the Library

Import Animation to Library
Creates a Movie Clip, Shape or Group Object with the animation frames in the Library as a Symbol

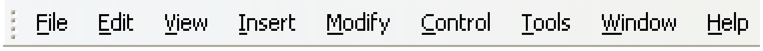
Import Image to Library
Adds an image resource to the Library

Import Vector to Library

Creates a Shape or Group Object with the vector graphic in the Library as a Symbol

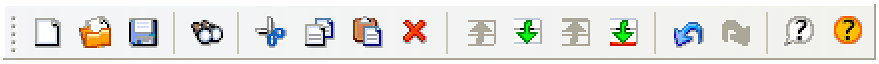
Import Text to Library
Creates a Text Symbol in the Library with the imported text

6.6.8 Menu Bar



This is the main SWiSH Max application menu.

6.6.9 Standard



New (Ctrl+N)
Creates a new SWiSH Max Movie.

Open (Ctrl+O)
Opens a .swi (SWiSH Max Movie) file.

Save (Ctrl+S)
Saves the current SWiSH Max Movie.

Find
Lets you search for text in your Movie.

Cut (Ctrl+X)
Deletes the currently selected Object or Effect and copies it to the Clipboard.

Copy (Ctrl+C)
Copies the currently selected Object or Effect to the Clipboard.

Paste (Ctrl+V)
Pastes the Object or Effect on the Clipboard to the current SWiSH Max Movie.

Delete (Delete)
Deletes the currently selected Object or Effect.

Bring Forward
Moves the selected Object one step closer to the top of the stack of Objects.

Send Backward
Moves the selected Object one step closer to the bottom of the stack of Objects.

Bring to Front
Moves the selected Object in front of all other Objects.

Send to Back
Moves the selected Object to the back of all other Objects.

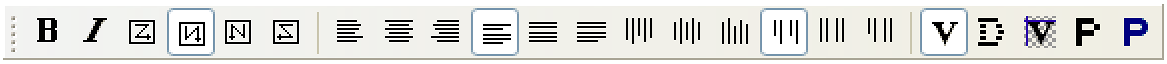
Undo (Ctrl+Z)
To undo the last change made to the current SWiSH Max Movie.

Redo (Ctrl+Y)
To redo the last change made to the current SWiSH Max Movie.

Help Topics (F1)
Lists help topics.

Context-sensitive Help (Shift + F1)
Provides help related to a specific topic. Click this button, then click the Object you require help with.

6.6.10 Text



Text Object operations, see Text Object Panel for more details.

Bold (Ctrl+B)

Italic (Ctrl+I)

Flow left to right, top to bottom

Flow top to bottom, right to left

Flow right to left, top to bottom

Flow top to bottom, left to right

Justify Left

Justify Center

Justify Right

Full Justification (*only available for Static text)*

Full Justification (all lines) *(only available for Static text)*

Full Justification (all except bottom line) *(only available for Static text)*

Vector Font

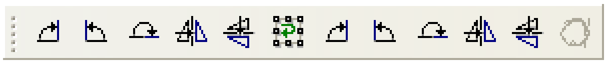
Device Font

Vector Font (pixel aligned)

Pixel Font

Pixel Fonts (smooth)

6.6.11 Transform/Reshape



Rotate 90
Rotates the selected Object by 90 degrees.

Rotate -90
Rotates the selected Object by -90 (270) degrees.

Rotate 180
Rotates the selected Object by 180 degrees.

Flip Horizontal
Flips the selected Object horizontally.

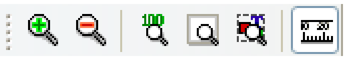
Flip Vertical
Flips the selected Object vertically.

Transform Reset
Resets the Transform to default settings, except for position.

You will notice that the above icons with the exception of Transform Reset appear to be repeated.
The left hand set of icons applies the transform using the transform method (ie final rotation and skew are displayed in the transform panel).
The right hand set of icons applies the transform using the reshape method (ie object is rotated / skewed and rotation and skew are reset to 0). Once applied, these transforms cannot be reset using the **Transform Reset** button.

Close Shape
Closes a vector shape (if open)

6.6.12 View



View | Zoom In (Ctrl+'+')
To increase the zoom factor of the Layout Panel, such that the Movie is displayed 50% larger than its previous size.

View | Zoom Out (Ctrl+'-')
To decrease the zoom factor of the Layout Panel, such that the Movie is displayed at two-thirds its previous size.

View | View at 100% (Ctrl+1)

Views the Movie at its actual size in the Layout Panel.

View | Fit Scene in Window (Ctrl+2)

To fit the entire Scene (stage area) into the Layout Panel.

View | Fit Objects in Window (Ctrl+3)

To fit all selected Objects into the Layout Panel. If no Object is selected, this function will fit all Objects in the Scene into the Layout Panel.

View | Show Rulers (Ctrl+R)

Shows or hides Rulers in the Layout Panel.

elp / Frequently asked Question

Chapter 7

7 Help / Frequently asked Questions

We have attempted to make SWiSH Max as easy to use as possible. However, the flash player is a very powerful tool and with this power comes a certain amount of complexity. There are a number of issues and tips that a new user will find useful. This section attempts to address those issues. Unfortunately, such a list can never be complete.

Issues with Device Fonts.
A device font will not scale correctly. If it is scaled, it may display at an incorrect size or may not display at all.
When previewing your movie on the stage, be sure to use a scale factor of 100% if you are using device fonts.
Note that if using html rendering of text, device fonts are used by default.

Scripting and Flash player versions.
Different flash player versions treat upper and lower case differently. See Flash Player History / Bugs for more information on the way different versions treat case.

Reducing .swf file size.
You can reduce the size of your movie by adding objects that are re-used to the library. Adding objects to the library can also simplify configuration as editing one item in the library will alter all of the objects that are linked to that item. If you have images in your movie, consider the Details... button which will allow you to adjust the size and compression of your image files.

Shapes with holes.
These can be made by using the Group as Shape option and answering Yes to the "Make the overlapped regions of objects with teh same fill style empty?" question.

Text and Button objects with Target checkbox checked.
These objects are wrapped within a movie clip to allow standard movie clip properties to be accessed. If you wish to access the underlying properties associated with the text object or button you should use the _text or _button property.

Learn to use the Autoshape and Subselection tools.
The autoshape tool allows you to create a variety of useful shapes such as rounded rectangle, heart, arrow, star, polygon, 3D cube, bevelled button and rounded bevelled button. The subselection tool can be used to further modify the appearance of an existing shape. The subselection tool allows the the item to be sliced, stretched, and anchor points / vertices to be added, deleted or modified. For example, you can use the subselection tool to create a circle segment.

Sound does not play on the first frame - SWF8, possibly later versions.
This is by design to prevent the playback of a snippet of sound during a stop() action on the first frame of a movie. The workaround is to place a sound on the second frame of a movie.

HTML text has missing spaces.
SWF7 and earlier may not preserver spaces in HTML text.
For example:
<B>This is</B> very <U>interesting</U> text
<B>This is</B> <U>very interesting</U> text
This <I>is</I> <U>very</U> <B>interesting</B> text
when exported as SWF7 or earlier give incorrect results with spaces lost:
This is very interesting text
This isvery interesting text
This isveryinteresting text

Button events.
Button events such as on (press) by default refer to the parent object. See Button Events for more information.

Drawing Circles and Squares.
Circles and Squares can be drawn by holding the shift key down as you size the object using the Ellipse or Rectangle tool.

Components.
Have a look at the supplied components. They provide a useful set of items that can be easily added to your movie with minimal configuration.

Movie control actions are not immediate.
Movie control actions such as gotoAndPlay() do not move the play head immediately. All actions associated with the current frame are completed before the play head moves to the nominated position.
For example:

```
onFrame (30) {
  gotoAndPlay(1);
  trace("frame30");
}
```

will cause "frame30" to be written to the debug panel on frame 30. ie. The play head does not move until frame 30 is complete.

Tutorials

Chapter 8

8 Tutorials

SWiSHzone.com offers a series of screencast and written tutorials to allow the new user to work through a lesson material and become familiar with SWiSH Max.

Flash Site Template
SWiSH Max includes a full Flash Web Site template. This template is useful to demonstrate the use of pre-loaders, loading sections and other advanced authoring techniques possible with SWiSH Max or the template can simply be customized and used as a personal website. As well as the step-by-step instructions available here, a screencast is available to demonstrate customization.

- Step by Step Setup Instructions
- Screencast tutorial

Screencast Tutorials
Check out the online screencast tutorials. Screencast tutorials are movies showing an actual PC screen to demonstrate many aspects of SWiSH Max including scripting and component development.

Scripting Tutorials
Scripting tutorials are a series of written tutorials covering the scripting language and how to apply scripting to a SWiSH Max movie. Also refer to Scripting for an introduction to basic scripting concepts.

Component Tutorials
Component tutorials are a series of written tutorials covering how to use and write components. Also refer to Scripting and Scripting tutorials for an introduction to basic scripting concepts.

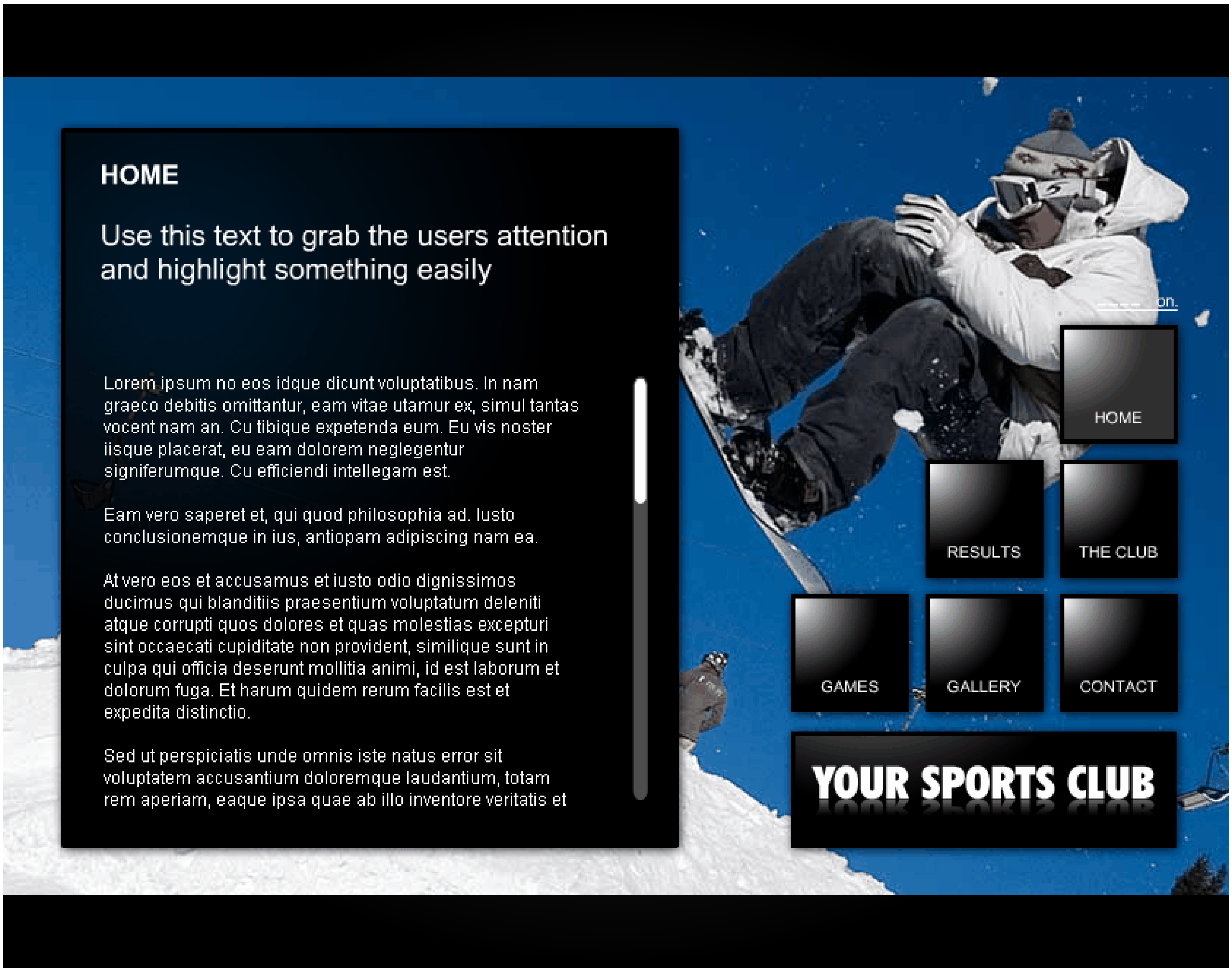
8.1 Flash Web Site Template

8.1.1 Sample Site - Sports Club

Step-by-Step Tutorial for setting up Sample Site - Sports Club using SWiSH Max2

This template includes 7 standard SWIs and 2 HTML pages. The site includes templates featuring styles for a gallery, a page with multiple sub-section links and a contact page.

As supplied, the site has 6 links in the menu (Home, Results, The Club, Games, Gallery and Contact) which each display a single page. These menu items, and all text displayed, are fully customizable. The Contact section has hot links for site and email contact details.



HOME

Use this text to grab the users attention and highlight something easily

Lorem ipsum no eos idque dicunt voluptatibus. In nam graeco debitis omittantur, eam vitae utamur ex, simul tantas vocent nam an. Cu tibique expetenda eum. Eu vis noster iisque placerat, eu eam dolorem neglegentur signiferumque. Cu efficiendi intellegam est.

Eam vero saperet et, qui quod philosophia ad. Iusto conclusionemque in ius, antiopam adipiscing nam ea.

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et

This Help File was made with EC Software Help & Manual.
The installer was created using MindVision Installer VISE.

Sample Site - 'Sports Club' User's Guide
Release: First Edition - July 2008 (SWiSH Max2 edition)

8.1.2 Getting Started

This guide will give you information for customizing your site template. It is assumed that you have a basic knowledge of SWiSH Max, that you know how to change text, import objects and sounds, copy and paste objects and that you can add and delete frames. It is also assumed that you have basic knowledge of photo editing and know how to resize, crop and optimize images.

For anyone who has completed the 'My First Site' screencast tutorials (see SWiSH Max2 Screencast Tutorials) or the 'Your First SWiSH Movie' tutorial series included with SWiSH2/SWiSH Max (version 1), customizing this template should be fairly simple. You do not need to change any of the actions that make this site work. The only Actions you will need to change or add are for links to external pages, files and email addresses.

These instructions include information on changing the text, logo and background music. If you are more familiar with SWiSH Max, you can make additional changes to the colors, sound effects, etc.

Backup, backup, backup

Save a backup copy of the installer that you downloaded, or individual files that were initially installed. You can revert to this copy if needed. Save and test frequently. It is also a good idea to 'Save As' after testing each change. This way if you make a mistake you can go back to your last saved version of the .swi file rather than starting from the beginning, or retracing your last steps.

Test, test, test

After each change, save and export the .swf file for testing. Due to the interdependence of the .swf files you can not preview the .swi within SWiSH Max. Export your Movie to the **UPLOAD** folder by selecting 'File | Export | SWF' (you do not need to export the HTML). You must export each Movie **without changing the file name**. When you export, a warning dialog box will notify you that the file already exists and ask you if you want to replace it. Click on 'YES' to overwrite the original file.

Support

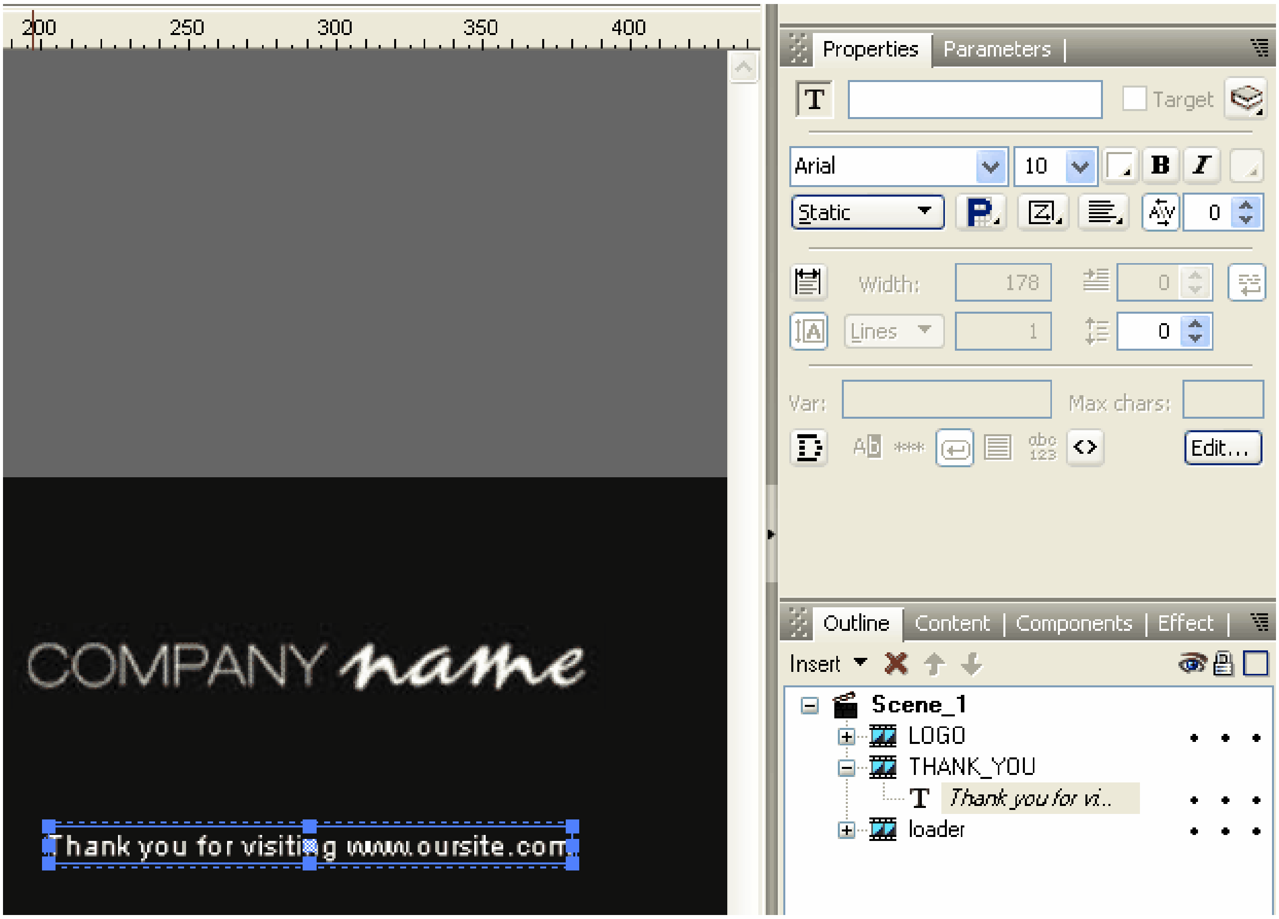
If you have any trouble please refer to the Help Manual supplied with SWiSH Max and don't forget that there is a lot of support available through the growing community of Support Forums for SWiSH Max and Site templates.

Conventions used in this document

Throughout this manual we refer to Menu selection options in the following format - Edit | Copy (Ctrl+C). This means to select the main titlebar 'Edit' Menu and from within it, the 'Copy' option. An equivalent quick key sequence is performed by pressing the Control key (Ctrl) and the C key together.

The location, dimensions and other information regarding objects to be customized have been inserted between the brackets [...] for anyone familiar enough with SWiSH Max to customize this template without further instructions.

Locations of items are referred to (where appropriate) in the 'Outline' panel in the following format - e.g. [Location: Scene_1 / Preloader / THANK_YOU/ T "Thank you for visiting www.oursite.com" "].



Further [Location: Scene 1/**1-5**/T] denotes Text items in '1' through to '5'.

Terms used in this document

Site, template, and **Movie** are all used interchangeably to refer to the final output and the group of source files
Menu Text refers to the text used in the template's links
Title Text refers to the text that appears on each different page to let the visitor know where they are on your site
Info Text refers to the text on the different 'pages' within your Movie/site.

8.1.3 QUICKSTART

If you like our choice of music, images, and menu selections, you'll still need to add some content (info text), your logo and contact details so these are the minimum steps to get you going:

Step 1: Change the info text
Step 2: Change the logo
Step 3: Change the Contact Info
Step 4: Change the index.html and mainmenu.html
Step 5: Export your Site template

If you want to be more creative you can change the appearance of the site, the music, the content menus and titles as well as the number of pages and sections.

8.1.4 Overview

This Site template template includes 7 standard SWIs, 2 HTML pages, and an UPLOAD folder with dummy SWFs to aid in the testing and uploading of your site.

The site consists of a main movie and uses the load movie feature to open the different pages. When menu item 1 (currently 'Home' in main.swi) is clicked, it will call for a movie named **section1.swf**, when menu item 2 (currently 'Results' in main.swi) is clicked, it will call for a movie named **section2.swf**, etc. (see Changing the title text) It also includes a smart preloader which will preload section1.swf thru section6.swf while your main movie is playing so your content will load in the background and speed your site.

If you are familiar with SWiSH Max2 and merely want to locate all the editable text in the files the following list provides references to changes necessary (see Getting Started for an explanation of terminology).

main.swi

This is the main movie; the main menu [Location: main/Scene_1/MENU/MENU_1 - MENU/MENU_6], logo/company name [location: main/Scene_1/logo] and background music [location: Frame 1 in the 'audio' Sprite in Scene_1] are in this file.

section1.swi

This movie includes some scrolling text [Location: section1/Scene_1/section/section/text_1] and a section image [location: section1/Scene_1/section/image - dimensions: 840w X 560h]

section3.swi, section4.swi and section5.swi

are additional templates for creating a page with multiple sub-section links, a photo gallery, and a contact page

index.html

This page is the first page a visitor will see when they visit your site (provided you have placed it in the main directory on your server). It contains a the spot for your company logo image (in this instance named logo.png). There is also a link to the main.swf and a javascript that will open your main movie (main.swf) in a pop-up window. See here for more details.

mainmenu.html

This is the page that your main movie (main.swf) will be embedded in. See here for more details.

UPLOAD folder

This folder has been included to help you test and upload your files to your server. It includes both HTML pages and a SWF for each SWI that you will need to upload. You should export your movies to this folder, overwriting the included SWFs.

You can open **index.html** (from your browser) from within this folder, and click on the **ENTER** link to test your site anytime after you have customized and exported main.swi, or any other SWI file.

NOTE: Additional templates for creating a photo gallery, a page with multiple sub-section links and a contact page are included (in the 'templates' folder) for intermediate and advanced users. See Templates.

Save and Test your changes.

8.1.5 Fonts used

The Arial font has been used for all customizable text. You can select any font, but may need to adjust the font's size for proper fit.

_sans is used for the **Section** text in the other sections
Arial 20 is used for the section **Heading** text in all sections
Arial 18 is used for the **Title** text

8.1.6 Changing the Menu Text

The **menu text** is the information which is displayed in the interactive menu, as supplied this is 'Home', 'Results', 'The Club', 'Games, 'Gallery' and 'Contact'

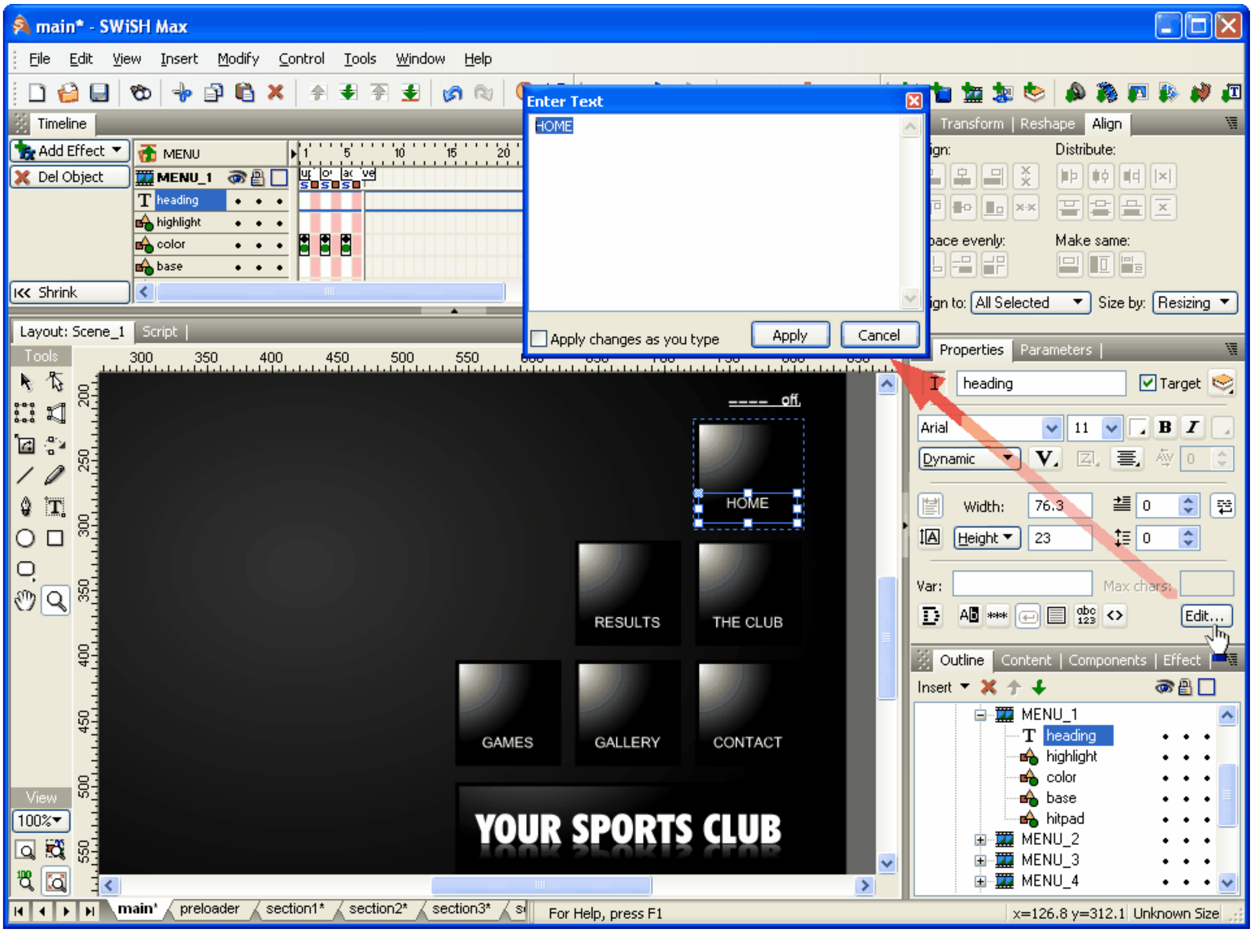
If you want to keep the same menu options ('Gallery One', 'Gallery Two', 'News', 'About', 'Services' and 'Contact'), you need not change anything here, and can continue to changing the section pages.

The main menu text can be found in **Scene_1** of **main.swi** within the **MENU/MENU_1/heading thru MENU/MENU_6/heading** objects.

What you need to do

1. Open **main.swi**
2. Select the 'heading' object from the Outline Panel (as illustrated below).
3. On the Properties Panel, click the [Edit] button to open the text editor and update the current text with your own. You can also use the Text Tool on the Tools panel to directly edit the text on the stage.

4. Repeat this process for the 'heading' object in the remaining MENU_1 - MENU_6 Sprites.



When you are happy with your changes, save this movie and export your movie as follows:

5. From the File Menu, select 'File | Export | SWF'.

Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

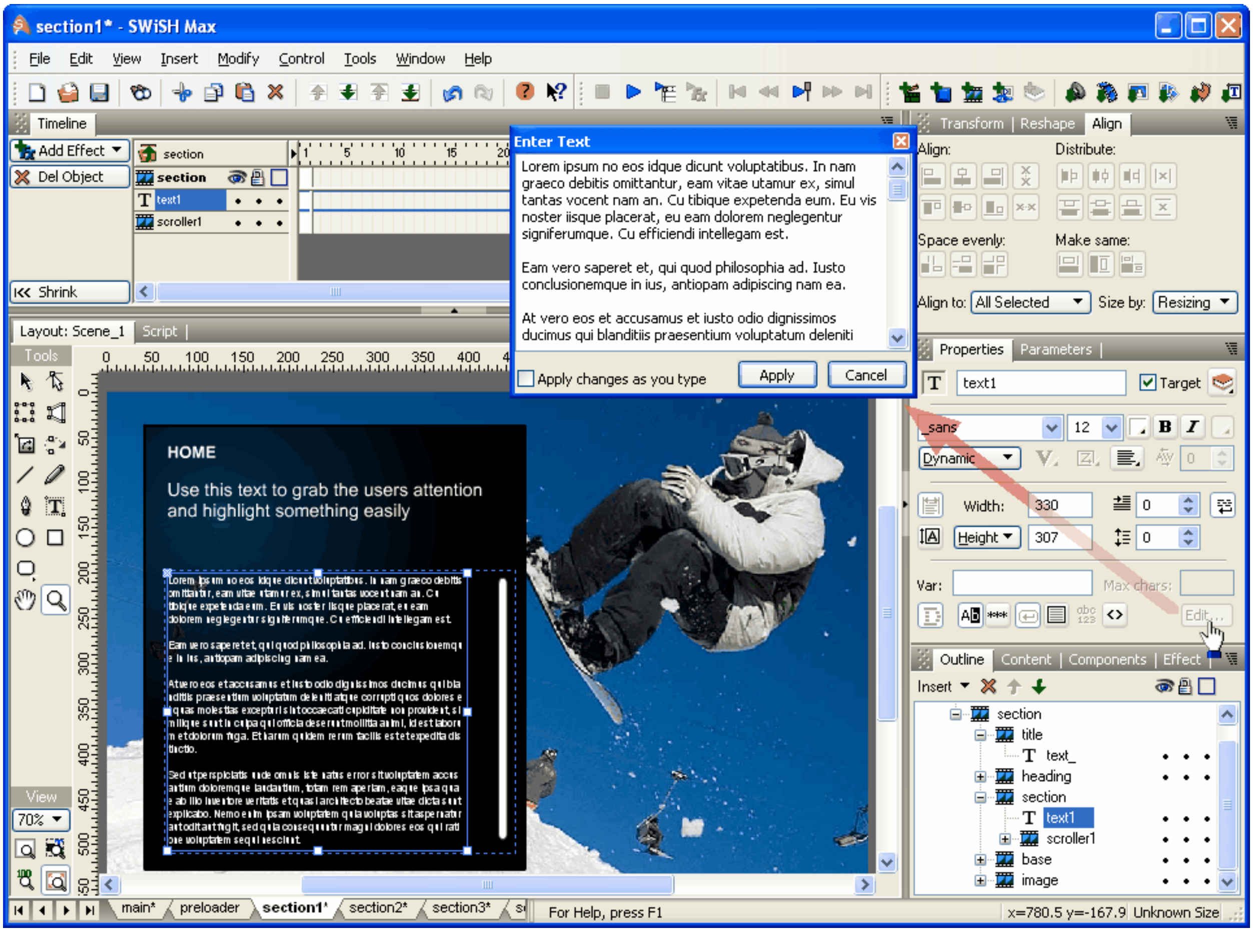
8.1.7 Changing the Info Text

The **info text** is the information displayed on the page when you select a menu link (example is shown below). As supplied, a single page of text is displayed for 'News', 'About' and 'Services'. **Note:** If you are familiar with SWiSH Max2 and want to use a different style page of information like a photo gallery, you can replace movies section1.swi thru section6.swi with these movies by overwriting any of these single files, by 'saving as' (File | Save As).

Changing the Info text

What you need to do

1. Select the 'text1' Text Object from the 'Outline' Panel (as shown below)
2. On the Properties Panel, click the [Edit] button to open the text editor and update the current text with your own. You can also use the Text Tool on the Tools panel to directly edit the text on the stage.



section1* - SWiSH Max

Enter Text

Lorem ipsum no eos idque dicunt voluptatibus. In nam graeco debitis omittantur, eam vitae utamur ex, simul tantas vocent nam an. Cu tibique expetenda eum. Eu vis noster iisque placerat, eu eam dolorem neglegentur signiferumque. Cu efficiendi intellegam est.

Eam vero saperet et, qui quod philosophia ad. Iusto conclusionemque in ius, antiopam adipiscing nam ea.

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti

□ Apply changes as you type Apply Cancel

HOME

Use this text to grab the users attention and highlight something easily

Properties: text1 ☑ Target; _sans 12; Dynamic; Width: 330; Height 307

3. From the File Menu, select 'File | Export | SWF'.

Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

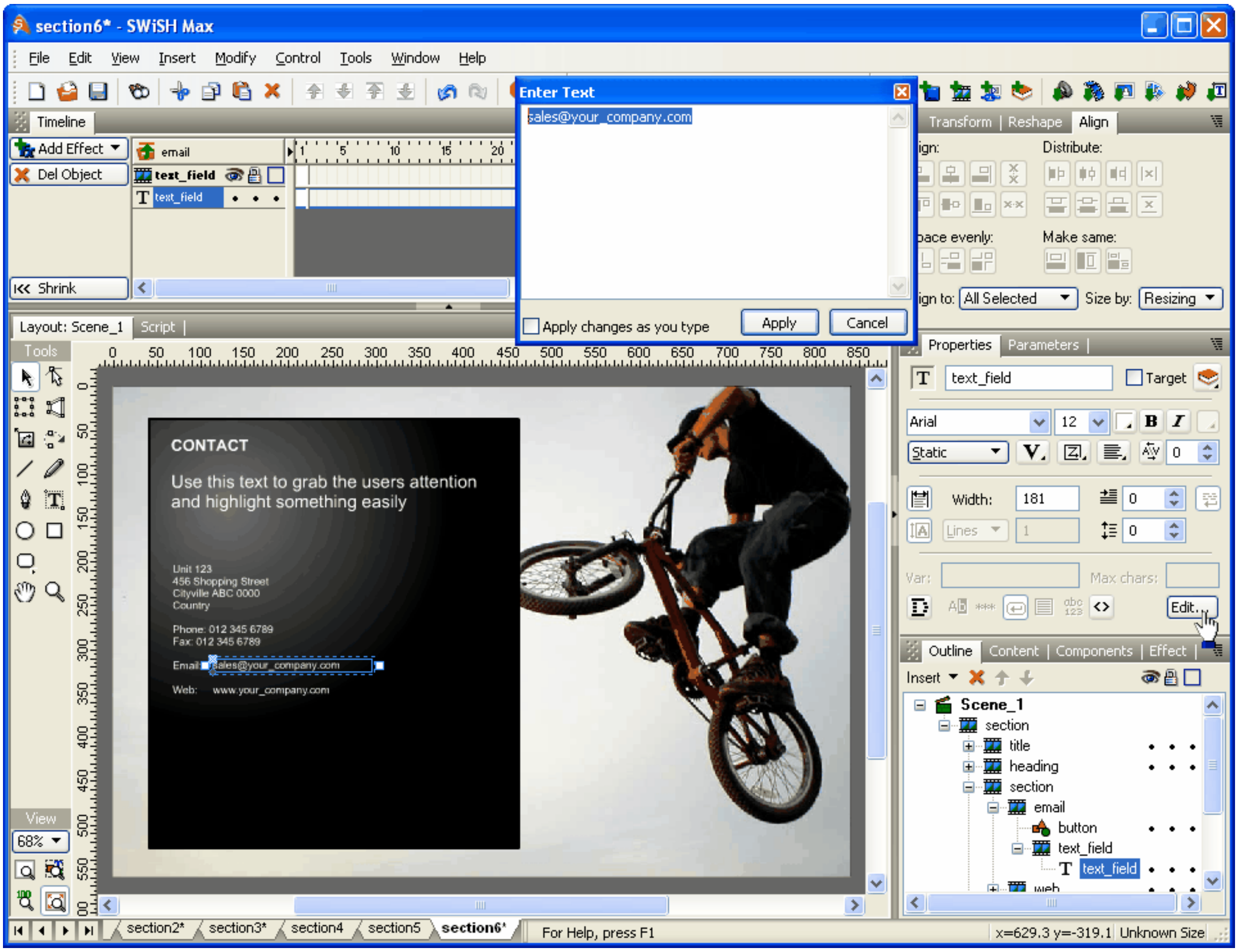
8.1.8 Changing the Contact Info

The Contact Information (address, email, URL etc.) is in the **'section'** Sprite in **section6.swi** and as a copy named 'contact.swi' in the templates folder

To change the **email address,** you need to replace both the address for the Text Object and the address for the On (Release) Mailto Action attached to the 'button' shape object.

What you need to do

1. Open section6.swi
2 Select the email information Text object in the Outline Panel tree as shown below. In the Text Object Properties Panel, and click the [Edit] button to update the current text with your own, or using the Tools panel's 'Text Tool' directly edit the text on the stage.



section6* - SWiSH Max

Enter Text

sales@your_company.com

□ Apply changes as you type

CONTACT

Use this text to grab the users attention and highlight something easily

Unit 123
456 Shopping Street
Cityville ABC 0000
Country

Phone: 012 345 6789
Fax: 012 345 6789

Email: sales@your_company.com

Web: www.your_company.com

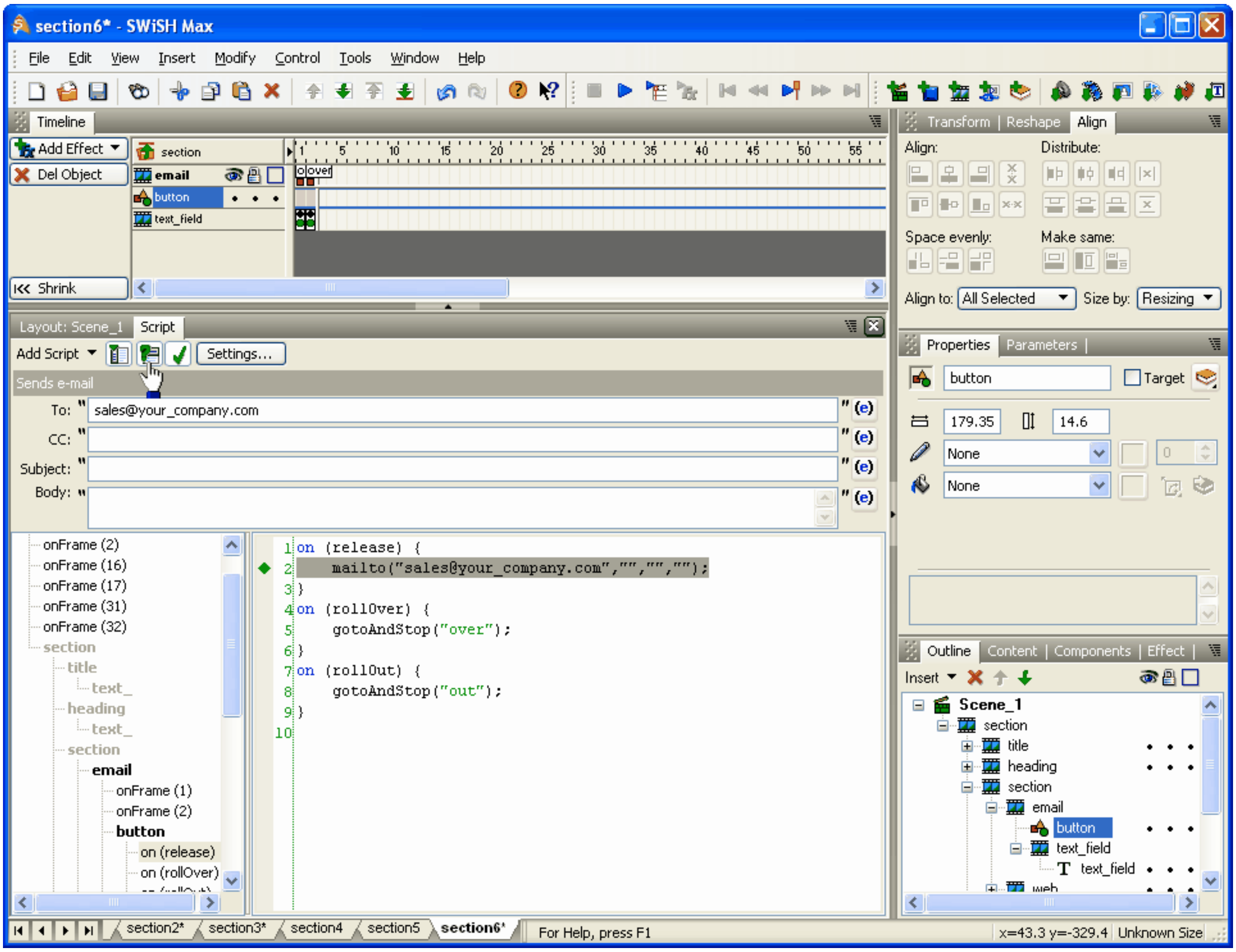
Properties: text_field; Arial, 12, Static; Width: 181

Outline:
- Scene_1
  - section
    - title
    - heading
    - section
      - email
        - button
        - text_field
          - text_field
      - web

For Help, press F1

x=629.3 y=-319.1 Unknown Size

3. Edit the email button properties as shown below by selecting the button object, opening the 'Script' Panel and selecting the GetURL action. Change the address in the 'URL' address edit box (in the 'assist' pane) from sales@your_company.com to your email address.
NOTE: you will need to make sure the 'assist' pane option is open on the 'Script' Panel.



Align to: All Selected

Size by: Resizing

To: "sales@your_company.com" (e)

CC: "" (e)

Subject: "" (e)

Body: "" (e)

```
on (release) {
    mailto("sales@your_company.com","","","");
}
on (rollOver) {
    gotoAndStop("over");
}
on (rollOut) {
    gotoAndStop("out");
}
```

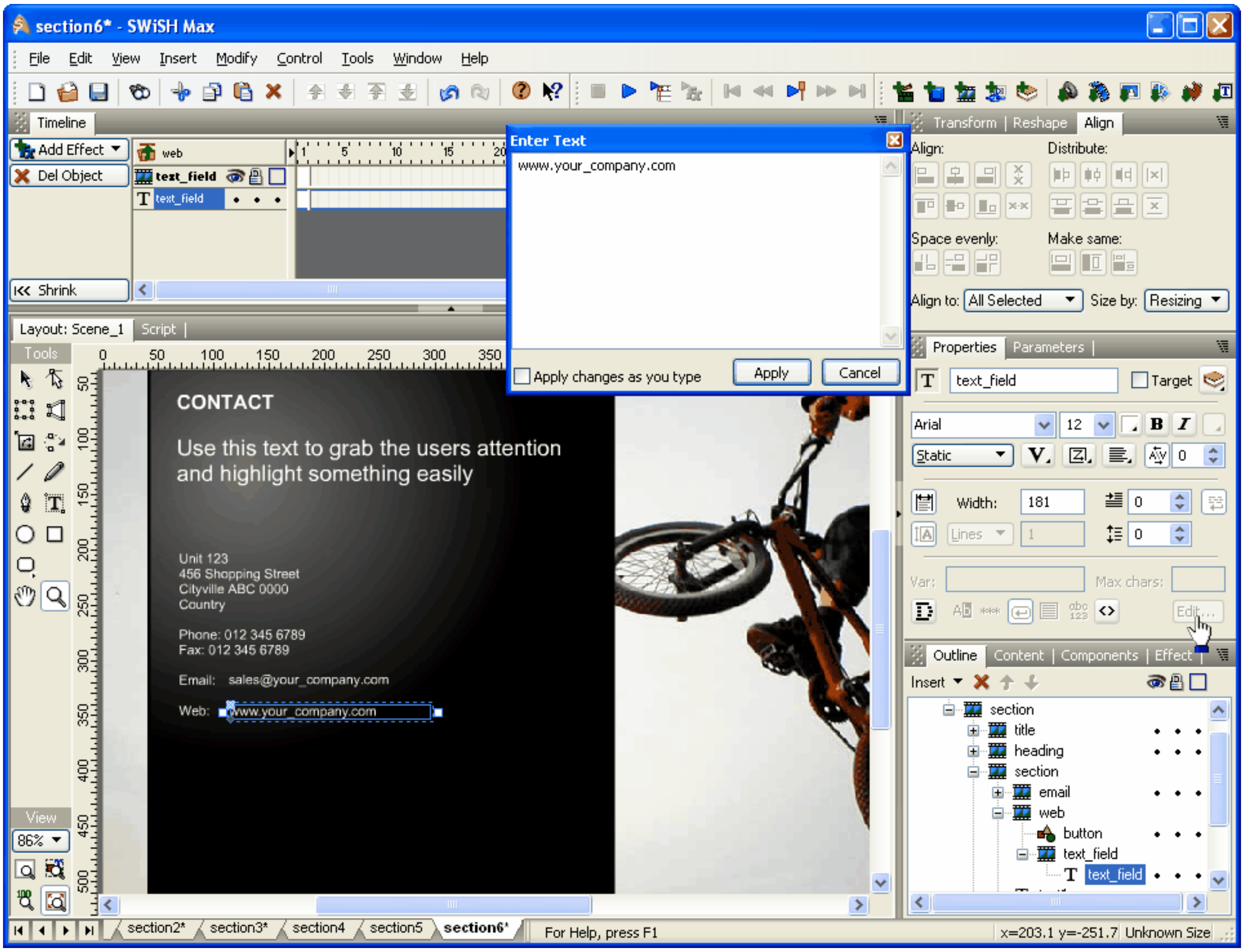
Properties: button □ Target

179.35 14.6

None 0

None

4. Select the web address Text object in the Outline Panel tree as shown below.
In the Text Object Properties Panel, click the [Edit] button to update the current text with your own, or using the Tools panel's 'Text Tool' directly edit the text on the stage.



Enter Text

www.your_company.com

□ Apply changes as you type | Apply | Cancel

Properties: T text_field; □ Target; Arial, 12, B, I; Static; Width: 181; Lines 1; 0; 0

CONTACT

Use this text to grab the users attention and highlight something easily

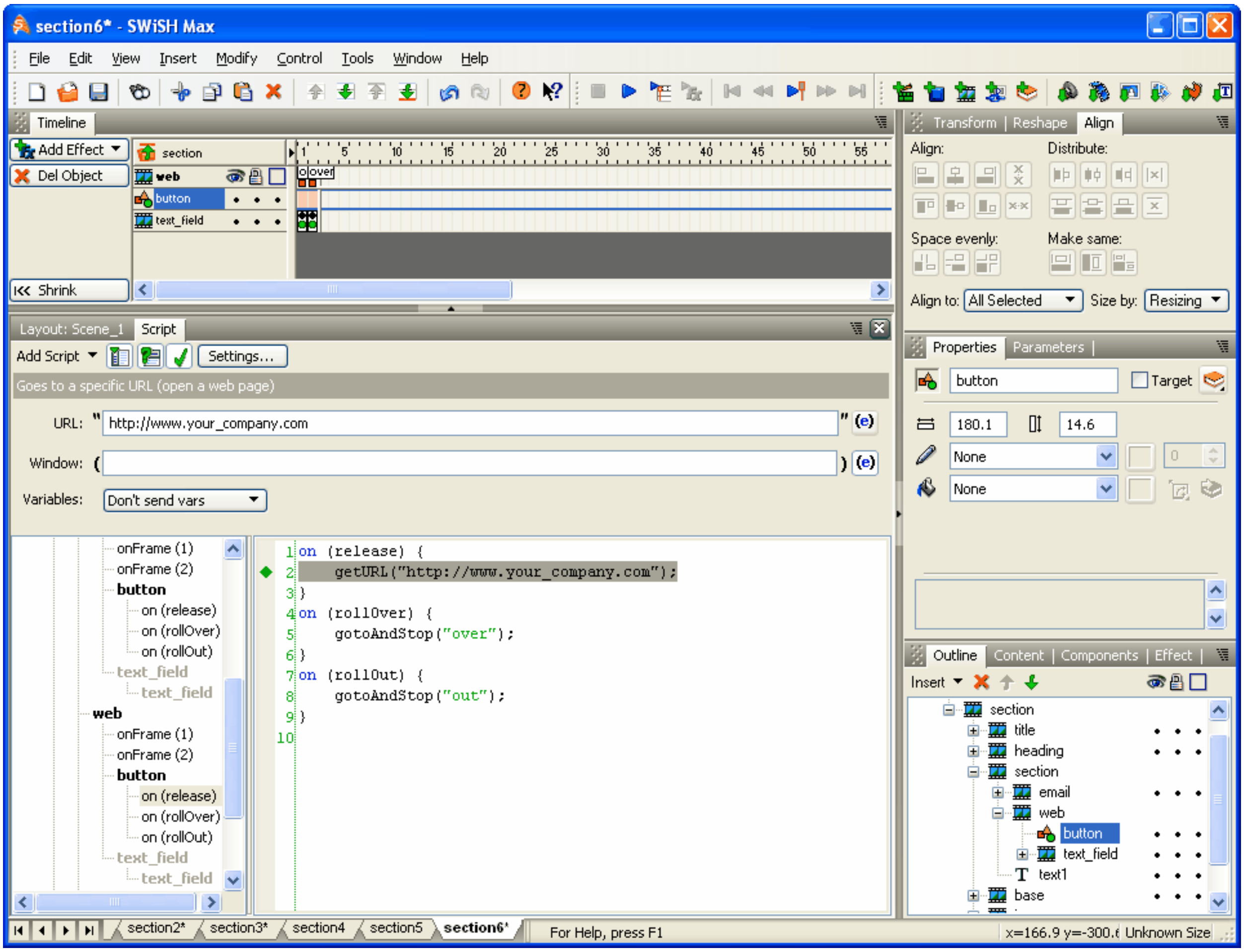
Unit 123
456 Shopping Street
Cityville ABC 0000
Country

Phone: 012 345 6789
Fax: 012 345 6789

Email: sales@your_company.com

Web: www.your_company.com

5. Edit the web address button properties as shown below by selecting the button object, opening the 'Script' Panel and selecting the 'on (release)' action. Change the web address in the URL edit box from http://ww.your_company.com to your web address. NOTE: you will need to make sure the 'assist' pane option is open on the 'Script' Panel.



Goes to a specific URL (open a web page)

URL: "http://www.your_company.com"

Window: ( )

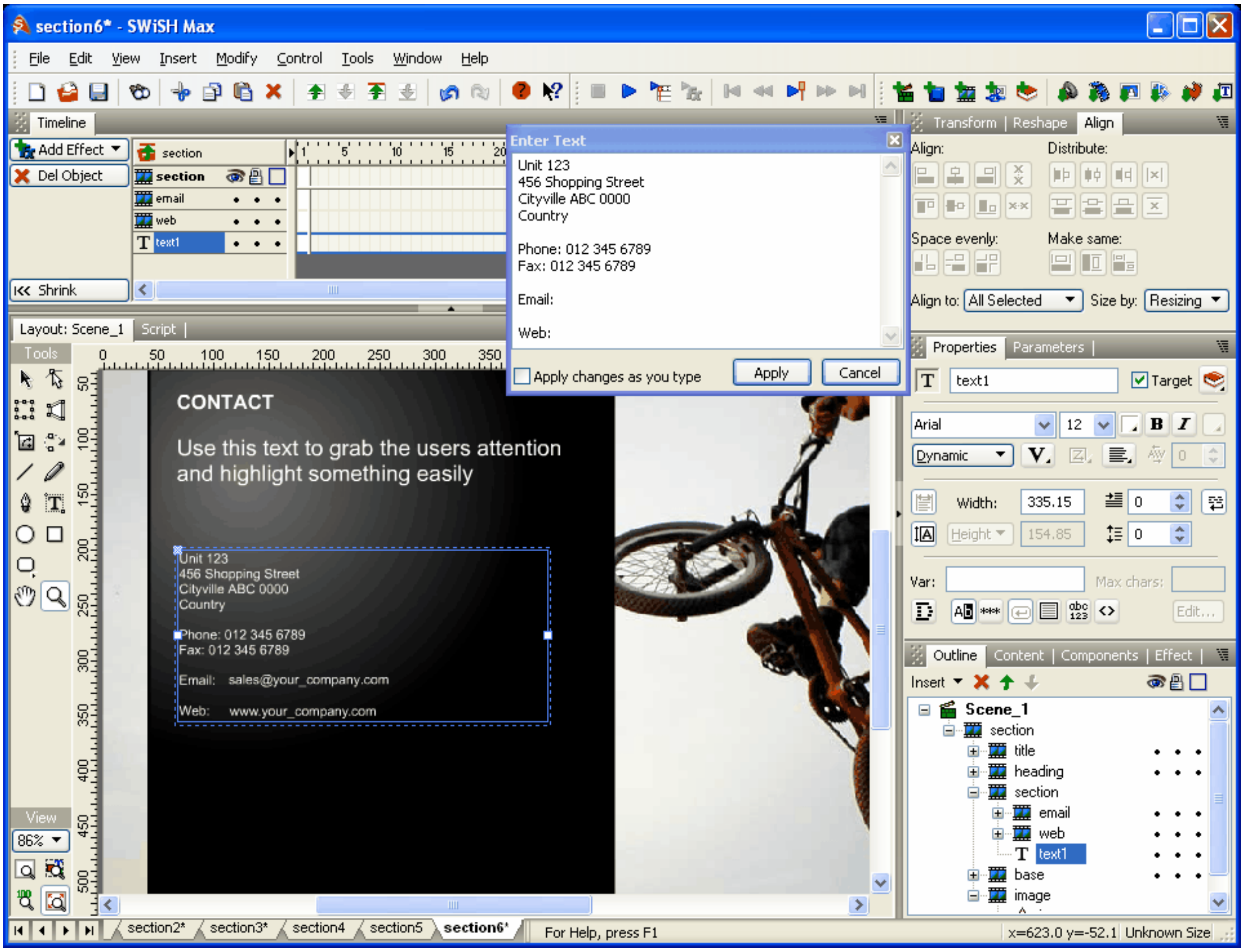
Variables: Don't send vars

```
on (release) {
    getURL("http://www.your_company.com");
}
on (rollOver) {
    gotoAndStop("over");
}
on (rollOut) {
    gotoAndStop("out");
}
```

Properties: button □ Target; 180.1; 14.6; None; None; 0

For Help, press F1 x=166.9 y=-300.(Unknown Size

6. Select the Contact details (address, phone number, etc.) Text object (text1), and, on the Properties Panel, click the [Edit] button to update the current text with your own, or using the Tools panel's 'Text Tool' to directly edit the text on the stage.



Enter Text

Unit 123
456 Shopping Street
Cityville ABC 0000
Country

Phone: 012 345 6789
Fax: 012 345 6789

Email:

Web:

□ Apply changes as you type | Apply | Cancel

CONTACT

Use this text to grab the users attention and highlight something easily

Unit 123
456 Shopping Street
Cityville ABC 0000
Country

Phone: 012 345 6789
Fax: 012 345 6789

Email: sales@your_company.com

Web: www.your_company.com

Properties: text1; Arial, 12; Dynamic; Width: 335.15; Height: 154.85; Var: ; Max chars:

Save this movie and export your movie as follows:

3. From the File Menu, select 'File | Export | SWF'.

Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

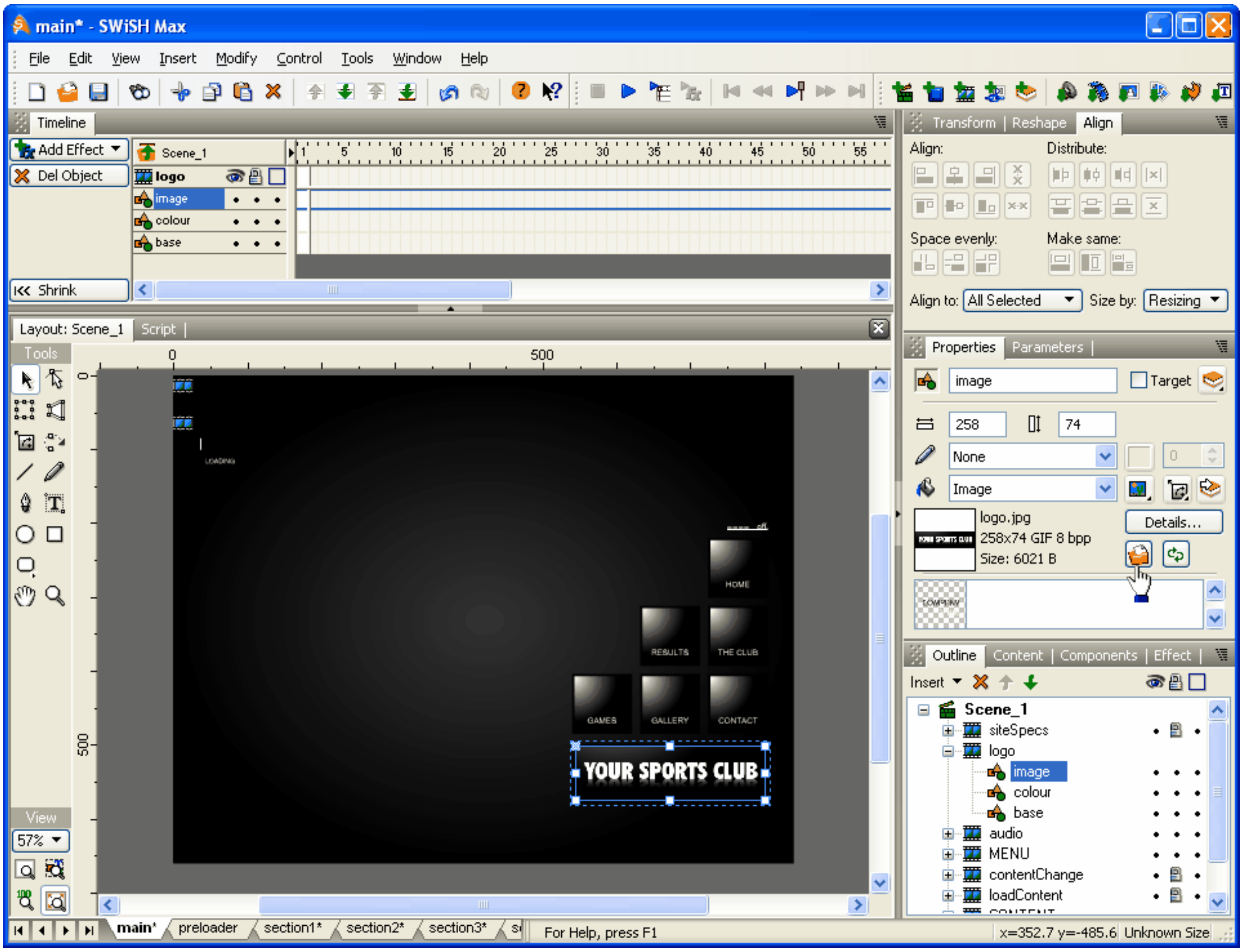
8.1.9 Changing the Logo

The logo or company name can be found in the **Scene 1** of **main.swi** (as shown below).

You can add your own logo image in any accepted format (jpg, gif or png). The placeholder example in this template is a jpg file with dimensions of 258px X 74px.

What you need to do:

1. Open **main.swi**.
2. Select and edit the **logo** image as shown below.
3. In the Properties Panel, select the folder icon and browse to find your logo.



main* - SWiSH Max

Align to: All Selected

Size by: Resizing

Layout: Scene_1

View: 57%

Properties: image

258 74

None

Image

logo.jpg
258x74 GIF 8 bpp
Size: 6021 B

For Help, press F1

x=352.7 y=-485.6 Unknown Size

When you are happy with your changes, save this movie and export your movie as follows:

3. From the File Menu, select 'File | Export | SWF'.

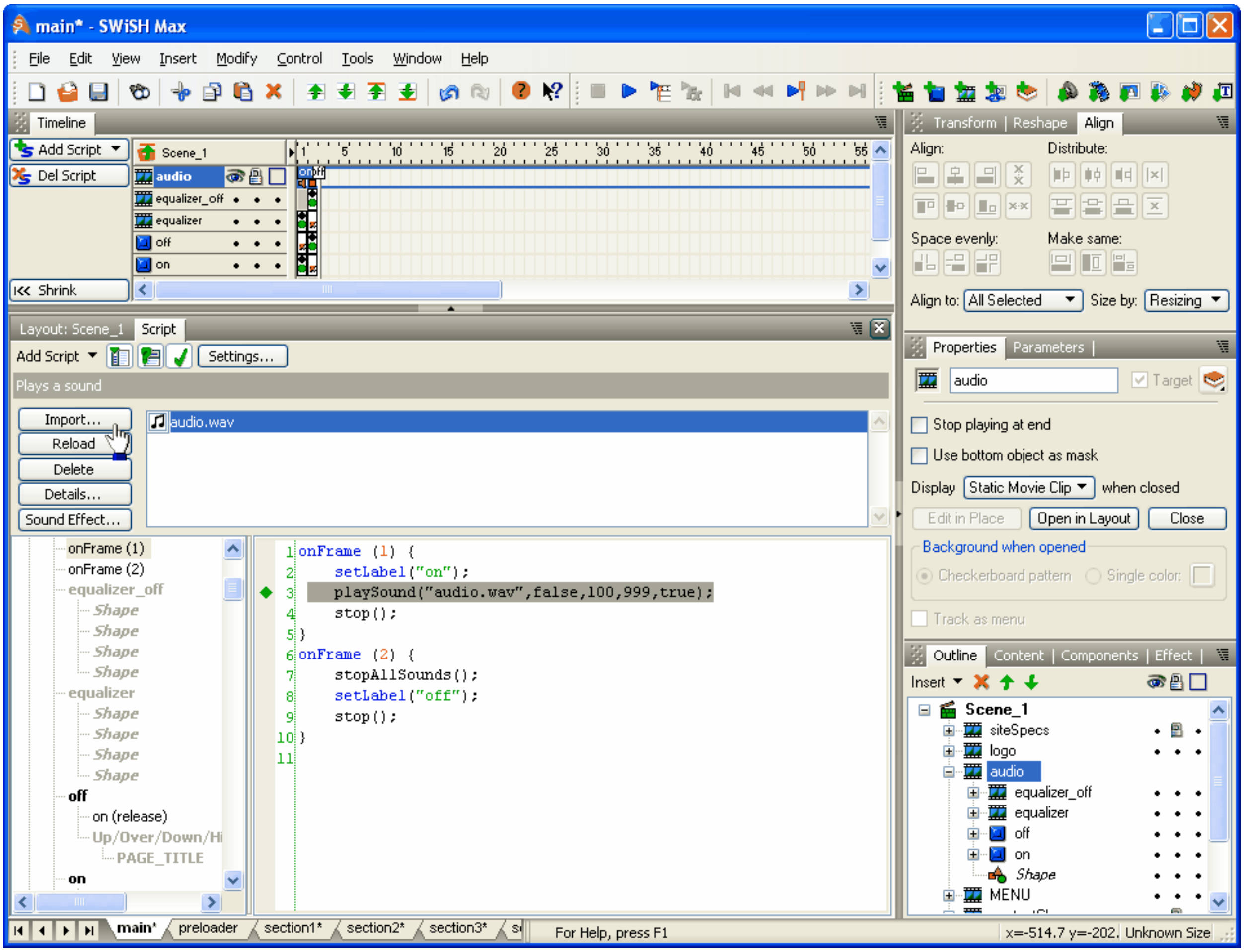
Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

8.1.10 Changing the Background Music

If you like the music we supplied, skip this step.

What you need to do

1. Open **main.swi**
2. From within the **Scene_1/audio** sprite, select **Frame 1**, as shown below
3. From the 'Script' Panel, select the '**playSound ("audio.wav" ...)**' Action from the Script Panel. Press the Import button then locate and import your desired loop, as shown below



```
onFrame (1) {
    setLabel("on");
    playSound("audio.wav",false,100,999,true);
    stop();
}
onFrame (2) {
    stopAllSounds();
    setLabel("off");
    stop();
}
```

Note: You can access the 'Sound Effect' dialog (click the 'Sound Effect' button) to adjust the volume.

Tips

You should not use full songs for background music, as this will greatly increase the file size of your Movie. As a rule of thumb, import short loops of less than 8-10 seconds. If you select large sound files, you may loose many visitors who will be forced to wait a long time for your Movie to begin.

It is best to import .wav files for use as background music in your Movies and leave the default MP3 compression, as it will result in the same .swf file size as the MP3 file with better sound quality.

When you are happy with your changes, save this movie and export your movie as follows:

4. From the File Menu, select 'File | Export | SWF'.

Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

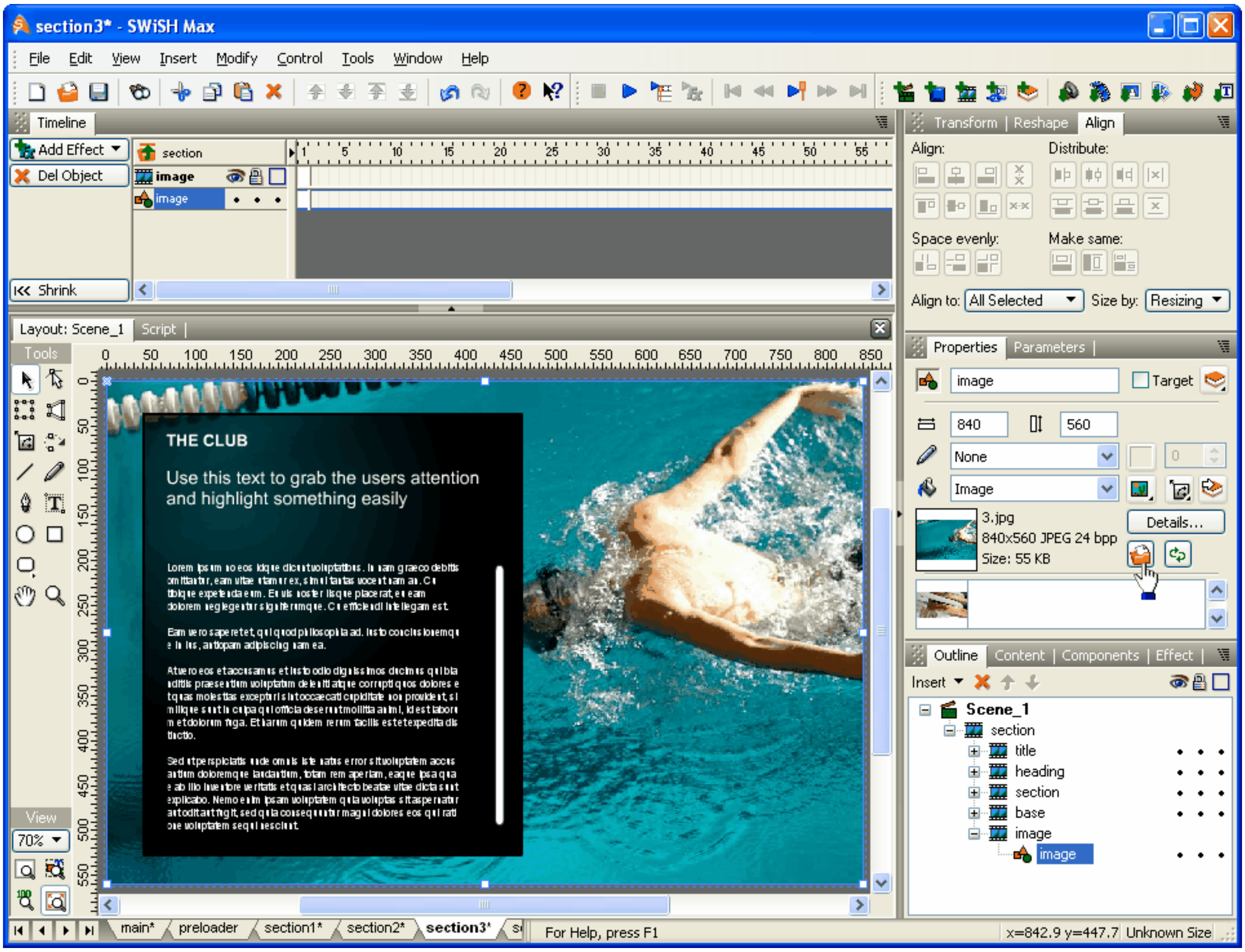
8.1.11 Changing the Images

If you like the images we supplied, skip this step.

Changing the SECTION Images

The current images shown for each **Section** can be replaced with your own images. These images are located in each of the section.swi templates.

You will find them via the **Scene_1/section/image/image object** as shown below. Any replacement images you use will need to be the same size as the current images i.e. 840px x 560px.



section3* - SWiSH Max

THE CLUB

Use this text to grab the users attention and highlight something easily

Layout: Scene_1

Align to: All Selected Size by: Resizing

Properties: image □ Target 840 × 560 None Image

3.jpg
840x560 JPEG 24 bpp
Size: 55 KB

For Help, press F1 x=842.9 y=447.7 Unknown Size

What you need to do

1. Select the **image object** in the **Scene_1/section/image/image** Sprite from the 'Outline' Panel (as shown above)
2. With the image object selected, click on the folder icon in 'Shape' Panel to display the 'Open' dialog box
3. Navigate to the folder that you have saved your image in, select your new image, and import it

Note: The image that you replaced was not compressed within SWiSH.

To reduce file size of the SWF, you may wish to compress your new image within SWiSH. To recompress your image, press the 'Properties' button located next to the 'Clipped image' button to access the 'Image Properties' dialog box. Within the 'Image' dialog box, check the 'Recompress JPEG' checkbox try a value of 50% and test the movie (in the browser or player) to see whether you need to adjust this new value. You can increase the value for better image quality (larger file size), or decrease this value for poorer image quality (smaller file size).

When you are happy with your changes, save this movie and export your movie as follows:

5. From the File Menu, select 'File | Export | SWF'.

Note: Please read important information on testing and exporting your site in 'Exporting your Site template'.

8.1.12 Gallery and other Templates

Template files

To alter the structure of the menu selections just replace the appropriate section.swi with one of the templates listed below and edit the content appropriately. For example to replace the section1.swi page to a page with mutiple sub-sections page, open **section2.swi** and 'Save As' section1.swi to overwrite section1.swi, edit the text, and export as section1.swf. Menu item 2 now references a page style with a number of sub-menu text items.

NOTE: ***All available page styles for this sample site are included in the default site structure, our commercial templates include a number of simlar alternative page styles in a seperate 'templates' folder.***

Special Note: We can only offer limited support for site templates when they are altered from the format as supplied. We recommend you become totally familiar with the site template as described in previous sections before attempting to modify the page layouts. These instructions are intentionally succinct presuming experienced operators.

Adding/deleting Menu items/titles/sections

Unfortunately although you can expand the number of pages in each of the menu items, it is not easy to add or remove main menu items. Also, though some Site templates would look ok, others will look off balance, so it is probably best to leave all the items and put a 'Coming Soon' or 'Under Construction' message in sections for which you have no content at this stage.

contact.swi

As per section6.swi supplied. This movie contains contact information, a **web** link and an **email** link. Edit text within this file using instructions given in 'Changing the Contact Info'.

multiple.swi

As per section2.swi supplied. This movie contains a submenu for 4 pages [Location: Scene_1/section/CONTENT/CONTENT_1 thru CONTENT_4], and info text [Location: Scene_1/CONTENT/CONTENT_1/TEXT_1/TEXT_1 thru CONTENT_4/TEXT_1/TEXT_1] for these links.

You can replace Movies section1.swi - section4.swi with this movie by overwriting any of these single page files, by 'Saving As' (File | Save As).

The MENU_1 - MENU_4 Sprites contain the buttons and numbers, and the CONTENT_1 thru CONTENT_4 Sprites contain the Info text.

The button object in:
MENU_1 is associated with CONTENT_1
MENU_2 is associated with CONTENT_2
MENU_3 is associated with CONTENT_3
MENU_4 is associated with CONTENT_4

Changing the info text

1. Select the CONTENT_1/TEXT_1/TEXT_1 Sprite and from the Sprite Panel click the 'Open in Layout' button to view the text.
2. Select the TEXT_1 text object from the Outline Panel.
3. Select the Text Panel and replace the current text with your own.
4. Repeat steps 1, 2, and 3 for the TEXT_FIELD objects in the remaining Sprites.

Save and Test your changes.

If you do not need all 4 pages, you can delete the unused 'MENU(+number)' Sprites and the associated 'CONTENT(+number)' Sprites.

gallery.swi

This Movie contains 15 images [Location: Scene_1/section/IMAGES/IMAGE1/IMAGE thru IMAGE15/IMAGE], thumbnail images for the 15 images [Location: Scene_1/section/THUMBS/THUMB1/image thru THUMB15/image]. You can replace Movies section1.swi - section5.swi with this Movie by overwriting any of these single files, by 'saving as' (File | Save As).

Changing the Images

These images are 386px X 294px. Any images you use will need to be at least this size. When preparing your images prior to importing, resize and/or crop as needed so that the image is close to the dimensions required to avoid making the file size of this Movie larger than necessary.

These images can be found in Scene 1 within the IMAGES/IMAGE1/IMAGE through IMAGE16/IMAGE Sprites, they are named **image**.

What you need to do

1. Locate and select the **image object** from the 'Outline' Panel
2. With the **image object** selected, click on the folder icon in the 'Shape' Panel to display the 'Open' dialog box
3. Navigate to the folder that you have saved your image in, select your new image, and import it
4. Repeat steps 1-3 for the images in the remaining Sprites

Note: The image that you replaced was not compressed within SWiSH.

To reduce file size of the SWF, you may wish to compress your new image within SWiSH. To recompress your image, press the 'Details...' button on the Properties Panel to access the 'Image Details' dialog box. Within the 'Image Details' dialog box, on the Export tab, you can change the Image Quality. Test the movie (in the browser or player) to see whether you need to adjust this new value. You can increase the value for better image quality (larger file size), or decrease this value for poorer image quality (smaller file size).

Changing the Thumbnail Images

These images are 68px X 44px. Any images you use will need to be at least this size. When preparing your images prior to importing, resize and/or crop as needed so that the image is close to the dimensions required to avoid making the file size of this Movie larger than necessary.

These images can be found in Scene_1/section/THUMBS/THUMB1/image through THUMB15/image Sprites, they are named **image**.

What you need to do

1. Locate and select the image from the 'Outline' Panel
2. With the **image object** selected, click on the folder icon in the 'Shape' Panel to display the 'Open' dialog box
3. Navigate to the folder that you have saved your image in, select your new image, and import it
4. Repeat steps 1-3 for the images in the remaining Sprites

Note: This movie should not be tested internally. As there are scripts in this movie that are not supported in the internal player, they will likely cause error messages to display when testing internally. Please test from the 'UPLOAD' folder, or in the external Player or Browser (File > Test > Test in Player or Browser).

Save and Test your changes.

8.1.13 Exporting your Site template

You will need to export, and test your site as a whole as you customize each of the files. Most of the movies have a stop action in them (to hide them until called for) so they can't be tested on their own - you must export and test either index.html, mainmenu.html or main.swf.

What you need to do

1. From the File Menu, select 'File | Export | SWF' (you do not need to export the HTML as this has already been created for you in the current HTML page).

Tip: From the Main Menu select 'Tools | Preference' and in the 'Load Movie from Folder' select SWF export folder if it not already selected.

2. From the 'Export to SWF' dialog box, locate the folder you wish to save your files to (e.g. **UPLOAD**), and press Save. When you export a warning dialog will caution you that the .swf already exists and ask you if you want to replace it. Press YES to overwrite the original file.

3. Test your movie from the **UPLOAD** folder, by opening **index.html**

4. If necessary return to edit the site and continue customizing.

5. If everything is finally done your movie, upload the contents of the UPLOAD folder, including all .swf and .html files to your server.

Your Site template is now ready to play on the web!

8.1.14 Changing index.html and mainmenu.html

What you need to do

1. Select the Index.html in the UPLOAD folder, right click and select 'Open With | Notepad' from the context menu.
2. Your title is the third line on the page. **<title>:::::::::: TYPE YOUR TITLE HERE ::::::::::</title>**
3. Replace the text between the <title></title> tags with your own and save.
4. Repeat for **mainmenu.html**.

Note: If you make a mistake with, or accidentally overwrite these files, there are copies included in both the site folder and the UPLOAD folder.

Save and Test your changes.

8.1.15 Important Notes

- Changing the stacking order of objects may cause the interface to stop functioning.

- You will need to upload all files to the same directory on your server. Make sure to upload all files (**main.swf**, **section1.swf**, **section2.swf**, **section3.swf**, **section4.swf**, **section5.swf**, **section6.swf**, **index.html**, **mainmenu.html**, and **logo.png/jpg/gif** to the same directory. **Note** that the SWFs are the files that play over the web; SWIs are the files that you edit in SWiSH, they will not play over the web!

- The first page a visitor sees when he/she goes to your site should be named **index.html** if it is not in the main directory, it will not be found.

- We have not discussed changing the background color as this is as easy as selecting another color from the appropriate panel. In most of the movies, the background color isn't seen, so you will need to edit elements such as buttons, bands, etc. To change the background color, select the Movie Panel and the color chip for the Background Color.

8.1.16 FAQ's

- **Error messages when testing/playing the site/main.swi**

  You will get error messages if you do not export one of the movies to the UPLOAD folder because the main movie calls for files that it cannot find.
  After editing each page/section export it to the 'UPLOAD' folder (File > Export > SWF) overwriting the file that you received to avoid these messages.
  Note that even though you receive these error messages in the internal player, this does not mean that there is a problem with the site.

- **Can't view pages/text**
  You need to export one of your movies (File > Export > SWF) to the UPLOAD folder so that the main movie can find section1.swf-section6swf to load them.

- **Can't Open Main.swi**

  The file was likely made on a later build than you have installed. You need to update to the latest build of SWiSH Max2.

- **Image(s) not fitting properly**

  If you have imported the correct size image, then it is possible that the 'Fill Transform' has changed during editing. To reset the transform, select the image and from the 'Properties Panel' click the 'Reset Transform' button.

- **Show movie full screen, Increasing the size of the movie**

You can select 'Size = 100%' from the Export Panel under 'Export options for: HTML'(if it is not enabled use Panels > Export to enable), note that this is not recommended as it will enlarge your images and text with some screen resolutions, which may cause them to display poorly.

INDEX

SWiSH
Max