macromedia® white paper

Best Practices for Accessible Flash Design

by Bob Regan

August 2005

Contents

Introduction

This document has been written for web designers who wish to make accessible, interactive rich media. This guide to creating accessible Macromedia Flash content assumes no previous knowledge of accessibility or assistive technology. Included within are the following key topics for developing accessible Flash content:

- Define User Requirements
- Technical Requirements
- Key Concepts
- Flash Accessibility Best Practices

The first section, Define User Requirements, presents a number of use case scenarios in which fictional characters provide a framework for understanding users with disabilities. These personas illustrate the broad range of disabilities that a designer needs to be aware of and highlights the unique issues, challenges, techniques and tools of each.

The second part, Technical Requirements, establishes the base set of requirements for developing accessible Flash. As well, hints, tips and special considerations of a technical nature are shared here.

The third topic, Key Concepts, introduces ideas common to computer programming that remain relevant to developing accessible Flash content. These concepts include label, role, state and structure and provide a means to explore how a user manipulates objects within a Flash movie.

Finally, Flash Accessibility Best Practices offers a number of recommendations for creating accessible design in Flash. These are:

- Provide Text Equivalents
- Provide Context
- Control Reading Order
- Control Animation
- Ensure Keyboard Access
- Progressive Disclosure
- Enable Component Accessibility
- Provide Captions
- Provide Control over Audio Playback
- Use Color Wisely
- Support Users with Low Vision

Define User Requirements

For many Flash designers, the single greatest challenge to understanding accessibility is how to best appreciate the experience of people with disabilities. A web designer's inherent talent is an ability to perceive the world in a unique visual way. The skill of the web designer allows her to view, conceptualize and translate visual information into layout and graphics. This is a fundamental and powerful way of seeing and understanding the world that should not be taken for granted.

To understand accessibility and implement it in practice is to ask designers to set their visual skills aside. The first thing to do when addressing accessibility is to step outside of our frame of reference and consider the perspective of users with disabilities. Key questions, such as the following, arise: What are the specific issues that prevent users from accessing content? What are the tools used by people with disabilities to navigate the Internet? What techniques and interfaces are used to make working with the web easier?

The following list of fictional characters illustrates common issues for people with disabilities and may be helpful when planning out your design project. The list is certainly not exhaustive but should serve as a model for the wide range of disabilities that must be considered. What is important here is that a consistent set of use case scenarios becomes an integral part of the plan, build and test phases of your accessible Flash web design project.

John, 53, a user who is blind:

- Uses the Window Eyes 5.0 screen reader
- Uses the keyboard exclusively for input and navigation
- Does not use the mouse
- Relies on audio cues to let him know if something has happened on screen

John is a university professor who uses the computer for all of the same tasks as his colleagues, who are not visually impaired. He writes email and research papers, and reviews student work.

John is part of a dance troupe in town and uses the web extensively to read up on a wide range of topics. He is easy going, most of the time, but what really he really finds frustrating is when web sites of interest do not make their content accessible.

Ava, 30, a user who is visually impaired (e.g. person with 20/300 vision):

- Relies heavily on the keyboard for input
- Uses a mouse, but only when she can't use the keyboard
- Uses ZoomText, a screen magnifier to receive information about the movie
- Will also use JAWS with ZoomText when there is a lot of text

Ava is a developer for a small consulting firm. She can see, but only with significant magnification or when objects are held very close. She uses the computer every day for work to review and write code for her clients. She uses ZoomText most of the time but will occasionally use a screen reader to read long text passages.

Ava loves music and is a fan of Stevie Wonder. Ironically, Stevie Wonder's site, a Flash site, is completely inaccessible. She has been waiting for the new album and checks the site from time to time for information but finds it a struggle to get around.

Jeff, 22, a user who is color blind:

- Needs visuals that are usable given specific color limitations

Like 10% of all men, Jeff is color blind and cannot distinguish between certain colors. Generally speaking, he does not have much trouble using the web. However, trouble arises on sites where red buttons are placed on a green background, or where red and green buttons are next to one another to "start" and "stop" movie actions. This lack of contrast in foreground and background colors and red and green combinations can be frustrating for users who are colorblind.

Makoto, 48, a user with a mobility impairment:

- Uses only the keyboard to navigate the site

Makoto was involved in an auto accident and lost the use of his hands. He relies on a mouth stick (a long stick in his mouth with a small rubber pointer tip on the end) to navigate web pages and applications via the keyboard. He uses the tab and Enter keys to move between objects on the screen, as well as the Windows sticky keys tool to create key combinations.

Makoto works at a local rehab center with other adults with disabilities. Computers are used to complete paperwork for clients and file weekly reports. A devoted New York Yankees fan, Makoto takes a few minutes each day to check the online Baseball scores. His favorite baseball page is The Sporting News. However, accessing this page can be a tedious experience due to the many navigation links he has to tab through to get to the day's top stories.

Karen, 29, a user who is deaf:

- Receives information from the movie in a visual form.

Deaf since birth, Karen is an art teacher at the local high school. With two young kids at home and needing to maintain her teaching license, Karen decided to take the required courses online. Accessing the course was no trouble for Karen; however, a problem arose concerning some instructional video content. While most of the videos used in the coursework contained closed captioning, a few did not. In the cases where no captioning was available, Karen had to request a full transcript from student services, which often took more than a week to be delivered. As a result, Karen fell behind in her course work.

Technical Requirements

Web accessibility can be broadly described as the capacity of any user, regardless of disability, to access the same content and information. With regard to accessible Flash web content, obstacles for users with disabilities have two sources: issues of design; or issues of assistive technologies. In a following section, the techniques for successful, accessible Flash design will be discussed. In this section, we'll investigate the technical requirements and review assistive technologies.

Developers creating accessible Flash must meet the following minimum requirements:

- Macromedia Flash Player 6 or later
- Windows 98, 2000 or XP
- Microsoft Internet Explorer 5 or later
- Screen readers:
  - GW Micro Window Eyes 4.2 or later
  - Freedom Scientific JAWS 4.5, 6.1 or later
  - IBM Home Page Reader 3.04
  - Dolphin HAL 6.50
  - KDS PC Talker (Japan)

The release of Macromedia Flash MX and Flash Player 6 marked the first accessible versions of the Flash platform. This version of the player serves as a minimum requirement for accessible Flash content.

All accessible Flash content must be tested on the Microsoft Windows platform. While there have been recent improvements to the Apple Macintosh OS 10.4 release (Tiger), including a built in screen reader called VoiceOver, the Flash Player does not support this screen reader.

All accessible Flash content must be tested using Microsoft Internet Explorer. At the time of publishing this document, Internet Explorer was still the only accessible browser available. The Mozilla Project has made some improvements with the Firefox Browser, and support for screen readers will soon be available. However, the version of the Flash Player that runs in Firefox is not yet accessible.

Designers often question the wisdom of testing accessible content using only one browser and one platform. While the future of cross platform and cross browser accessibility looks promising (with the recent improvements at Apple and Mozilla), the reality is that today, people with disabilities are almost exclusively restricted to using Windows with Internet Explorer. For now, designers should feel comfortable testing their content using only this configuration.

As part of your development, production and testing phases for accessible Flash content, designers should rely on a minimum of one screen reader. While Flash content does not behave exactly the same in all screen readers, it is generally true that an application that functions properly in one screen reader will likely work in another. Applications with greater complexity or a wide range of users should plan to test using more than one screen reader.

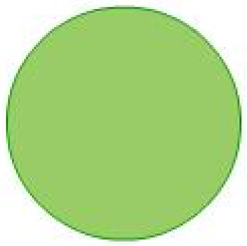
Here's how rich media content is passed from the web page to the screen reader. Flash uses Microsoft Active Accessibility (MSAA) to deliver information about Flash movies to screen readers and other assistive technologies. MSAA operates as the go between for the Flash player and the screen reader. The Macromedia Flash Player creates a list of objects on the screen and records them on the MSAA "data tree". The screen reader will then read this list as it encounters Flash content. As changes are made to the screen, the MSAA data tree is updated. Changes to the movie prompt the screen reader to return to the top of the movie and commence reading through the list again.

By default, text objects in a Flash movie are read by screen readers. Screen readers are also able to identify buttons and movie clips with attached scripts. Screen readers, however, cannot look at a graphic element on the screen and determine its meaning. It is up to the designer to assign a text description of any graphic or animated elements in a Flash movie. This information can be assigned via either the accessibility panel or ActionScript. Some properties, such as "Make Child Objects Accessible" or "`.forcesimple`" have no counterpart in HTML. Designers will need to rely on information in this document as well as information found on the Macromedia Accessibility Resource Center to learn more about these properties and the associated techniques.

Screen readers and MSAA shape the experience of Flash content for users with visual disabilities in ways that are often quite unfamiliar to sighted designers. Given that screen readers always start from the top of the movie and can only read one thing at a time, there are some complex forms of Flash content that simply can not be made accessible. For example, many simulations require users to attend to several objects at the same time. Decisions must be made based on multiple factors and relayed back to the simulation quickly. This type of multitasking activity may be easy to do in the real world for someone who is blind, but can pose a real challenge while using a screen reader.

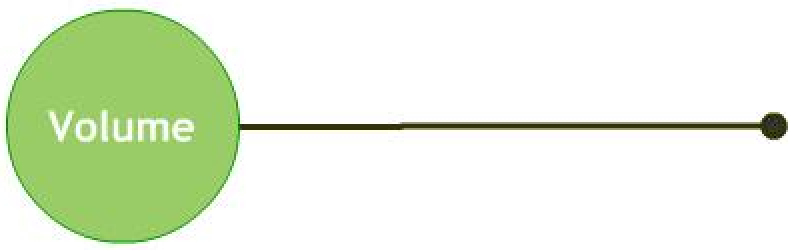
Key Concepts

In its flexibility, Flash allows designers to create simple objects that can be used as a number of different controls. Take this simple circle for example.



If we add a label and a script to this circle, it becomes a button.

Another slight change and it becomes a slider.



Yet another alteration and this same circle is now a dial.

Visually, it may be obvious what these controls are and how to operate them. However, to a screen reader user it may not be obvious. For simple buttons, no extra effort is required. Yet as controls become increasingly sophisticated, additional information about the control is required.

This is where Flash accessibility begins to approach the accessibility of desktop applications. In HTML, there are only 12 different types of controls. The screen reader knows how to handle them, and an experienced screen reader user understands how the controls work. In Windows, there are 76 different types of controls and an infinite variety of custom objects. While many of these controls are also familiar to screen reader users, there are an equal number of users who may not understand how a complex control works. Further, the combinations of controls can add greater levels of difficulty to an application.

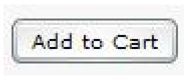
We must return to the point of expecting designers to have the perspective of a disabled user. It is important to create accessible controls that work both in a technical sense (are accessible to a screen reader) and in a practical sense (controls that are familiar and work as expected). In other words, the user has an opportunity to understand how a control functions without significant effort or help from someone else.

Macromedia Flash allows authors to create these types of controls in ways that other applications simply can't. While technologies such as SVG and Ajax have generated interest in the last couple of years, there is no way to expose these types of controls to a screen reader using these technologies. When accessibility is a priority, these technologies should be avoided.

The following set of key concepts provides designers with the fundamentals to understanding what makes an accessible control in a Flash movie.

Labels

The label for a control answers the question, "What is this thing?" Every control in a movie should have a label that describes the function of the control. Labels should be concise and read immediately before, or together with the control. In this simple example, the label for the button is clearly displayed as text inside of the button. It is important that the screen reader also reads the label.



When the label for a control is an icon, a text equivalent for the label should be provided as a text equivalent for the button. For example, the button below shows a commonly used icon for "play". It is up to the designer to make sure this button has a label associated as a text equivalent.

If the function of the control changes, so should the label. The example below shows a pause button from a movie controller. Typically play and pause are included in the same button. If the user presses play, the button transforms to a pause button. As the user activates this button, the label should update as well.

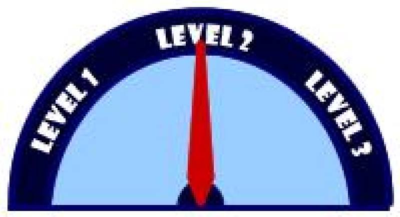
Role

The role of a control answers the question, “What does this thing do?” As shown previously, a simple control made of a circle can be a button, a slider or a dial, or any number of possibilities. Flash automatically assumes that if a script is attached to an object, then that control is a button. The designer can formally specify a role in MSAA using the Flash component architecture. However, this is a significant amount of effort and well beyond the scope of all but the most complex development projects. Instead, the role can be exposed using simple text objects that provide hints to the user.

The role helps the user to understand how a control operates. If a control is defined as a combo box, then the user will naturally assume it will behave similarly to other combo boxes they have come across in the past. The role provides a quick and easy method of explaining to a user how a control is operated.

State

The state of a control answers the question, “Is this thing on or off?” or “At level 1, 2 or 3?” and so on. Many controls have multiple settings. Think of a dial that sets a skill level in a game using the control shown below.



In this case, there are three possible settings: level 1, level 2 and level 3. The position of the red arrow lets us know visually that level 2 is selected. However, screen readers have no way of knowing anything about the red arrow or how many possible levels there are.

In controls with multiple settings, designers need to let the user know how many possible states there are for a control and what state is selected. In the example above, we might use a text object or even the name property to read, “Level 2 of 3 is selected.”

Structure

The structure of a control answers the question, “How does this thing relate to the other things on screen?” It is important that individual elements within a control not be read separate from the rest of the control. In Flash, this is generally a matter of controlling the reading order. In the example above, it is crucial that the designer ensure that the text, “level 3” is read right after the text “level 2.” In some cases, Flash reading order can get jumbled based on the position on screen. Mixing up elements of content and controls can result in a terribly confusing experience for the end user.

Accessibility Best Practices

When developing accessible Flash content, there are a few basic rules that must be observed. The following comprises a list of best practices for accessible rich media design. This list is not intended to be fixed, nor is it comprehensive. It is up to designers to make decisions about individual applications and whether they meet the requirements as outlined in the use case scenarios.

Provide Text Equivalents

Screen readers are not able to discern the meaning of graphic or animated elements on the stage. As a result, designers must provide a brief text description of graphic elements. Text equivalents can be provided for either an entire movie, a single object within a movie or a group of objects within a movie.

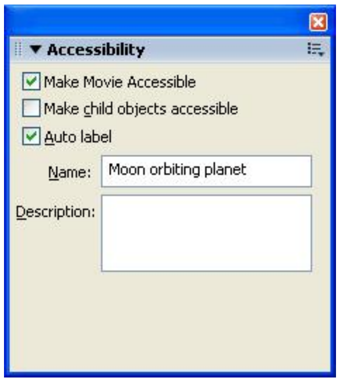
Providing text equivalents for an entire movie

Text equivalents should be provided for an entire movie in cases where the movie can be conveyed using a single text equivalent. Examples of this include movies that show a simple animation, banner ads or complex movies that cannot otherwise be made accessible.

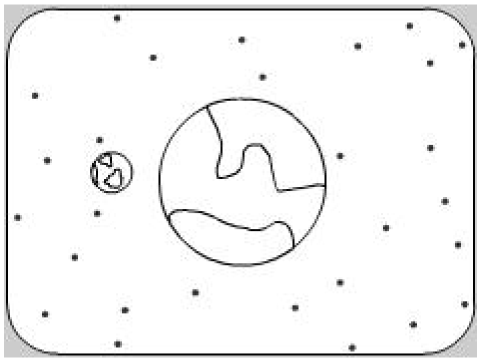
The text equivalent should be placed in the name field. It is generally advisable to make the contents of the name field short and focused in order to describe the function of the movie. The description field can be used for longer descriptions. However, be aware that both JAWS and Window Eyes read this content automatically rather than upon user request. As a result, long descriptions used in this field can result in a tedious listening experience.

In cases where a single text equivalent is used for an entire movie, the child objects of the movie should be made inaccessible. This will prevent animations within the move from causing frequent updates to the screen reader. It will also facilitate automated testing of the content for accessibility.

The text equivalent may be assigned using the accessibility panel. In this screen shot below, a text equivalent is placed in the name field, "Moon orbiting a planet."


Accessibility

[✓] Make Movie Accessible
[ ] Make child objects accessible
[✓] Auto label
Name: Moon orbiting planet
Description:



To provide a text equivalent using ActionScript, a new object must be created for each instance and then the accessibility information is assigned. Once the name value has been assigned, the accessibility objects must be updated. This is done once for all objects when a change is made. It is not necessary to update each instance of the object. Notice the sample code below includes a line to create the new object for the entire movie. Next, the value is assigned for the `.name` property and then the child objects are made inaccessible using the `.forcesimple` property. A complete list of the ActionScript properties is shown below with the corresponding fields on the accessibility panel.

```
_root._accProps = new Object();
_root._accProps.name = "Moon orbiting planet";
_root._accProps.forcesimple = true;
Accessibility.updateProperties();
```

The following is a list of accessibility properties in ActionScript.

Table 1: *Accessibility Properties in ActionScript*

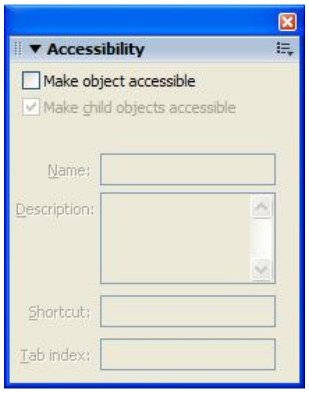
Property	Type	Equivalent in accessibility panel	Applies to
.silent	boolean	Make Movie Accessible/ Make Object Accessible *(inverse logic)*	whole movies buttons movie clips dynamic text input text
.forceSimple	boolean	Make Child Objects Accessible *(inverse logic)*	whole movies movie clips
.name	String	Name	whole movies buttons movie clips input text
.description	String	Description	whole movies buttons movie clips dynamic text input text

Providing text equivalents for single movie clip

Unlike HTML, not every movie clip or button in a Flash movie requires a text equivalent. There are at least three cases that need to be considered here.

Case 1: Setting a movie clip to be silent

First, there are elements that are purely decorative, are repetitive or convey no content. These movie clips should be made inaccessible. This can be done using the accessibility panel by deselecting the option, “Make object accessible.”



This same property can be set using ActionScript. The following code example shows an object set to be inaccessible using the `.silent` property.

```
logo_mc._accProps = new Object();
logo_mc._accProps.silent = true;
Accessibility.updateProperties();
```

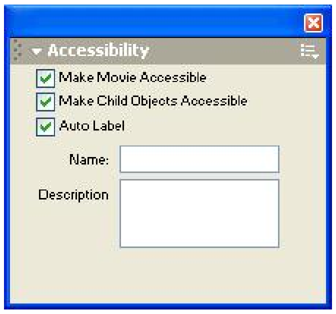
Case 2: Labels for buttons and controls

Flash includes an option called "auto-labeling." If a text object is used within a button, the Flash player will assume that text object is the label for the button. The same holds true for movie clips used as buttons. It is important in these cases that the child objects of the movie clip *are* accessible. It is important to keep in mind that only a single text object can be used as a label and the text object must fit completely within the hit area of the button.

Auto-labeling also works for components such as radio buttons and list boxes used in a movie. The Flash player will assume text objects above or to the left of the control should be used as the label. Again, only a single text object will be used as a label.

If a text equivalent is assigned via the name property using either the accessibility panel or ActionScript, that value will override the auto-labeling without disabling the auto-labeling completely. Auto labeling can only be turned off on the accessibility panel and it cannot be changed dynamically once set.

The screen shot below shows the auto-labeling option enabled.



Case 3: Text equivalents for a single movie clip or button

If a text equivalent is assigned for a single object in a movie, this can be done via the accessibility panel. The text equivalent should be relatively short and should address the function of the object, rather than a longer or more detailed description. This will help prevent the movie from becoming verbose and tedious to navigate.

The description field can be used for longer descriptions. However, JAWS and Window Eyes will both read this field by default. As a result, there is no advantage to using this field at this time.

Provide Context

Screen readers are not able to provide clues to a screen reader user about the layout or structure of a Flash movie, or individual controls within that movie. As a result, it is important that complex movies provide descriptions about the movie itself, as well as its important parts and controls. Determining when a movie is sufficiently complex to merit a description or when a control requires additional cues is really up to each designer.

Describe the movie

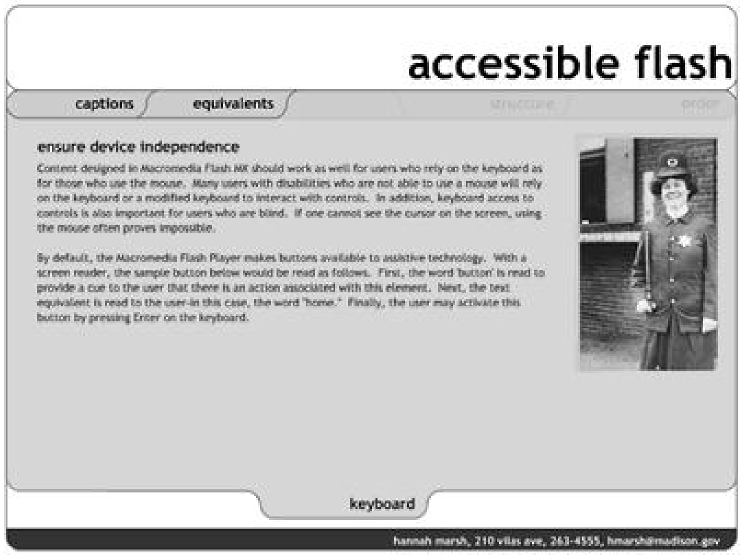
Designers should provide a description of the movie at the root level to let the user know what the movie or application is about. This description should be written to help the user get oriented to the application quickly and understand the key controls and shortcuts used.

This can be accomplished in a number of ways. First, this information may be placed on a separate screen in the Flash movie or a separate HTML page. It is recommended that in this case, a link to the information screen be placed at the top of the page in a button titled “site info.” Using this short title will help prevent the application from becoming overly verbose. This button can be hidden if desired, but to enable sighted users with disabilities to access the same information, it is recommended that a second link to the same information be placed elsewhere on the screen (most likely at the bottom).

Expose state

Flash allows designers to create an infinite variety of controls. In cases where the user is provided visual cues about the state of a control or a screen within the movie, this information should also be made available via a dynamic text field that is updated as the control is activated.

Take the example below. In this case, the movie uses tabs that drop down to indicate which screen is the active screen. While this is a helpful visual cue, this information is not available to screen reader users.

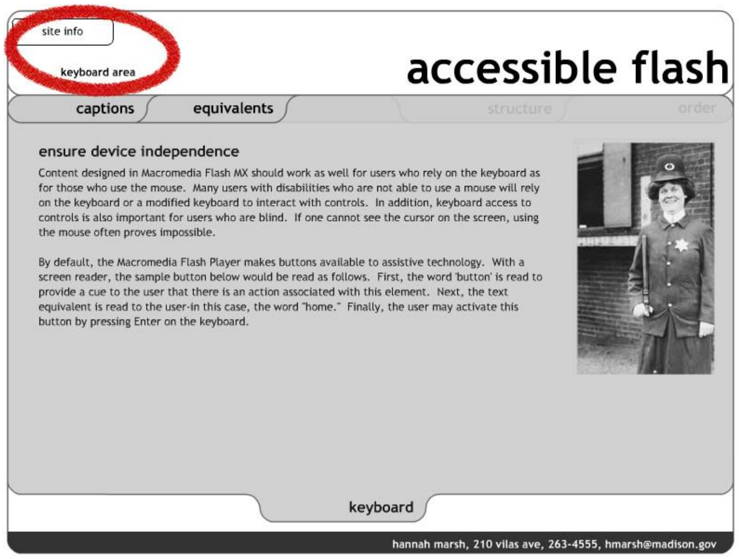


ensure device independence

Content designed in Macromedia Flash MX should work as well for users who rely on the keyboard as for those who use the mouse. Many users with disabilities who are not able to use a mouse will rely on the keyboard or a modified keyboard to interact with controls. In addition, keyboard access to controls is also important for users who are blind. If one cannot see the cursor on the screen, using the mouse often proves impossible.

By default, the Macromedia Flash Player makes buttons available to assistive technology. With a screen reader, the sample button below would be read as follows. First, the word 'button' is read to provide a cue to the user that there is an action associated with this element. Next, the text equivalent is read to the user-in this case, the word "home." Finally, the user may activate this button by pressing Enter on the keyboard.

To provide a cue for the screen reader user, a text field is hidden under the banner (see circled area below, where the hidden text is made available). Since it is intended to provide information for screen reader users, it need not be visible. As the user moves between screens, the contents of this field can be made to reflect the active screen.



ensure device independence

Content designed in Macromedia Flash MX should work as well for users who rely on the keyboard as for those who use the mouse. Many users with disabilities who are not able to use a mouse will rely on the keyboard or a modified keyboard to interact with controls. In addition, keyboard access to controls is also important for users who are blind. If one cannot see the cursor on the screen, using the mouse often proves impossible.

By default, the Macromedia Flash Player makes buttons available to assistive technology. With a screen reader, the sample button below would be read as follows. First, the word 'button' is read to provide a cue to the user that there is an action associated with this element. Next, the text equivalent is read to the user-in this case, the word "home." Finally, the user may activate this button by pressing Enter on the keyboard.

In this sample screen shot, a screen reader accessing this as a web page would read:

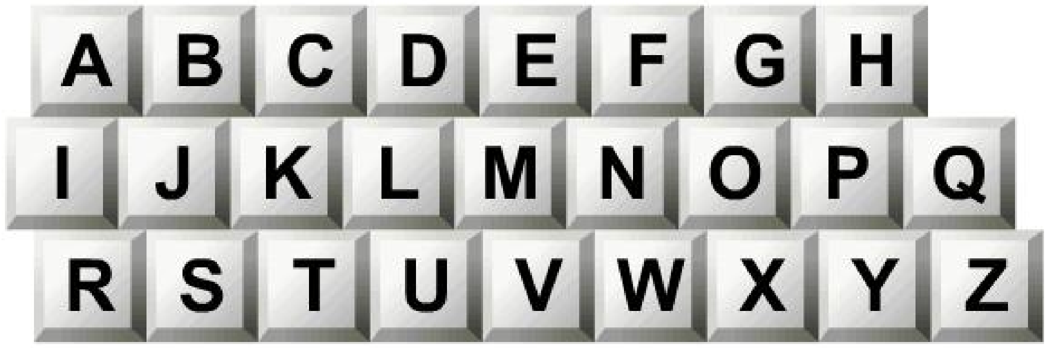
- Accessible Flash
- Site info button
- Keyboard area

- Captions button
- Equivalents button

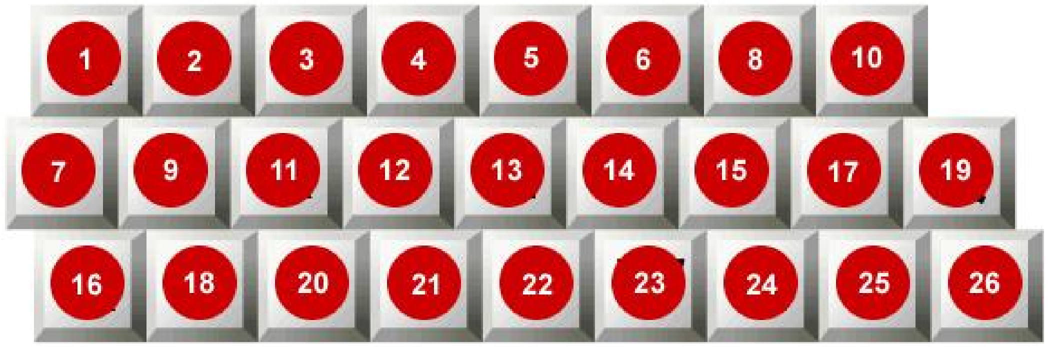
With each area of the site, the text “keyboard area” will update to reflect the active area.

Control Reading Order

The default reading order of a Flash movie does not follow a predictable left to right, top to bottom order. As a result, contents of a Flash movie can be difficult to understand. Take the example below. Based on the visual presentation of the alphabet in three rows, it would be natural to expect the reading order to follow alphabetical order.



However, the actual reader order jumps between letters in each row. The resulting order is shown below.



There are two strategies for controlling reading order. The simplest is to keep the physical size of the movie small. The second strategy requires controlling the reading order using the accessibility panel or ActionScript.

With the release of Macromedia Flash Player 8, handling issues with the reading order have been significantly simplified. In previous releases, a designer had to specify values for all objects in a movie. Missing one would cause the reading order to revert to the default. This made setting the reading order a tedious and time intensive task. In Flash Player 8, the reading order can be partially specified. The author need list only the objects that they want to control. The remaining objects will follow the default reading order after the specified values.

It is still important to consider the reading order of a movie from the beginning of the development process. It is certainly more work to retrofit for proper reading order of a Flash movie after the movie has been completely built. Therefore using a screen reader to evaluate during an ongoing development process can definitely make finding and fixing problems much easier.

The key to creating an appropriate reading order is to ensure that the order reflects the structure of the screen. At a high level, the elements of the screen should be reading in a reasonable and consistent order from one screen to the next. For example, one would most often expect that the title of the site would be followed by the navigation, followed by the content. Placing the title of the application at the end of the reading order might create confusion.

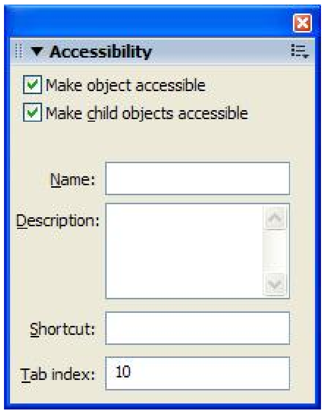
Finally, the reading order should ensure that elements from each of these higher level groups are not mixed together. For example, the title of the application should not be read between buttons on the navigation bar or pieces of the content on screen. To extend this concept, buttons at one level of a navigation bar should not be read mixed in with buttons on the next level.

Limit the size of the stage

A small Flash movie that is less than 300 pixels wide and consists of a single column or a single row of objects does not require any specific control over the reading order. Examples might include small animations or applications that pop up in a separate window, a navigation bar that consists of a single row or an application that consists of a single column.

Controlling reading order using the accessibility panel or ActionScript

The most precise means for controlling reading order is to use tab index values. Tab index values are easily assigned using the accessibility panel or ActionScript. The designer simply assigns each object a value that reflects the desired reading order.



Make object accessible: ✓
Make child objects accessible: ✓
Tab index: 10

Objects within a Flash movie that are assigned a tab index value are then placed at end of the reading order, following the default order.

Tab index values need not be sequential. In fact, it is recommended that the order NOT be sequential, to allow designers to add objects into the movie at a later time. Intervals of 10 should be sufficient.

Assigning a reading order using ActionScript

In cases where the reading order is dynamic or objects are created on the fly, designers will need to use ActionScript to assign tab index values. This is an existing property of all objects in ActionScript. There is no need to create an accessibility object to assign a tab index value.

Consider the following simple example. Three circles displayed with the text, “Third”, “Second” and “First” displayed on screen. Using tab index values assigned in ActionScript, we will reverse that reading order.

Third
Second
First

When the following ActionScript is added to movie, the reading order reverses to read “First”, “Second”, “Third.”

```
first_mc.first_txt.tabIndex = 10;
second_mc.second_txt.tabIndex = 20;
third_mc.third.txt.tabIndex = 30;
```

In general, it is best practice to place the ActionScript controlling the reading order at the root level of the movie in the first frame.

Scoping

When controlling the reading order using ActionScript, the `.tabIndex` property value has to be assigned to the text object itself, not to the parent. Consider again the example shown above. In this case, the `.tabIndex` property is assigned to the text objects in the circle movie clips. The values of the parent are not inherited by the child. Thus, assigning the `.tabIndex` property values to the circles would not result in the reading order reversing.

Static text cannot be controlled

It is not possible to provide an instance name to static text objects. As a result, it is not possible to assign a `.tabIndex` property to a static text object and the reading order of static text objects cannot be controlled.

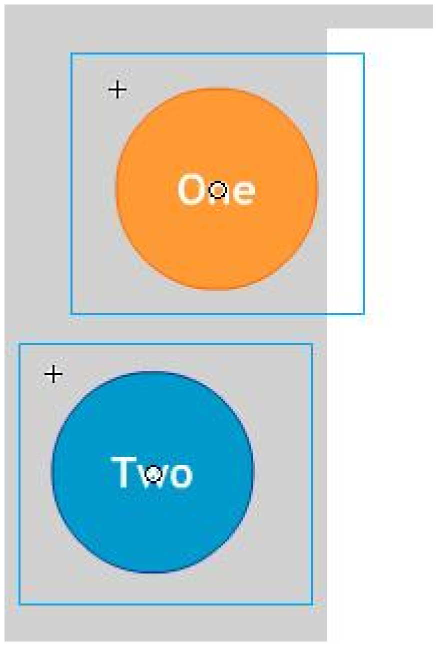
In cases that do require precise control over the reading order, designers are encouraged to use dynamic text objects. This will have implications for the font used in the application and potentially impact the overall file size. To learn more about handling font symbols in Flash, please visit:

www.macromedia.com/support/flash/ts/documents/flashfonts.htm

Removing objects from the reading order

There are numerous examples where a designer will want to hide elements from the user and remove them from the reading order temporarily. This is a relatively simple task. However, it should be noted that objects placed behind another object or tucked just off stage are not necessarily removed from the reading order. Screen readers have no concept of whether one object is in front of another or whether it is just barely on stage or completely off stage. Designers need to more formally hide objects if they wish to remove them from the reading order.

The easiest means of removing an object from the reading order is to move them completely off stage. The bounding rectangle of the object needs to be more than 1 pixel off stage. Even if the visible object itself is off stage, the bounding rectangle is what matters here. Consider the follow example. Circle “One” is offstage. However, we can see that the blue bounding rectangle is still on stage. As a result, this object will be read by the screen reader. Circle “Two” is shown with the bounding rectangle completely off stage. In this case this object will not be read by the screen reader.



A more formal way to make sure a screen reader does not read an object, is to set the `._visible` property to false or the `.silent` property to true. Referring back to this example used earlier, the circles "Third" and "Second" from the reading order can be temporarily removed.

Third
Second
First

When the `._visible` property is set to false, the circles are not read by the screen reader. However, they are not visible either.

```
second_mc._visible = false;
third_mc._visible = false;
```

Using the .silent property will ensure that the screen reader will not read these objects without affecting the visibility of these objects.

```
second_mc.silent = true;
third_mc.silent = true;
```

Controlling reading order when loading SWFs at runtime

In cases where a series of child .SWF files are loaded into a parent movie, the list of `.tabindex` values must be listed in the child movie clip. However, it is important that the values list in the reading order of each child SWF be unique. For example, if two child movies loaded into a parent movie each have three elements with `.tabindex` values of 1, 2 and 3, the screen reader will read the first value of the first movie loaded, then the first value of the second movie loaded. Next, the screen reader will read the second value of the first movie clip loaded and then the second value of the second movie clip loaded and so on. In order to read the contents of the first movie followed by the contents of the second movie, the list `of .tabindex` values for the first movie should be 1, 2, 3 while the list of values for the second move should be 4, 5, 6. These values need not be sequential, but they should be unique.

Screen reader detection

Since this case is intended to benefit screen reader users alone, this method is frequently used in conjunction with screen reader detection. Flash has a unique advantage over JavaScript in that it is able to use MSAA to detect the presence of a screen reader. The method `Accessibility.isActive()` will return a value of true if a screen reader is present and it is currently focused on the Flash content. It is important that this method not be called in the first moments in the timeline of the movie, or it could return a false negative. Rather than calling this method in the first frame of a movie, many designers will attach this method to the first button in the movie.

Control Animation

While Flash is now much more than an animation tool, it is still commonly used to create and deliver animations for a variety of purposes. There are three key issues to consider when using animations in Flash.

Hide Child Objects

As mentioned earlier, constant changes to the screen can cause a screen reader to refresh constantly. This can be very frustrating to screen reader users when they are trying to read through content and the screen reader is repeatedly returning to the top of the page.

To prevent constant screen reader refreshes, hide the child objects of the movie clips that contain animation. This can be accomplished via the accessibility panel by deselecting "Make Child Objects Accessible." Using ActionScript, the `.forcesimple` property should be set to true.

Settle Motion

Avoid constant motion on the screen when the user is expected to read text on the screen. For users with certain learning and cognitive disabilities, moving objects or patterns on the screen can be a distraction. It is acceptable to use motion together with text, but the screen must settle after a moment or two.

Avoid Blinking

It is very important that Flash movies avoid blinking for more than a second across a large viewable area of the screen. Photo-sensitive epilepsy can be triggered by blinking at specific rates if viewed at close quarters across a significant portion of one's field of vision. There is no way to predict a flicker rate precisely because it is dependent on the end user's machine. Subsequently, it is important to avoid flicker completely.

Ensure Keyboard Access

It is important that all controls that can be manipulated via the mouse are also accessible via the keyboard. This is intended to support screen reader users as well as users with mobility impairments. The Flash player facilitates keyboard access on its own by automatically making mouse defined events accessible via the keyboard. However, there are two specific techniques commonly used among Flash designers that should be avoided. In addition, designers should add keyboard shortcuts to facilitate keyboard access in complex applications. Finally, designers should be aware of an issue with the Flash Player 6 and earlier in pages that blend HTML and Flash content.

```
on (click) {
getURL(index.html);
}
```

For example, the script shown above might be used to open a web page. It is directly associated with the instance of the movie clip used as a button. This script should instead be placed in a frame, likely the first frame, of the movie. The revised script could be as follows.

```
home_mc.onRelease = function () {
 getURL(index.html);
}
```

Avoid empty hit areas

Hit areas are empty button clips with a shape defined in the hit state. These allow designers to reuse single library objects repeatedly by placing them over text objects and varying only the scripts used. Be aware, however that screen readers may find a problem with this technique. The screen reader assumes that if the button clip up state is empty, then it is not a button at all.

The solution to this issue is quite simple. By placing a transparent movie clip in the up state, screen readers will recognize the button and allow the user to activate it.

Assign keyboard shortcuts for most essential controls

In complex applications with multiple controls, it is extremely helpful for users to navigate the application using keyboard shortcuts. For many users with mobility impairments, pressing keys may be very difficult. Using keyboard shortcuts helpsreduce the number of keystrokes required to perform important tasks.

Using the shortcut field on the accessibility panel or the `.shortcut` property in ActionScript is not sufficient for this purpose. Creating a keyboard shortcut requires that a listener event be defined and a script associated with that listener. The shortcut field merely announces a shortcut value via MSAA. It does not create the listener. Moreover, no screen readers support this feature in MSAA as of the writing of this document.

Be aware of ActiveX controls trapping focus within hybrid pages

When using Macromedia Flash Player 6 or earlier versions, users may find that they are not able to tab out of a Flash movie to the HTML contents elsewhere on the page. This is an issue common to all ActiveX controls. The easiest solution to this issue is to update the Flash player to Version 7 or newer.

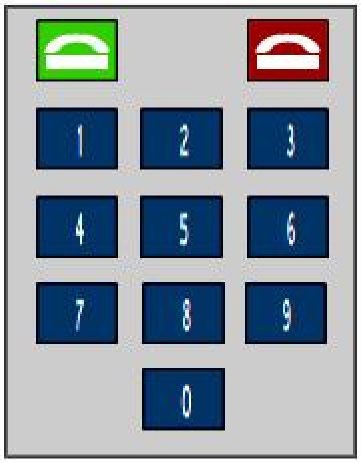
A workaround, which requires modifications to the HTML and Flash content, exists for this issue in earlier versions. Details on this technique can be found at:

www.sonokids.com/tabnew.html

Progressive Disclosure

One of the greatest challenges for screen reader and keyboard users in navigating a complex site is moving from the top of the screen to the content they are trying to access. Sites with numerous controls or links can result in a time-consuming and tedious experience for the user. As result, the best user interface for someone who relies on a screen reader or a keyboard is one that is very narrow, offering a limited number of options at the top and increasingly more as the user drills down.

A tactic increasingly used in Flash content is to hide multiple controls under a single control. As the user requests more information, additional controls are progressively revealed to the user. For example, a phone with many buttons is placed near the top of a Flash application. With multiple buttons, quite a few key presses would be required to get past it. If it reappears on other screens, then it would require additional key presses on each of these screens just to progress further.



To reduce the complexity of this control, the phone object is grouped into a single movie clip with the name property, “Pick up Phone.” By default, the .silent property for each of the buttons is set to true. The screen reader user hears only one button.

If the user presses the button to pick up the phone, four things happen. First, the individual buttons are exposed. Second, the “Pick up Phone” button is temporarily hidden. Third, all of the other content on screen outside of the phone is hidden. This allows the user to focus on just the phone for a brief period of time. Fourth, a slight change is made to the “Hang Up” button. It not only disconnects the call, but it reverts the phone to its default state, hiding the individual buttons, revealing the “Pick up Phone” button and exposing other content on screen.

The advantage of this technique is that it allows designers to create interfaces that are narrow for users with disabilities without making significant changes to the layout. In general, this technique can be applied quickly with a few extra lines of code. The only consideration that a designer should take is to use movie clips to strategically group objects and facilitate the showing and hiding of content.

Enable Component Accessibility

With the release of Macromedia Flash MX 2004, the following accessible components were included with the application.

- Simple Button
- Check Box
- Radio Button
- Label
- Text Input
- Text Area
- Combo Box
- List Box
- Window
- Alert
- Data Grid

For each component, the designer or developer need only enable the accessibility object by using the command `enableAccessibility()`. This includes the accessibility object with the component as the movie is compiled. Because there is no simple means of removing an object once it has been added to the component, these options are turned off by default. It is therefore very important that the designer or developer enable accessibility for each component. This step needs to be done only once for each component; it is not necessary to enable accessibility for each instance of a component. Here is the sample code added for the label component:

```
import mx.accessibility.LabelAccImpl;
LabelAccImpl.enableAccessibility();
```

Again, this code is added only once for each component. It is best to attach this script to the first frame in the movie. The code required for each of the components is listed below.

Simple Button
```
import mx.accessibility.ButtonAccImpl;
ButtonAccImpl.enableAccessibility();
```

Check Box
```
import mx.accessibility.CheckBoxAccImpl;
CheckBoxAccImpl.enableAccessibility();
```

Radio Button
```
import mx.accessibility.RadioButtonAccImpl;
RadioButtonAccImpl.enableAccessibility();
```

Label
```
import mx.accessibility.LabelAccImpl;
LabelAccImpl.enableAccessibility();
```

Combo Box
```
import mx.accessibility.ComboBoxAccImpl;
ComboBoxAccImpl.enableAccessibility();
```

List Box
```
import mx.accessibility.ListAccImpl;
ListAccImpl.enableAccessibility();
```

Window
```
import mx.accessibility.WindowAccImpl;
WindowAccImpl.enableAccessibility();
```

Alert
```
import mx.accessibility.AlertAccImpl;
AlertAccImpl.enableAccessibility();
```

Data Grid
```
import mx.accessibility.DataGridAccImpl;
DataGridAccImpl.enableAccessibility();
```

No additional code is required for the text input and text area components. The player implements the accessibility for these components automatically.

There is a known issue with screen readers and the combo box, list box and data grid components. The Macromedia Flash Player 7 is not able to pass information past the first instance of one of these objects without entering forms mode. This significantly reduces the usefulness of these components.

Provide Captions

Flash is frequently used to deliver audio and video content. Any audio used to deliver substantive content should include a synchronized text equivalent in the form of captions. There are three key strategies for including captions. Captions may be included by importing previously captioned audio content, by placing text objects directly on the stage, or by streaming caption data via XML.

Importing audio content that is already captioned

A simple if rather limited approach to captioning in Flash is to import content that has been captioned in another application. The limitation of this solution is that it limits the design options and flexibility of the application. By building the caption tool directly into Flash, the designer has an increased number of options in terms of formatting and the user interface.

Placing text directly on the stage

A second strategy for captioning Flash content relies on placing text objects directly on the stage. This method is the most precise in terms of synchronizing audio content with the captions, yet it is the most tedious in terms of the effort required on the part of the designer. One particular advantage of this method is that it allows for captions to be positioned on the stage to indicate different speakers as well as emphasis and emotion.

There is a third party tool from Hi-Software available to help facilitate delivering captions by placing text objects directly on the stage. The single greatest challenge in delivering captions is creating the transcript of the audio. While voice recognition software has improved tremendously, it still is not a reliable means of converting speech to text automatically. Once the transcript is available, there are two commonly used tools for creating captions: Hi-Caption SE from Hi-Software and MAGpie from the National Center for Accessible Media. These tools help designers break the transcript into individual screens of text and then set the timing for each of those screens.

Hi-Caption SE has the additional advantage of being able to convert this information directly into text objects on the timeline. This is particularly useful in situations where use of XML can result in sandbox violations, such as a Macromedia Breeze presentation.

Streaming XML caption data

The most flexible and straightforward means of delivering caption data in a Flash movie is to stream caption data at run time. This is possible with both Hi-Caption SE from Hi Software and MAGpie from the National Center for Accessible Media. Both create a custom XML file with caption data specifically for Flash. This file strips down xml data included in captioning standards such as SAMI for improved performance in the Flash player.

Hi-Caption SE includes a component that imports this XML file and delivers it via a pre-built closed captioning interface. The designer may modify the look and feel of the icon as well as the text used to deliver the captions.

For more information on Hi-Caption SE, visit:

www.macromedia.com/software/flash/extensions/hicaption/

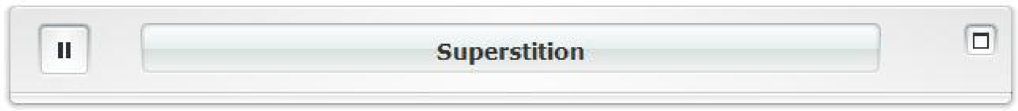
For more information about MAGpie visit:

ncam.wgbh.org/webaccess/magpie/

Provide Control over Audio Playback

Music and audio that plays as the site loads presents a serious challenge to screen reader users. The audio from a Flash movie can interfere with the end user's ability to hear the contents of a movie using a screen reader. As a result, it is important to make sure that the user is given control over when music is played.

The simplest strategy for handling audio playback is simply to allow the end user to control audio with a play and pause button. Allowing the end user to initiate audio provides the experience of the audio without creating additional hurdles.



A more advanced strategy for controlling playback relies on the use of keyboard shortcuts for audio playback. Providing global keystrokes that allow the user to control the audio can greatly enhance the experience for end users. Here are several controls to consider.

- Play / Pause
- Mute
- Volume

Play and Pause is typically controlled using a single keystroke, such as the letter P, as a toggle. The first time the button is pressed, the audio starts to play. The second time the button is pressed, the audio is paused. A mute button, such as the letter m or the number 0, silences but does not stop the audio. This provides the screen reader user the opportunity to listen to the screen reader temporarily without stopping the audio. Finally, volume controls allow the user to quietly play the audio in the background while still listening to the screen reader. This is most appropriate in cases where the audio does not require the focused attention of the user, as in the case of a music stream.

Use Color Wisely

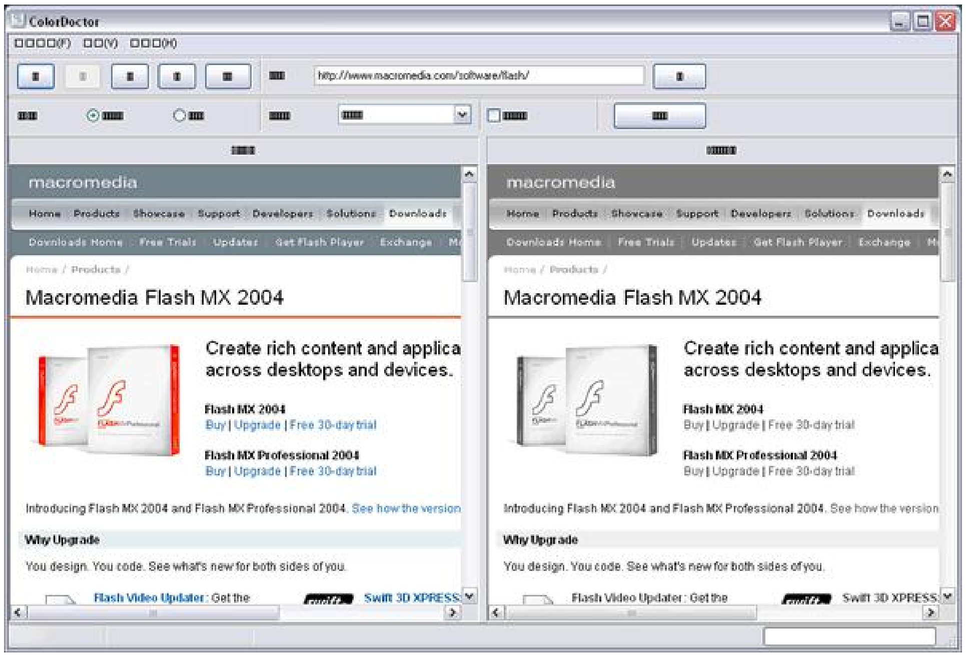
Given the range of colors available to Flash designers, it is important to emphasize that color selection in Flash content should consider issues for people with color deficits and low vision. This means that color should not be the sole means of providing information. For example, you should never say, "Click on the red button to move forward and the green button to move back." It is acceptable to reference color, but a second indicator should be used at the same time. The same example would be fine if we add a reference to position as well, "Click on the red button on the right to go forward and the green button on the left to go back."

A second issue related to the use of color is to ensure that foreground and background colors have sufficient contrast to ensure readability. One way to think of this is if the application were displayed on a black and white television, would the colors be readable. Colors that lack contrast can make it very challenging to read elements on the page.

A helpful tool in making decisions about the use of color is the Color Doctor from Fujitsu. This free tool simulates grayscale as well as three different types of color blindness. The Color Doctor can be found at:

design.fujitsu.com/jp/universal/assistance/colordoctor/

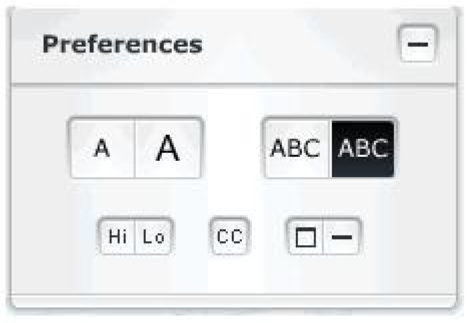
One major issue to note about the Color Doctor is that it is currently only available in Japanese. Just the same, it is helpful to English speaking designers, even if it takes a little longer to decipher the controls.


ColorDoctor
http://www.macromedia.com/software/flash/
macromedia
Home | Products | Showcase | Support | Developers | Solutions | Downloads
Downloads Home | Free Trials | Updates | Get Flash Player | Exchange
Home / Products /
Macromedia Flash MX 2004
Create rich content and applica across desktops and devices.
Flash MX 2004
Buy | Upgrade | Free 30-day trial
Flash MX Professional 2004
Buy | Upgrade | Free 30-day trial
Introducing Flash MX 2004 and Flash MX Professional 2004. See how the version
Why Upgrade
You design. You code. See what's new for both sides of you.
Flash Video Updater: Get the
Swift 3D XPRESS

Support Users with Low Vision

One of the most complicated situations in accessible Flash design is that of a person with low vision who does not rely on the use of a screen magnifier. Users with moderately to severely low vision rely on the use of a screen magnifier to view content on the screen. Magnifiers, such as Zoom Text from AI Squared, not only make the contents of the screen larger but also move the point of focus to the center of the screen to make working with content easier. Many magnifiers also include functionality that allows users to view content in a variety of contrast modes and include screen reader functionality. Screen magnifiers are based on very similar technologies to screen readers. As a result, content that reads well in a screen reader will also tend to read well with a screen magnifier.

However, many people with low vision do not rely on screen magnifiers but instead use the browser settings to change the font size. The Macromedia Flash Player 7 does not support browser settings for font size. As a result, designers need to take additional steps to consider and support users with low vision. There are two key strategies identified here. First, applications built in Flash should incorporate options that allow users to modify the text size of an application when possible. A simple example shown below allows the user to globally increase the text size from 12 pts to 18 pts by pressing a button on the home screen of the application.



Conclusion

If you are a web designer new to Flash accessibility, it is best to keep in mind that the biggest challenge to creating accessible rich media may not be the technical aspect, but rather the mental. Understanding how people with disabilities use web applications and what makes a great web experience requires designers to revisit their assumptions about user interfaces and interactivity. Designers must create and revisit their use case scenarios regularly while continuing to test and review using assistive technologies. Getting to know and understand these use cases will help designers develop the instincts they need to build applications that are not only accessible but truly usable for people with disabilities.

The techniques reflected in this document are neither technically advanced, nor are they aesthetically challenging. Building a great Flash movie depends on designers developing an instinct for how all users, including those with disabilities, use content and making sound design and development decisions along the way.

Resources

Macromedia Accessibility Resource Center at:
http://www.macromedia.com/macromedia/accessibility/

Rich Media Accessibility Center at:
http://ncam.wgbh.org/richmedia/mediatypes/FL.php

Accessibility Blog at:
http://weblogs.macromedia.com/accessibility/

Hi-Caption SE at:
http://www.macromedia.com/software/flash/extensions/hicaption/