

Animating Activators for Morrowind in 3ds Max 4

This is a tutorial that shares my findings while attempting to animate activators for Morrowind using 3ds Max 4. First of all I tried to use 3ds Max 5 to make a massive mechanical animation, in the end I found out that it will not export animation to NIF properly. So finally, after feeling the cold bitterness of being beat, I tried it over again in 3ds Max 4 at a much smaller scale and many problems occurred. Herein is the errors I encountered and how to get around them. The first section is a list of discoveries about what will and will not export to NIF. The second section is a walkthrough of how to animate a double piston assembly that will actually export to NIF.

Disclaimer: This tutorial will not be the only way to do what I am showing you to do; it is the way that I found worked best for me. This tutorial is solely for 3ds Max 4, using other versions may create a massive headache for you. Lastly, the Max file that is used for this tutorial is not anatomically correct so to speak. Anyone who is familiar with motors will notice the piston assembly will literally blow out the engine because the pin sticks past the piston body. The pin is placed the way it is because I am lazy and I want this tutorial idiot proof. And yes, I misspelled hypotenuse in most of my diagrams.

Section One: What makes an animation export properly?

You will find these suggestions in every Morrowind animation tutorial you come across. But I will mention them one more time because they are vital in making your animation work in game.

1. **The Root Bone:** The Root Bone is an invisible Point or a Dummy, named “Root Bone”, that is placed somewhere within the 3d scene, typically at 0,0,0. In the NIF format the animation looks to the Root Bone to dictate the center of the mesh and also looks to the Root Bone as a director of all animation movements. The Root Bone **must** be the parent object of all the objects in the scene. Also, the Root Bone must have a Key Frame for every Key Frame attached to every moving object in the scene. All this will be explained in the tutorial.
2. **The Key Note:** A Note Track will be added to the Root Bone. In the Note Track is where Key Notes are made. Key Notes are what Morrowind looks for to dictate what part(s) of the animation are to be played next. Two examples of these are: Idle: Loop Start...Idle: Loop Stop. They must be spelled exactly right with exact spacing between Idle: and Loop Start (One space). If they are spelled incorrectly you get Mr. Error Message.
3. **The Bounding Box:** A static mesh does not need a Bounding Box; it will export with no issues. However, an animated NIF will not be able to recognize the Root Bone unless there is a Bounding Box surrounding the entire mesh. The Bounding Box gives the mesh a collision boundary.
4. **Moving the Root Bone:** When an animated mesh is setup properly, moving the Root Bone will move the entire mesh, don't do it unless that gives the desired effect you are looking for. If you do move it you must move it back to the original location at the end of the animation.

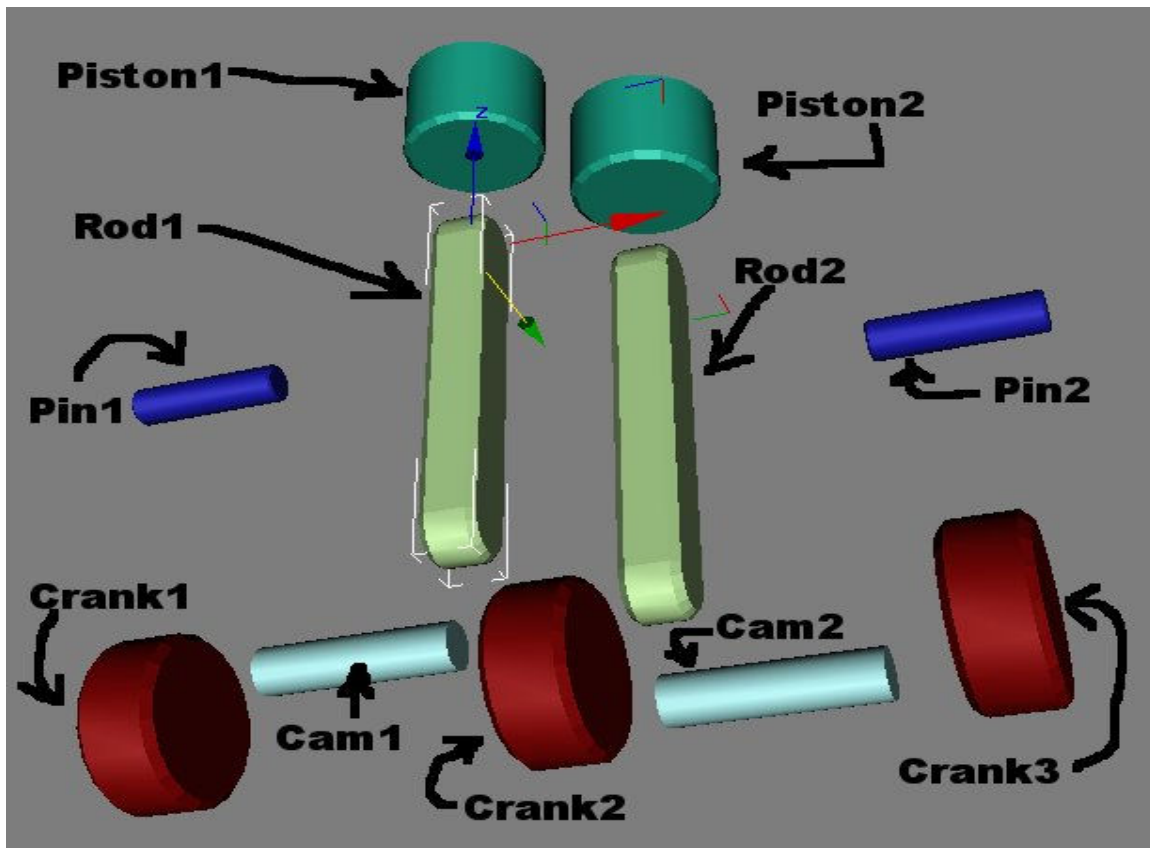
5. **Only Primitive Movements:** Morrowind cannot handle dynamic movements much like the piston example in the walkthrough portion of this Tutorial. So, you must break these movements down into little chunks so Morrowind can recognize it. 3ds Max features like Inherited Movements and IK Chains will not be noticed by Morrowind. This is a pity, because these two things can make the difference between the easy way and the hard way of animating a mesh.
6. **Using the Correct Version of 3ds Max:** Using 3ds Max 5 to animate activators will result in major issues and will crash every time you export. Only Max 3 and 4 have proper exporters for the NIF format. Exporters: <http://tinyurl.com/et38e>

Section 2: Complex Animation of Activators Walkthrough

Section 2.1: Establishing The Design And How The Animation Will Work

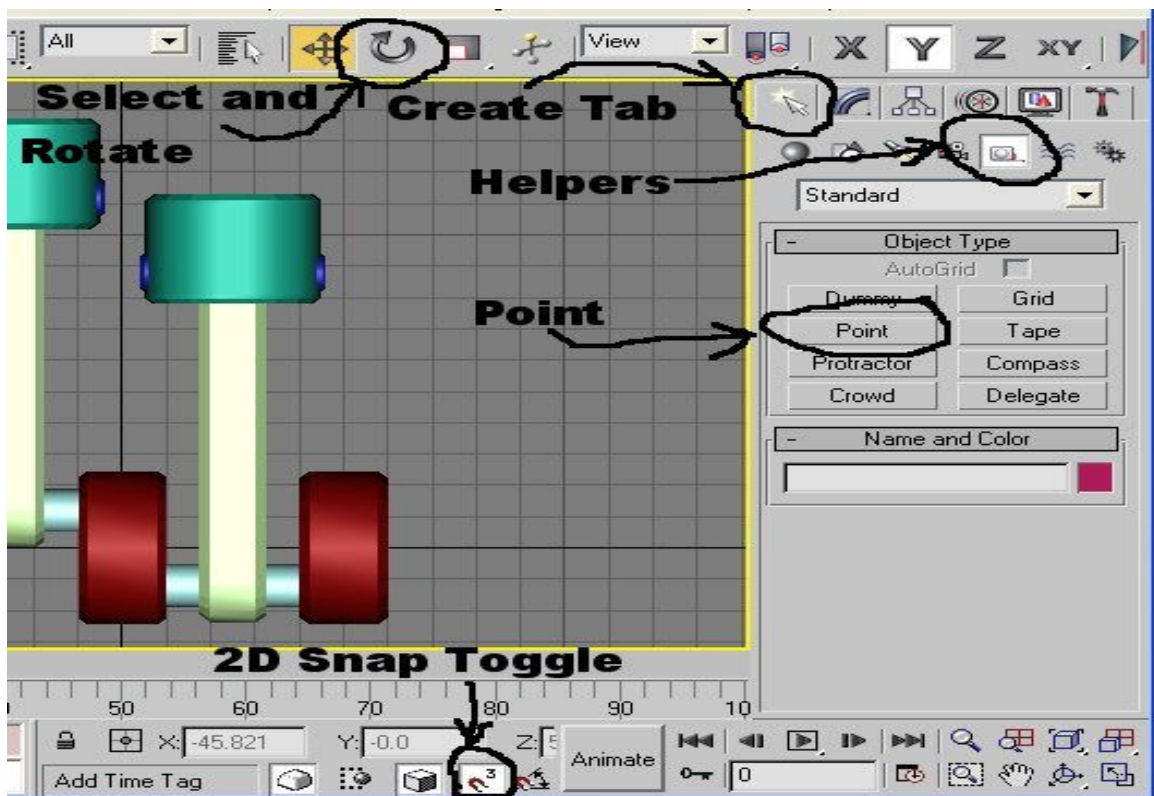
Download the PistonExample.max mesh here: <http://tinyurl.com/rhmhf>

Place the file into your 3ds Max 4 directory under scenes. Load up 3ds Max 4 and open the mesh you just downloaded. You should be looking at a not so pretty double piston assembly running off the same cam shaft. If you press “h” you will see that there are multiple objects in this mesh, each of which has an important purpose. Here is a picture of an exploded view of the separate objects:



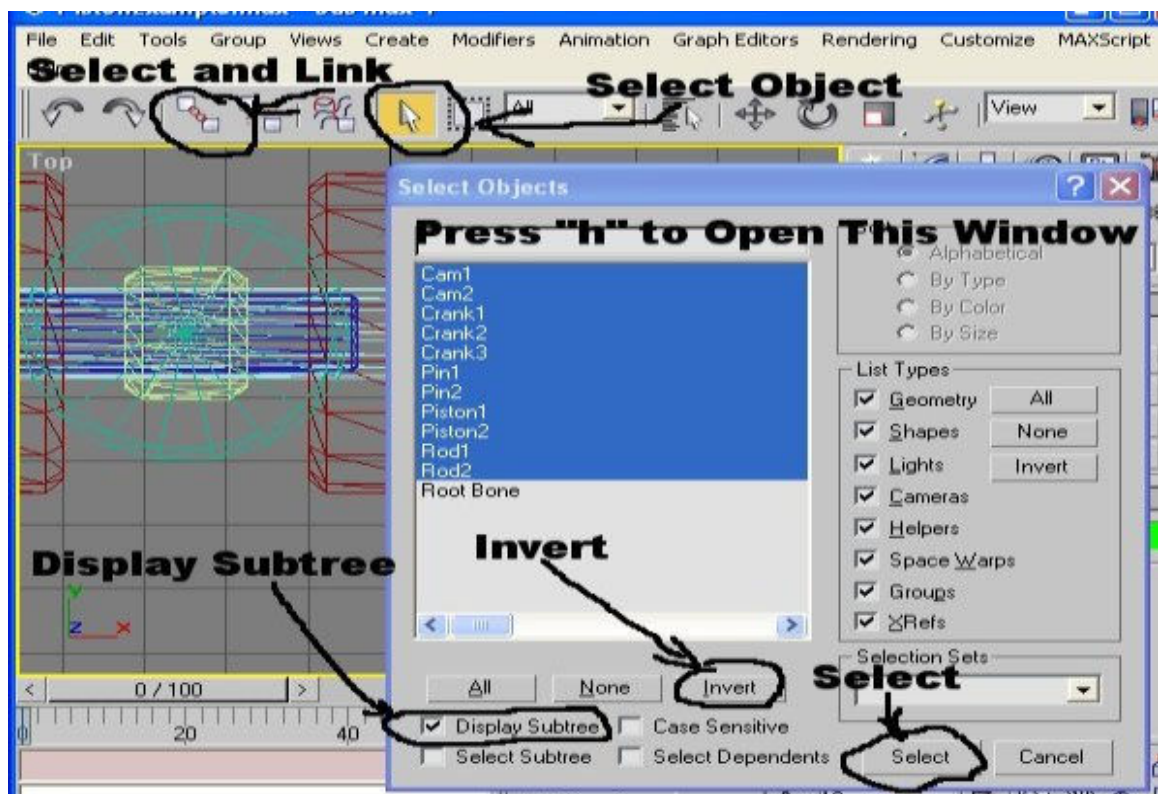
Now that we understand how this is to work, we need to plan out how it needs to be linked and aligned to interact in the correct manner. We will concentrate on pivot points to start with. We don't need to worry about the Cranks or the Cams right now, their pivots are aligned properly and they are dealt with later. However, the pivots on the Pistons and the Rods are just centered on the objects. Their pivots must be moved so they all are lined up with the pivot points of the Pins. So turn on Select and Move, yellow background means it is on (*see picture above*), then either click on Pin1 to select it, or press "h" and select it from the list then click "Select" in that window. In the bottom window of the screen you will see three axis values $x=-25.0$ $y=0.0$ $z=112.0$, these axis values need to be the pivot points for Rod1 and Piston1. Now click the Hierarchy tab, also shown in the picture above, ensure "Pivot" is selected, and finally click "Affect Pivot Only", it will turn blue when on. Remember, "Select and Move" should still be turn on. Select Rod1 and look at the axis values at the bottom middle of the screen, it should read $x=-25.0$ $y=0.0$ $z=62$. Now click on the 62 in the z axis and type in 112 and hit enter, the pivot will now be completely in line with the pivot of Pin1. Next, with "Affect Pivot Only" still turned on select Piston1. You will get an axis reading of $x=-25.0$ $y=0.0$ $z=119.5$. Once again the only thing that is different is the z axis so change it to 112 again. Repeat the process with Pin2, Rod2 and Piston2. Please note the values will be different than Pin1. Select Pin2 and you will see the values are $x=25.0$ $y=0.0$ $z=88.0$. Once you have completed this task, turn off "Affect Pivot Only", it will de-highlight and turn grey when turned off.

Section 2.3: The Root Bone, God of Morrowind Animation



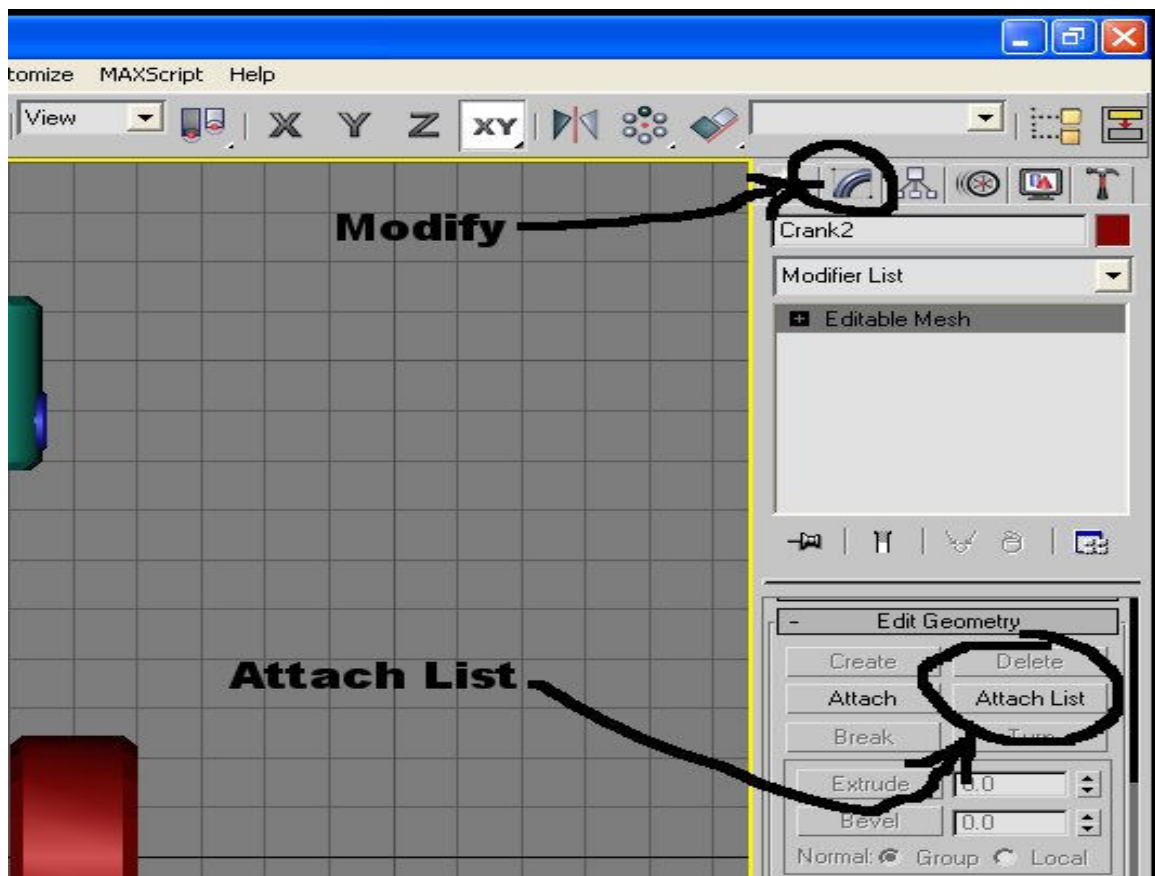
Before we link any objects to another we will make the god of your scene in the eyes of Morrowind, the Root Bone. We have been doing everything through the “Front” view port so far, we will want to place the Root Bone or “Point” through the top view port. *(For what ever reason, placing the Point in any view other than the top view will cause the pivot of the “Point” to be turned 90 degrees on one or more axis; this has bad side effects on the child objects of the scene that we will establish in the next step.)* Press “t” to change to the top view port. First click on the “Create” tab, then select “Helpers”, and then select “Point”. Now right click on the “2D Snap Toggle” and a window will popup, ensure that only “Grid Points” has a check mark beside it. Now left click on the “2D Snap Toggle” to enable it. (Its background will turn white when it is enabled.) Move your cursor close to the 0,0,0 axis of the top view port, your cursor will become a blue cross hair when you are close enough to the grid point for it to snap to 0,0,0. Left click and “Point01” will be created at 0,0,0. On the middle right side you will see a “Name and Color” tab with the name “Point01” directly under it, click on the name and change it to “Root Bone” without the “.”. **The name must be exact, with one space between both words.** Turn off the “2D Snap Toggle” by left clicking on it again. To ensure the Root Bone is not rotated in the manner described above, keep it selected and turn on “Select and Rotate”. Look at the rotate values in the bottom middle of the screen. X, y, and z must be 0.0, if they are not, rotate the object until it is.(0.0345 will not do either it must be exactly 0.0)

Section 2.4: Parent and Child Objects, A Guide to Linking Objects



Time to “Link” all objects to the Root Bone. *(I will be referring to Parent and Child objects from here in. Child objects are objects that are linked to another (Parent) object. If a Child object is moved, it will have zero effect on the linked Parent object, but if a Parent object is moved it will have an effect on its Child object. This concept will be made clear later at which point you can tinker with the mesh to see what I am talking about.)* Press “f” to use the front view port again. Turn on “Select object” (This clears out the create point device) then press “h” to enter the “Select Object” window. Click on “Root Bone” then click the “Invert” button, this will highlight everything but the Root Bone. Click the “Select” button in that window, the window will close and all the objects except the Root Bone will be selected. Now turn on the “Select and Link” button and press “h” again. Only the Root Bone will be displayed in the “Select Object” window, click on the Root Bone and press the “Link” button which is in the same location as the “Select” button in the “Select Object” window. The window will close; the Root Bone is now the Parent object of the entire mesh. Click on the “Select object” button to disable the “Link” selection. Press “h” one more time and this time click the “Display Subtree” option, the Root Bone should go to the top of the list and all the others should be indented under it signifying the Root Bone’s parenthood over them.

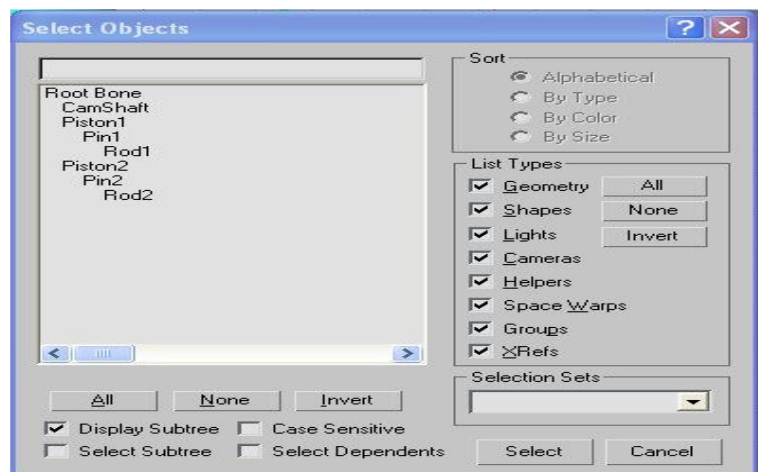
Section 2.5: Attaching Objects For Uniform Animation



Next, the Cams and the Cranks. I originally made them separate just to establish the pivot points. Turn on “Select and Move” and click on Cam1, take note that the z axis= 12 and we can tell just by looking that it is in the top or 0 degree position of the “Shafts”. Now click on Cam2, take note that the z axis= -12 and we can tell just by looking that it is in the bottom or 180 degree position of the “Shafts”. So the Cams will be rotating on a 12 pixel radius. This all the information we need from the entire Cam Shaft assembly. Now we can “Attach” the 3 Shafts and the 2 Cams into one object. Turn on “Select object” and click on “Shaft2” or select it from the “Select Object” window by pressing “h”. Now click on the “Modify” tab and scroll down in the modify spinner (Middle right side) until you get to the “Edit Geometry” area. Click on “Attach List” and select Shaft1, Shaft3, Cam1, and Cam2 in the window that pops up then click “Attach”. The window will close and the five parts of the crank shaft will become one. Just below the “Modify” tab will be the name “Shaft2”, click on the name and type in CamShaft and press enter. *Renaming objects can take the guess work out of your mesh work. Having 45 boxes all named Box01-Box45 is not very descriptive of what the boxes are doing. The more organized you are, the quicker you will become.*

Section 2.6: Developing a Dynamic Hierarchy

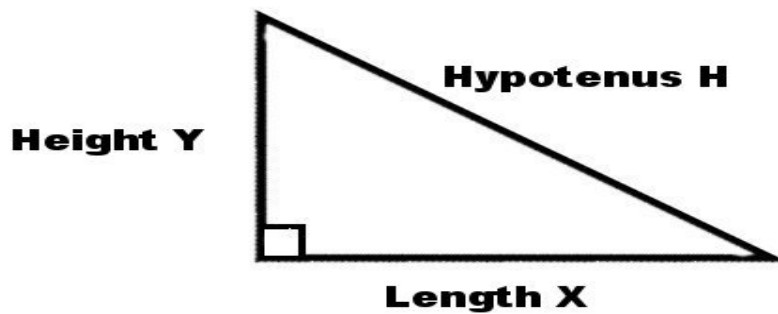
We are getting close to animating this mesh, but first we must develop the hierarchy for the piston movement. In this tutorial we will have the Piston as the parent, then the Pin as the first child, then finally the Rod as the lowest child. *To better define this one more time, any movement, rotation, and or scaling of the **Piston** will result in the same actions taking place on the **Pin** and the **Rod**. Any of these three actions made on the **Pin** will result in the same actions taking place on the **Rod**. But any of these three actions on the **Rod** will only affect the **Rod**.* So turn on the “Select and Link” button and select the lowest child, which is Rod1. Press “h”, select Pin1 and press “Attach”. The window will close then select Pin1 and press “h” again then select Piston1 and click “Attach”. Follow this procedure starting with Rod2 to Piston2 next. After that turn on the “Select object” button to disable the “Select and Link” function. Press “h”, ensure “Display Subtree” is still turned on. Your hierarchy list should look like this (If not, undo it and re-link it):



Section 2.7: Trigonometry, Your Worst Enemy, But Your Best Friend

Here is something you might not be expecting; this next step is the trigonometry section. And don't whine about your inability to do trigonometry or math in general. First off it is not true, you just think it is. And also, I will teach you what you need to know and how it works. It is very simple, and it has a very simple pattern. We will start with the Pythagorean Theorem or PT as I will be referring to it. This formula applies to any right angled triangle (A triangle that has one side that is exactly 90 degrees) and **only** right angled triangles. PT is used when you know the measurement of two sides of a right angle triangle and you want to know the measurement of the third side. The following diagram illustrates a right angle triangle and the various formulae generated from the PT:

Pythagorean Theorem: $X^2 + Y^2 = H^2$



Therefore: $X = \sqrt{H^2 - Y^2}$

Therefore: $Y = \sqrt{H^2 - X^2}$

An example of these formulae would be this:

$X=3$ $Y=4$ $H=?$

$3^2 + 4^2 = H^2$

$9 + 16 = H^2$

$25 = H^2$ (To finalize the calculation we must take the square root of H and 25)

$H = 5$

$X=?$ $Y=4$ $H=5$

$X = \text{Square Root of } (5^2 - 4^2)$

$X = \text{Square Root of } (25 - 16)$

$X = \text{Square Root of } (9)$

$X = 3$

$$X=3 \quad Y=? \quad H=5$$

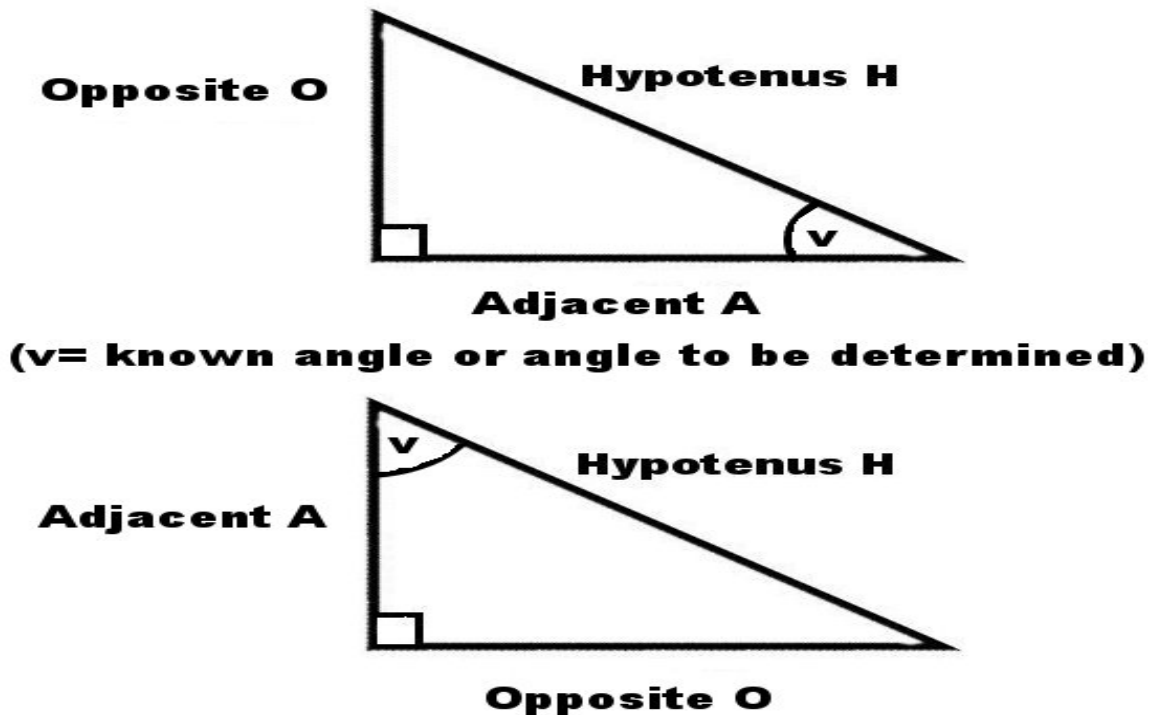
$$Y=\text{Square Root of } (5^2 - 3^2)$$

$$Y=\text{Square Root of } (25 - 9)$$

$$Y=\text{Square Root of } (16)$$

$$Y=4$$

Now we will move to the next level, trigonometry. Trig is actually easier than most people let themselves believe. I will sum up the three main trig functions that you will need to know for dynamic animation like we will be doing. Once again, trigonometry only works with right angled triangles. The benefit of Trigonometry is the ability to calculate unknown lengths of sides and or angles of the triangles you are working with. This is possible because of the unchanging properties of a right angle triangle. This diagram will illustrate what I am about to teach you:



There are three trig functions you will need to know, they will be capable of calculating needed angles and lengths of sides, given you have the minimum data to make a calculation. The minimum data is either one angle and one length, or the length of two different sides. *A, O, and H are referring to the lengths of the corresponding sides. Note, ArcSin/Cos/Tan is known as Inverse Sin/Cos/Tan on your scientific calculator, it returns an angle value.* The three functions are:

$$\text{Sin } v = \text{O (Opposite) / H (Hypotenuse)}$$

$$\text{O} = \text{H} \times \text{Sin } v$$

$$\text{H} = \text{O} / \text{Sin } v$$

$$v = \text{ArcSin (O / H)}$$

Cos v = A (Adjacent) / H (Hypotenuse)

$$\mathbf{A = H \times \text{Cos } v}$$

$$\mathbf{H = A / \text{Cos } v}$$

$$\mathbf{v = \text{ArcCos } (A / H)}$$

Tan v = O (Opposite) / A (Adjacent)

$$\mathbf{O = A \times \text{Tan } v}$$

$$\mathbf{A = O / \text{Tan } v}$$

$$\mathbf{v = \text{ArcTan } (O / A)}$$

Example: Using the diagram above, if v=35° and H=125 what is A and O?

Use Cos equation to get A.

$$\mathbf{A = H \times \text{Cos } v}$$

$$\mathbf{A = 125 \times \text{Cos } 35^\circ}$$

$$\mathbf{A = 125 \times 0.819}$$

$$\mathbf{A = 102.394}$$

Use Sin equation to get O.

$$\mathbf{O = H \times \text{Sin } v}$$

$$\mathbf{O = 125 \times \text{Sin } 35^\circ}$$

$$\mathbf{O = 125 \times 0.573}$$

$$\mathbf{O = 71.697}$$

ArcTan Example: Using diagram above, if A=100 and O=60 what is v?

Use Tan equation to get v.

$$\mathbf{v = \text{ArcTan } (O / A)}$$

$$\mathbf{v = \text{ArcTan } (60 / 100)}$$

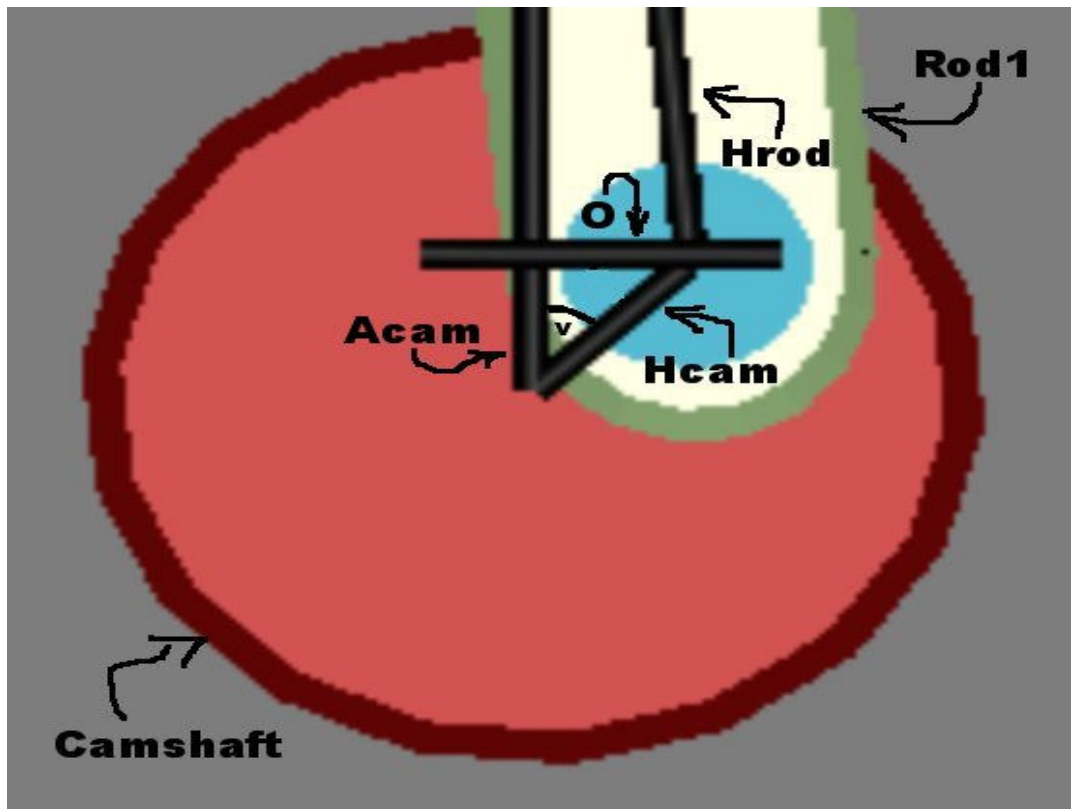
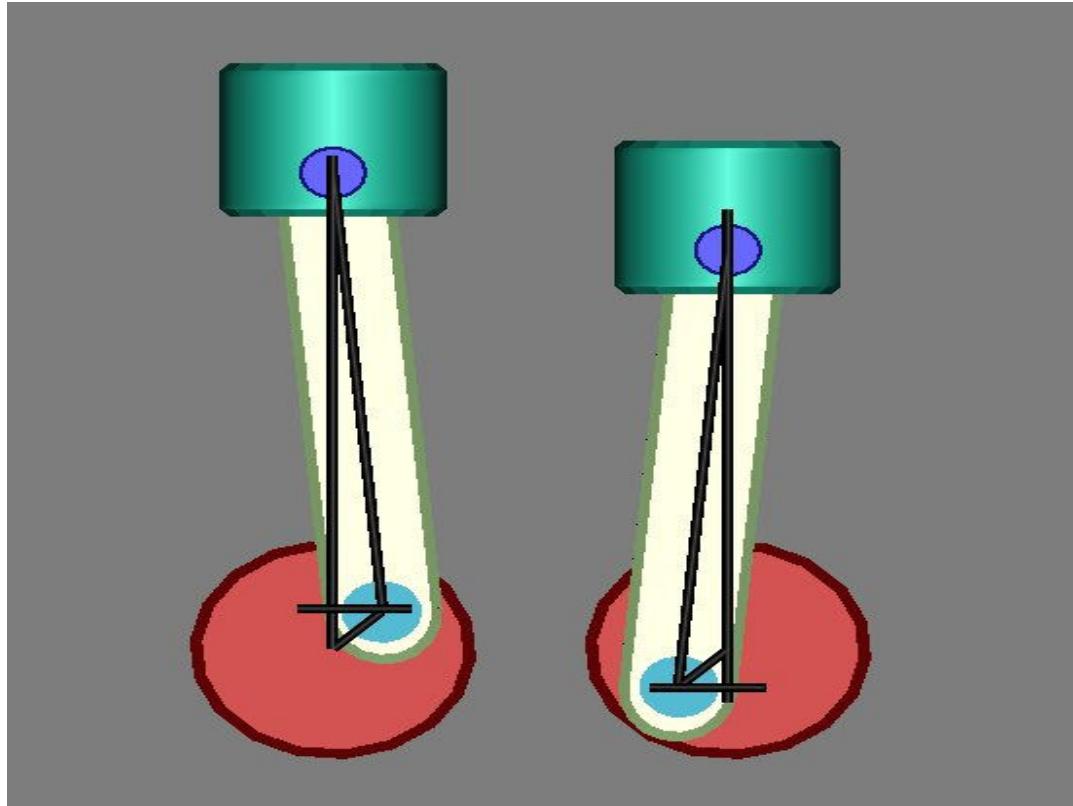
$$\mathbf{v = \text{ArcTan } (0.6)}$$

$$\mathbf{v = 30.964^\circ}$$

The final benefit of using trigonometry is the properties of the sin and cos wave. I first thought I could only calculate 0° to 90° with the sin and cos functions. I was dead wrong; both functions can calculate any positive or negative number including fractions. The **sin** function returns a **positive** result between **0° and ±180°**, but it returns a **negative** result between **±180° and ±360°**. The **cos** function returns a **negative** result between **±90° and ±270°**, but it returns a **positive** result between **±270° and ±450° (Or ±90°)**. This is excellent because it will greatly simplify the needed calculations to complete this tutorial.

Your math lesson is over. The next section involves the application of these math principles. Why are we going through this you ask? It will start to make sense soon.

Section 2.8: Determining the Variables Needed for the Animation



Now we must design a vectoring formula, ugly isn't it? *The first picture on page 11 depicts an out of scale triangular path of both pistons. (Pistons are split apart so we can see what is happening) The picture shows that the CamShaft has been spun 45° in the left view port. The rest of this step will be focusing on the second diagram. The second diagram depicts Piston1 assembly; Piston2 operates exactly the same as Piston1 but we will need to make a minor adjustment, we'll cover that later. Lets define the different variables I have labeled to the Piston1 assembly:*

Hrod = Hypotenuse of Rod1 is a constant meaning it will always = 100

Hrod = 100

Hcam = Hypotenuse of CamShaft is a constant meaning it will always = 12

Hcam = 12

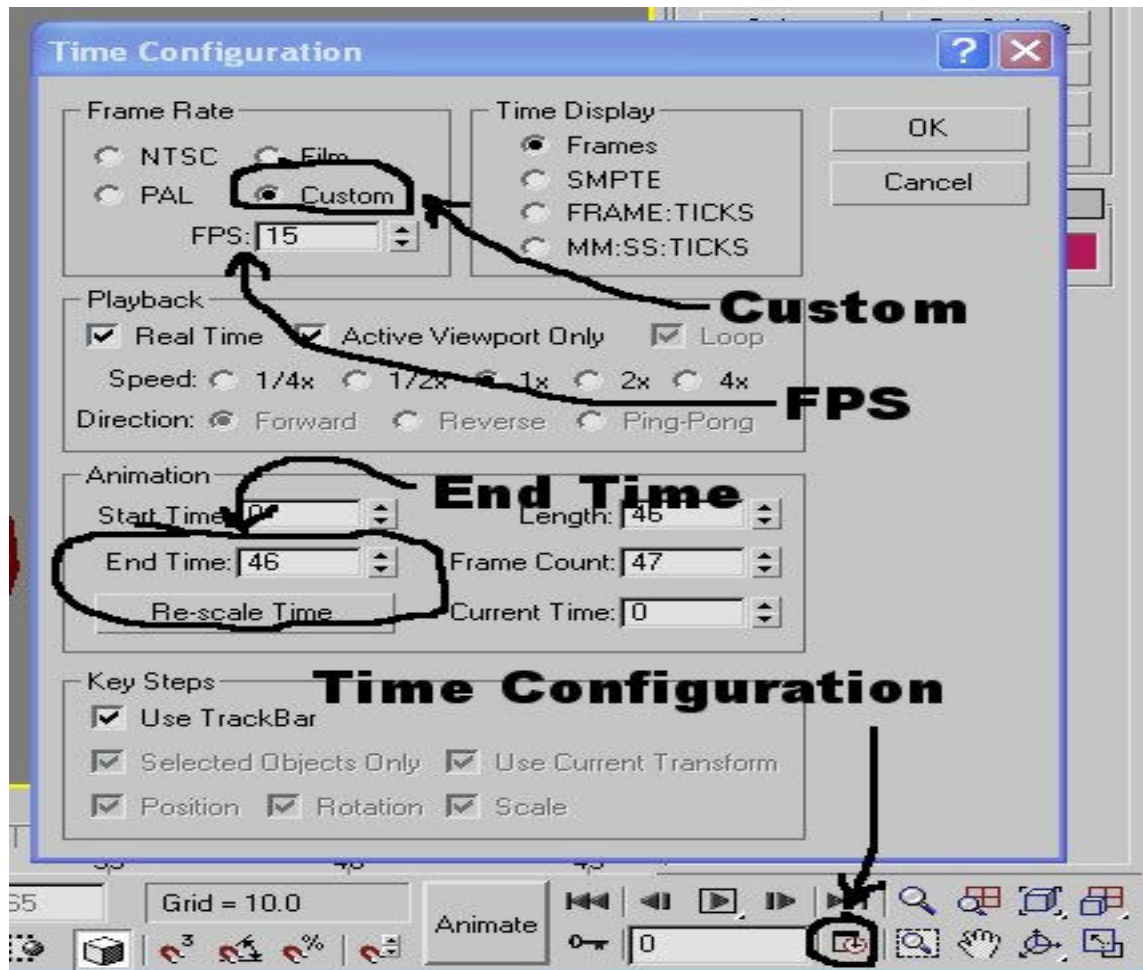
Acam = Adjacent side or vertical CamShaft movement (Needs to be calculated)

O = Opposite side of CamShaft movement (Used to calculate movement of Rod1)

v = Angle of CamShaft from 0° (We set this, and base our all calculations from it)

vRod = Angle to be calculated for Rod1 rotation (Not show in diagram)

Section 2.9: Setting the Time Configuration for a Clean Animation



Now we must decide how many seconds the animation will run for, how many rotations of the CamShaft will make, and how many frames per second we will use. (Morrowind uses 15 fps) For simplicities sake, let's use 15 fps, and the entire animation will last for 8 seconds. So, 15 frames per second multiplied by 8 seconds = 120 frames. Now, click the "Time Configuration" button at the bottom right of the screen, a window will popup. Click the "Custom" selection and then enter 15 in the FPS spinner. Now change the 'End Time' spinner to 120 and click "OK". The window will close and the number of frames will change to 120.

Finally, the CamShaft, how many times do you want it to rotate? Also to keep everything as simple as possible, we will rotate the CamShaft two full rotations or 720° . So $720^\circ / 120 \text{ frames} = 6^\circ$ of CamShaft rotation per frame. We will base all of our trig calculations off the angle of the CamShaft which, as stated above, will occur in multiples of 6° .

Section 2.10: Determining the Formulae to Make the Animation Work Seamlessly

Given: Hrod=100, Hcam=12, $v=48^\circ$ (Experimental value, multiple of 6°)
 Find: Acam, O, Arod (the unlabeled vertical side of the Rod1 triangle), vRod

$$\begin{aligned} \text{Acam} &= \text{Hcam} \times \cos 48^\circ \\ \text{Acam} &= 12 \times 0.669 \\ \mathbf{\text{Acam} = 8.029 \text{ (Pixels)}} \end{aligned}$$

$$\begin{aligned} \text{O} &= \text{Hcam} \times \sin 48^\circ \\ \text{O} &= 12 \times 0.743 \\ \mathbf{\text{O} = 8.918} \end{aligned}$$

$$\begin{aligned} \text{Arod} &= \text{Square Root of } (\text{Hrod}^2 - \text{O}^2) \\ \text{Arod} &= \text{Square Root of } (100^2 - 8.918^2) \\ \text{Arod} &= \text{Square Root of } (10000 - 79.526) \\ \text{Arod} &= \text{Square Root of } (9920.474) \\ \mathbf{\text{Arod} = 99.602} \end{aligned}$$

$$\begin{aligned} \text{vRod} &= \text{ArcSin} (\text{O} / \text{Hrod}) \\ \text{vRod} &= \text{ArcSin} (8.918 / 100) \\ \text{vRod} &= \text{ArcSin} (0.089) \\ \mathbf{\text{vRod} = 5.116^\circ} \end{aligned}$$

Once again what the heck does this all translate to? Let's look at vRod first.

When v (Cam angle) = 48° , $\text{vRod} = 5.116^\circ$. Remember that we set the pivot point on the Rods near the top of the object. This vRod calculation will keep the Rods at the correct angle relative to the angle of the CamShaft.

Next the significance of Arod and Acam:

At maximum height, Arod and Acam added together make 112 pixels. So, if you take $112 - \text{Arod} + \text{Acam}$ you will get the distance the Piston, Pin, and Rod have moved down when the CamShaft has spun 48° . $112 - 99.602 + 8.029 = 4.369$. Without O, we simply cannot calculate anything to do with the Rods.

So, if you wanted you could go figure this all out long hand (calculate every 6° increment) and manually enter it into 3ds Max and eventually you would have a working animation after some long hours of work. I personally do not like that idea, so lets get 3ds Max to do it for us. This leads us into the next part of this section, MAXScripting.

Section 2.11 A Beginners Guide to MAXScripting

Save your work before starting this section. MAXScripting is not like Morrowind scripting, it is more complex, but not too complex. You may cringe at the thought of scripting, but don't, it is going to save you a butt load of time. The numbers calculated above, mean nothing to us right now. But we will be using the formulae we created above in our MAXScript.

Here is what you will need to know for what we are doing:

Variables: you don't need to declare them, but the way you call upon them will define what they are. Examples:

G=1 (G becomes an integer value)

G=1.0 (G becomes a float value with a number of decimal places)

G= "Your mom is my father" (G becomes a string value, we won't use this)

Commands: There are many, here is a few we will use:

(Do not worry about upper or lower case letters, MAXScript reads them as the same value.)

For I=0 to 100 do (loops 100 times, very useful)

Animate On (turns on animation, resets after script is finished)

At time 5 (places defined animation at 5th frame, 5 can be a variable too)

Move \$Piston1 [x,y,z] (Moves the called upon object how you tell it to)

Angle = eulerangles x y z (Defines rotational movement of variable)

Rotate \$Rod1 Angle (Rotates object by angle defined above)

Sqrt (number or variable) (gives the square root of the given number)

Pow 4 2 (returns 16, it means 4^2 , or 4 to the power of 2)

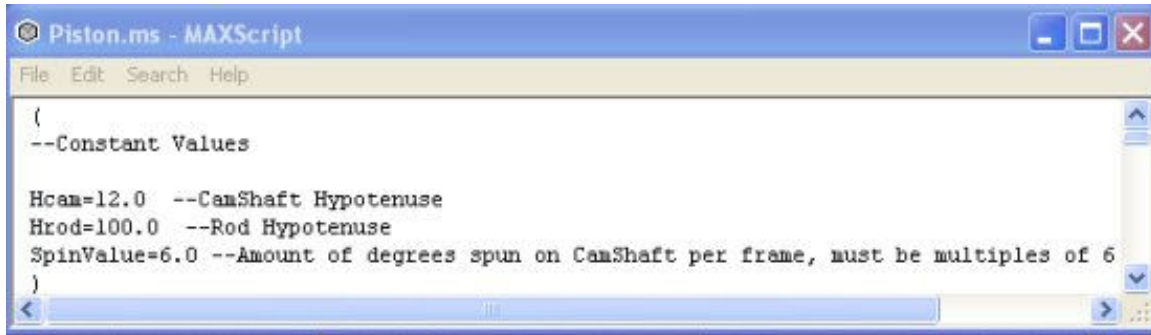
Sin/cos/tan (number or variable) (returns trig value)

asin/acos/atan (number or variable) (returns degrees of trig function)

-- (placing 2 hyphens will tell MAXScript not to read the following comment)

Section 2.12: Designing the MAXScript

Click on the “MAXScript” tab located at the very top middle of the 3ds Max window. That will cause some options to roll down; click “New Script” and the script creation window will open. First we will start by placing an open bracket “(“ then hit enter twice then place a closed bracket “)”. The entire script will fall between these two brackets. Next we will define our constant variable values. On line two type:

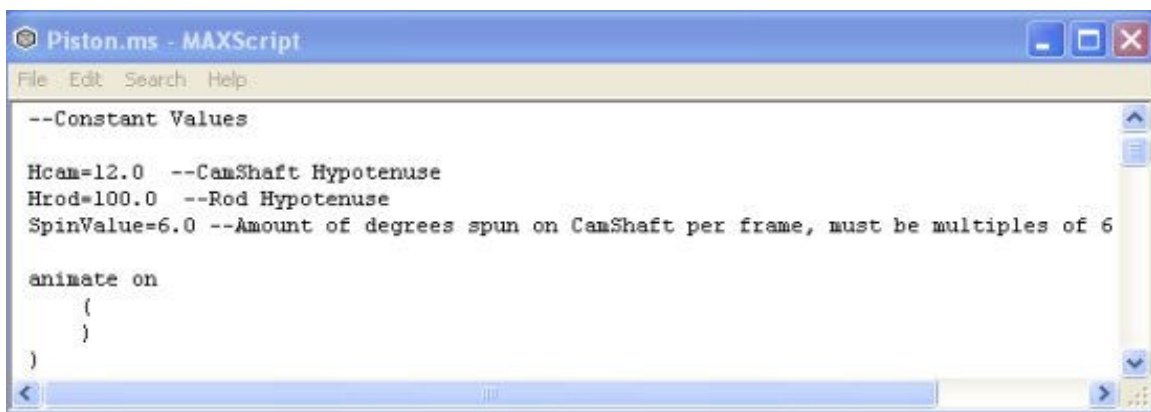


```
{
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6
}
```

As discussed above, the hypotenuse of the CamShaft and the Rod never change. So, we will declare their values at the top of the script and not change them. The one thing we can change is the SpinValue. We will write the script to make all the necessary changes by changing that one value. That means we will be able to exponentially up the speed of the entire piston assembly. In order to make the animation stop where it started, we must only set SpinValue to multiples of six. If you change it to 8, it will have a 180° hiccup from the end to the start of the animation when it loops. *Notice all the constants above have one decimal place, this ensures that they carry a float designation, calculation may screw up if this is not present.*

Next command to type in:

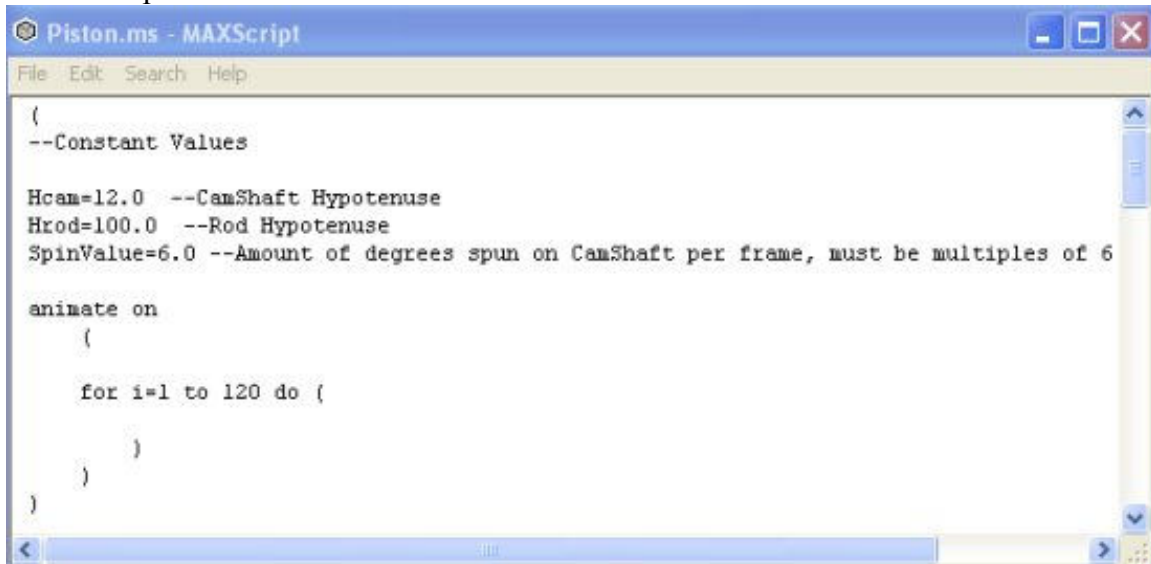


```
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
{
}
}
```

The animate on makes any movements in the script be translated into animation movements. Every bracket must have an end bracket. Next we enter the “for do” loop:




```
{
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
(
for i=1 to 120 do (
)
)
}
```

The 120 represents the number of loops to take place, “i” represents a variable that will equal 1 through 120. Notice how the number of frames equals the number of loops, this is not a necessity, but for my tutorial it is.

Next we move to all those boring calculations we made earlier:



```
{
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
(
for i=1 to 120 do (
--Piston1 Calculations
v = i * SpinValue --Cam Angle
vCam = eulerangles SpinValue 0 0 -- Clockwise

Acam = Hcam * cos v

O = Hcam * sin v

Arod = 112.0-(sqrt((pow Hrod 2)-(pow O 2)))+(Acam)

vRod = asin (O / Hrod)

Piston1Move = Arod
Piston1Rot = eulerangles vRod 0 0
)
)
}
```

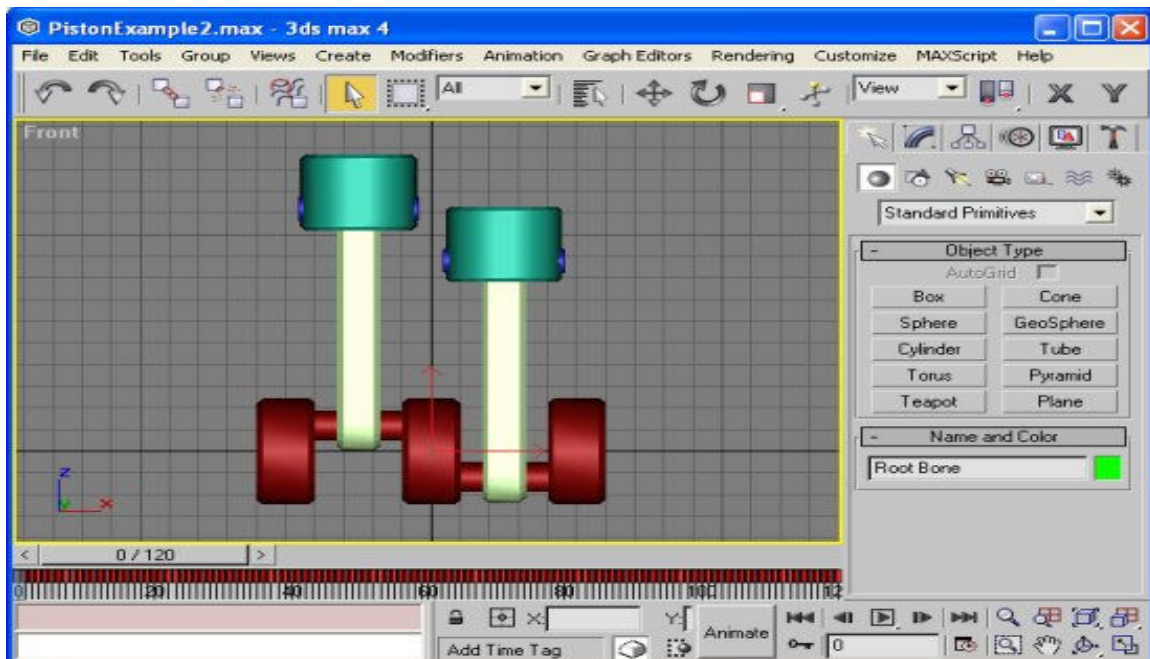

We get to see a bunch of the commands in the “for do loop” now. Now we see how the “for do loop” can really help us. Under Piston1 Calculations we see our first calculation using the looping variable “i”. This takes the SpinValue constant and multiplies it by the value of “i” which in the first frame is 1 then 2 then 3, after calculation equals 6, 12, and 18. Half of the calculations for Piston1 are generated from the value “v”.

Next we see the eulerangles command, it sets rotational parameters to the variable vCam. The parameters relate to the rotation on the x axis equal to the SpinValue.

vRod calculates the angle of the Rod using the asin or arcsin function. Without this calculation we cannot shift the angle of the Rod to line up with the CamShaft as it spins.

Finally we come to the Arod = calculation. This is a combination of three things. This is 112 (Or maximum height of rod and cam combined) minus the Arod PT calculation, plus the Acam calculation. As mentioned above, it calculates the distance the Rod, Pin, and Piston need to move up or down in relation to the angle the CamShaft has spun to. *Keep in mind “O” is the letter O standing for opposite side of the triangle. The (pow O 2) command stands for O(pposite) to the power of 2.*

Ok, let’s animate this sucker. The “at time i” function will place keyframes on the called upon time. In this setup, i will equal 1 through to 120 because of the for do loop. So, once this script has run, there will be 120 keyframes attached to the CamShaft, Piston1, and the Root Bone. Here’s the next bit:



```

Piston.ms - MAXScript
File Edit Search Help

{
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
(
for i=1 to 120 do (
--Piston1 Calculations
v = i * SpinValue --Cam Angle
vCam = eulerangles SpinValue 0 0 -- Clockwise

Acam = Hcam * cos v
O = Hcam * sin v
Arod = 112.0-(sqrt((pow Hrod 2)-(pow O 2)))+(Acam)

vRod = asin (O / Hrod)

Piston1Move = Arod
Piston1Rot = eulerangles vRod 0 0

at time i (
rotate $CamShaft vCam
move $Root_Bone [0,0,0]

--Piston1 Movement
Move $Piston1 [0,0,Piston1Move]
Rotate $Rod1 Piston1Rot
)
)
)
}

```

Let's review the new commands. The "rotate" command rotates the called object (\$CamShaft, you need the \$ for MAXScript to look for the named object in the scene) with the parameters set by the vCam eulerangles command. The move \$Root_Bone [0,0,0] sets a key on every keyframe for the Root Bone, (See picture on page 17) but the Root Bone does not move. **(Note: You will need a _ between Root and Bone, MAXScript errors if you don't)** The last two commands rotate and move the Piston1, Rod1, and Pin1 to the desired locations throughout the animation. After you have entered all this stuff in the script editor, or copy blocked it in, right click in the script editor and click select all. Now, in the keypad, press enter. The script will run and the script listener window will popup and say OK. Close the listener window and animation drag the slider along to see what happens.

Great gobs of galloping goose grease!! The CamShaft looks fine but the Rod, Piston, and Pin fly into the air like a bullet. That is because we have an error in our calculations, better yet, something is missing. Every calculation we made gives the distance moved from frame zero, when all we need is the distance moved from the last frame. So we will add a group of calculations that we can subtract from to find the actual movement per frame. Press undo and the animation should be erased. Try this instead:

```

Piston.ms - MAXScript
File Edit Search Help

(
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
(
for i=1 to 120 do (

--Piston1 Calculations
v = 1 * SpinValue --Cam Angle
vOld = (i-1) * SpinValue
vCam = eulerangles SpinValue 0 0 -- Clockwise

Acam = Hcam * cos v
AcamOld = Hcam * cos vOld --Last movement calculation

O = Hcam * sin v
Oold = Hcam * sin vOld --Last movement calculation

Arod = 112.0-(sqrt((pow Hrod 2)-(pow O 2)))+(Acam)
ArodOld = 112.0-(sqrt((pow Hrod 2)-(pow Oold 2)))+(AcamOld) --Last movement cal

vRod = asin (O / Hrod)
vRodOld = asin (Oold / Hrod)

Piston1Move = ArodOld - Arod --Piston movement per frame
PRot = vRodOld - vRod
Piston1Rot = eulerangles PRot 0 0

at time i (
rotate $CamShaft vCam
move $Root_Bone [0,0,0]

--Piston1 Movement
Move $Piston1 [0,0,Piston1Move]
Rotate $Rod1 Piston1Rot
)
)
)

```

So, I take every calculation, except vCam, and make it calculate based on v and vOld. If you examine vOld you will notice it calculates the same number as v less the SpinValue. That would be the actual location from zero in the last frame. So, when v=24, vOld=18.

The next entry I will point out is the `Piston1Move =` This takes the Old values and subtracts them from the new values. That with the `PRot=.....` will give us the actual movement of the Piston1 assembly. Once again select all and press enter in the keypad to see what we have done. The Piston1 and CamShaft assembly should work perfectly now. Press undo to erase the animation again.

Finally, let's complete the animation. Notice that the Piston1 will have the same movements as Piston2, except they are 180° apart from each other. So, we will copy and paste all the old calculations and slightly modify it to suit the movements of the Piston2 group:

```

Piston.ms - MAXScript
File Edit Search Help
(
--Constant Values

Hcam=12.0 --CamShaft Hypotenuse
Hrod=100.0 --Rod Hypotenuse
SpinValue=6.0 --Amount of degrees spun on CamShaft per frame, must be multiples of 6

animate on
{

for i=1 to 120 do (

--Piston1 Calculations
v = i * SpinValue --Cam Angle
v0ld = (i-1) * SpinValue
vCam = eulerangles SpinValue 0 0 -- Clockwise

Acam = Hcam * cos v
Acam0ld = Hcam * cos v0ld --Last movement calculation

O = Hcam * sin v
O0ld = Hcam * sin v0ld --Last movement calculation

Arod = 112.0-(sqrt((pow Hrod 2)-(pow O 2)))+(Acam)
Arod0ld = 112.0-(sqrt((pow Hrod 2)-(pow O0ld 2)))+(Acam0ld) --Last movement calculation

vRod = asin (O / Hrod)
vRod0ld = asin (O0ld / Hrod)

Piston1Move = Arod0ld - Arod --Piston movement per frame
PRot = vRod0ld - vRod
Piston1Rot = eulerangles PRot 0 0

--Piston2 Calculations
v2 = (i * SpinValue) + 180 --Cam Angle
v0ld2 = ((i-1) * SpinValue) + 180

Acam2 = Hcam * cos v2
Acam0ld2 = Hcam * cos v0ld2 --Last movement calculation

O2 = Hcam * sin v2
O0ld2 = Hcam * sin v0ld2 --Last movement calculation
)
)

```

```

Arod2 = 112.0-(sqrt((pow Hrod 2)-(pow O2 2)))+(Acam2))
Arod0ld2 = 112.0-(sqrt((pow Hrod 2)-(pow Oold2 2)))+(Acam0ld2)) --Last movement calculation

vRod2 = asin (O2 / Hrod)
vRod0ld2 = asin (Oold2 / Hrod)

Piston2Move = Arod0ld2 - Arod2 --Piston movement per frame
PProt2 = vRod0ld2 - vRod2
Piston2Rot = eulerangles PProt2 0 0

at time 1 {
    rotate $CamShaft vCam
    move $Root_Bone [0,0,0]

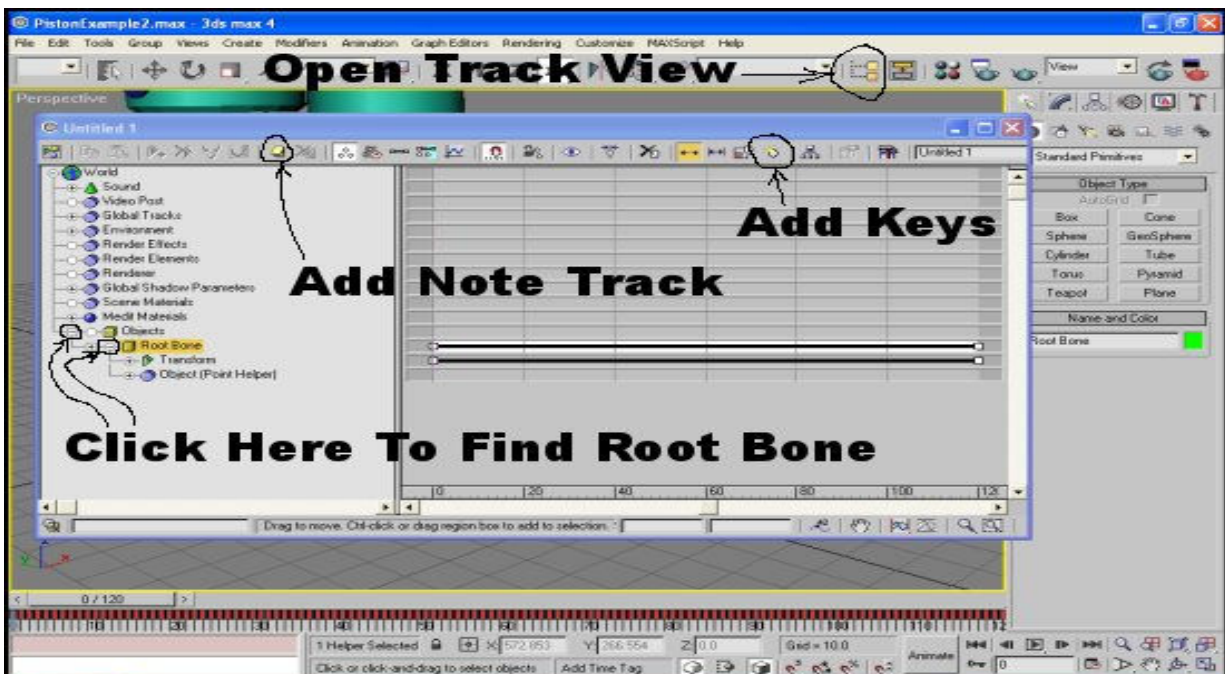
    --Piston1 Movement
    Move $Piston1 [0,0,Piston1Move]
    Rotate $Rod1 Piston1Rot

    --Piston2 Movement
    Move $Piston2 [0,0,Piston2Move]
    Rotate $Rod2 Piston2Rot
}

```

The biggest change lies in the first two calculations in the Piston2 section. Both calculations are the same as for Piston1 except there is 180° added to both to compensate for its starting position. Equally important is the changing of every variable in Piston2 section so it will not match and overwrite the calculations from Piston1. I added a 2 at the end of every variable in the Piston2 section. Try this one out, it is the finished product. *If you want to tinker, increase SpinValue in increments of 6. I set it at 96 once, it made the piston and cam spin much faster.*

Section 2.13: Making the Note Track

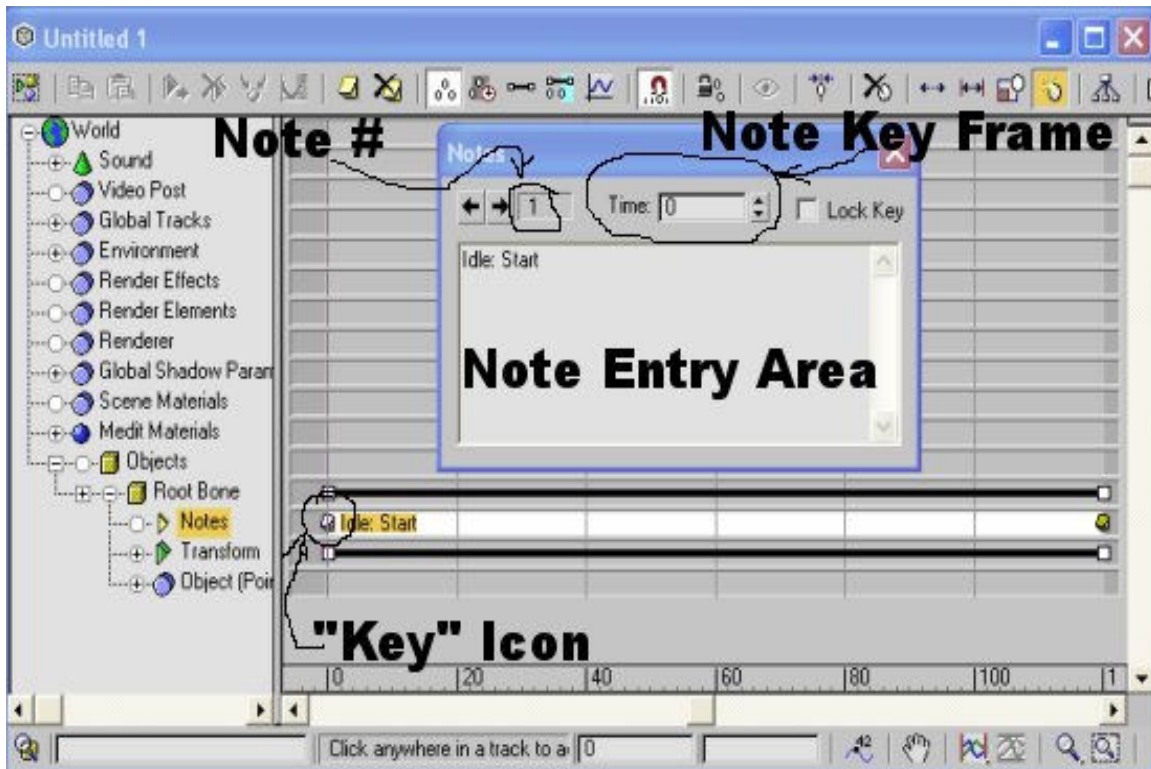


The animation should be working flawlessly now. Click the play button in the bottom right corner and watch the animation through whatever view port you want. *If it is not working it is probably because you entered the script in wrong or ran two different versions of the script on the mesh without undoing the old scripting.* When it works good, click the “Open Track View” button and then click the first small plus mark beside the “Objects” folder and then click the second small plus mark beside the “Root Bone”. That is illustrated above as “Click Here To Find Root Bone”. Click “Root Bone” to highlight it and then click the “Add Note Track” button. Click the “Notes” folder that just was created to highlight it and then click the “Add Keys” button.

Now, there should be a white line to the right of the highlight “Notes” folder with the key frames numbered underneath it. Use your mouse wheel to zoom in if you must, but left click on key frame 0 and key frame 120. A little icon should appear in the white area. If you missed the right key frame don’t worry, we can easily change that in the next step. Right click on one of the “Key” icons that you just created. A little window will popup, see picture below. In that window ensure “Note #” 1 equals “Time” or “Note Key Frame” 0. Also ensure that “Note #” 2 equals “Time” or “Note Key Frame” 120. The following step must typed perfectly and with the exact same capitalizations and spaces.

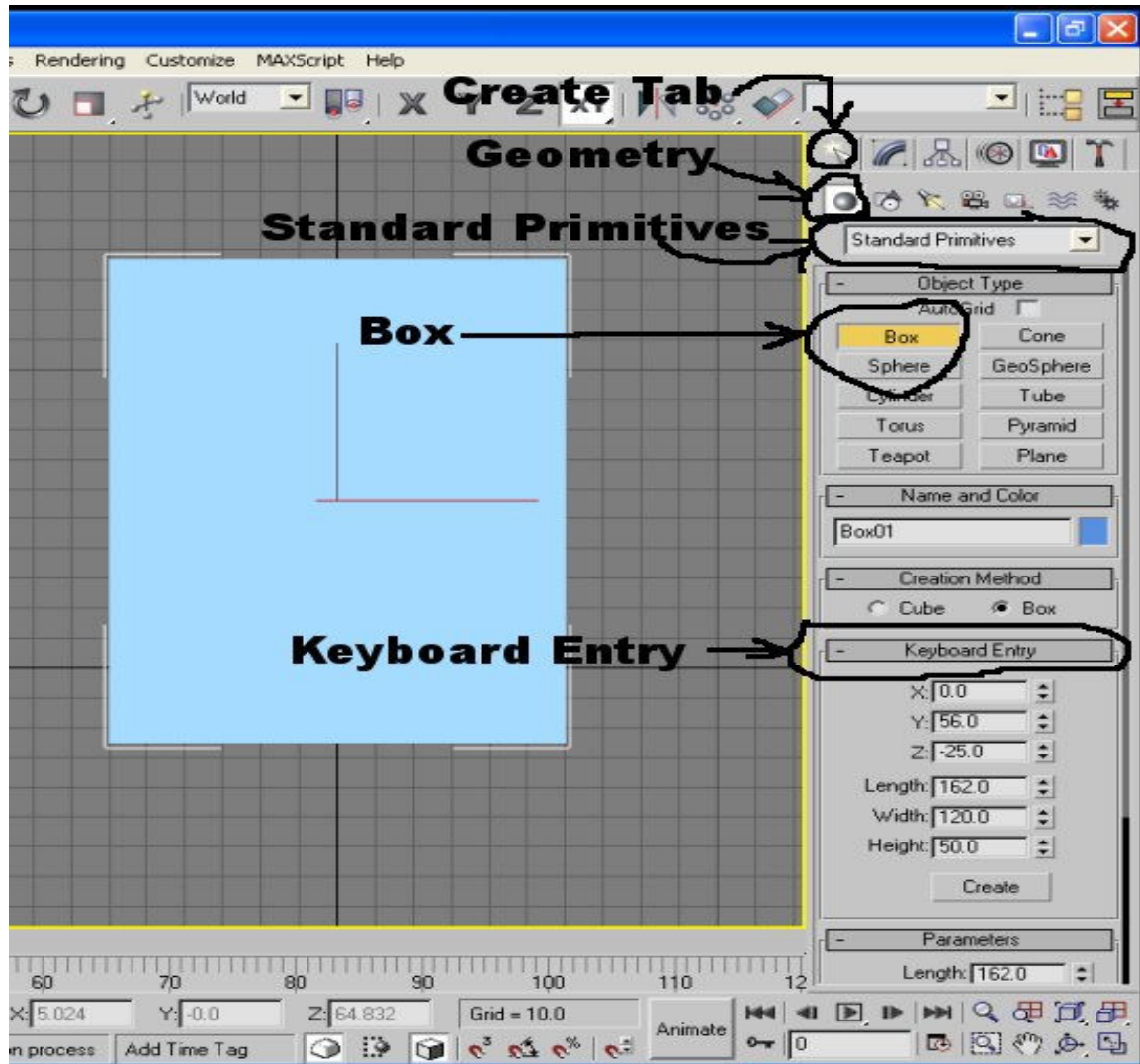
In the “Note #” 1 “Note Entry Area” type: **Idle: Start**
 In the “Note #” 2 “Note Entry Area” type: **Idle: Stop**

Close the “Track View” window and save your work.



Section 2.14: The Bounding Box

I am going to explain how to create and rename a box. *If you don't know how to do this you may need to consider reading some basic 3ds Max tutorials.* Refer to the following picture to see what I am talking about:

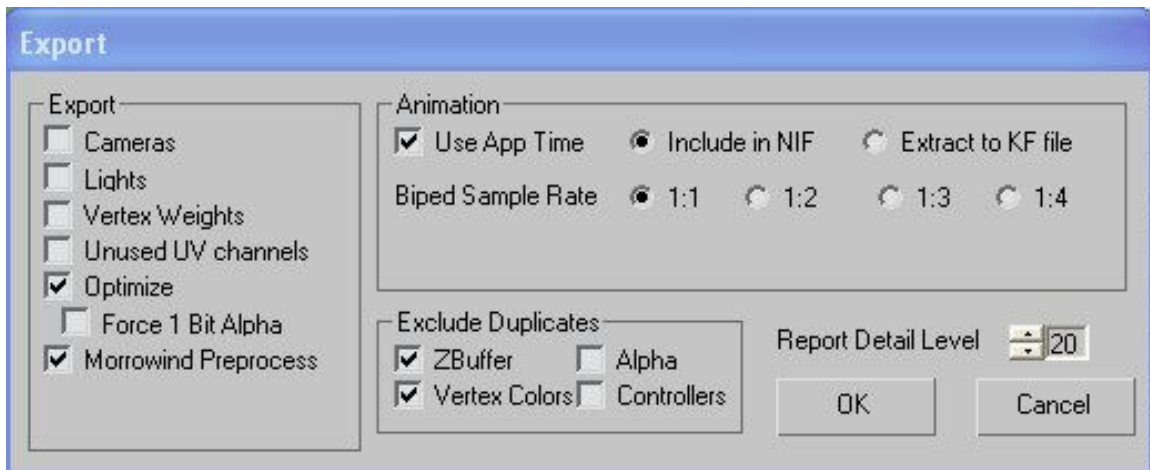


Press “f” to go to the front view port. Click the “Create” tab, click Geometry if needed, ensure “Standard Primitives” is selected in the spinner, click “Box”, and finally click “Keyboard Entry”. In the Keyboard Entry area type 56 in the X, -25 in the Y, 162 in the Length, 120 in the Width, and 50 in the Height then press the “Create” button just below it. It should look exactly like it is in the picture above. Click in the “Name and Color” area and change the name “Box01” to Bounding Box, it must be that exact spelling. With the Bounding Box selected, right click anywhere in the screen and click “Hide Selection” in the popup window. Save your work. *You don't actually need to make the Bounding Box that way, you can just eyeball it as long as you just cover the entire mesh.*

Section 2.15: Exporting The Animated Mesh to NIF

Well, let's see if this actually works. Click the "File" tab (Or press Alt-f) and click "Export" which is about half way down the list. An export window will popup. First, in the "Save in:" area find your Morrowind directory, if you did a standard install it will be in- Program Files/Bethesda Softworks/Morrowind/Data Files/Meshes. Next, go to the "File name:" area and give it a name like Piston. Now, go to the "Save as type:" area and scroll down to "TESExporter (*.NIF)". Lastly, click the "Save button".

Now the TESExporter window will popup. It will look like this:



Everything should be set up like the picture above. The "Report Detail Level" does not need to be at 20, that is for debugging, if you know how to read it. Now press the "OK" button. You will see some dialogue popup, and it is finished.

Section 2.16: Checking the Animation File in Game

Make a new activator in the TESCS with the new NIF file you just created. Place it somewhere in the world map like Balmora. If you get one or more error messages while placing it in the world map, you will need to figure out what it is. If you do get error messages the animation will typically not work. If you get no error message, save your new esp file and load it up and go find it in the game. You should be looking at a poorly coloured double piston that is always moving. Congratulations, you just made a perpetual motion machine.

Reasons for error messages:

1. Key Notes or Bounding Box are misspelled or do not have proper spacing
2. There is no Bounding Box made
3. Root Bone is not the parent object of all other objects in scene
4. You cannot follow instructions ☺