Chapter

12

Animating the Beast

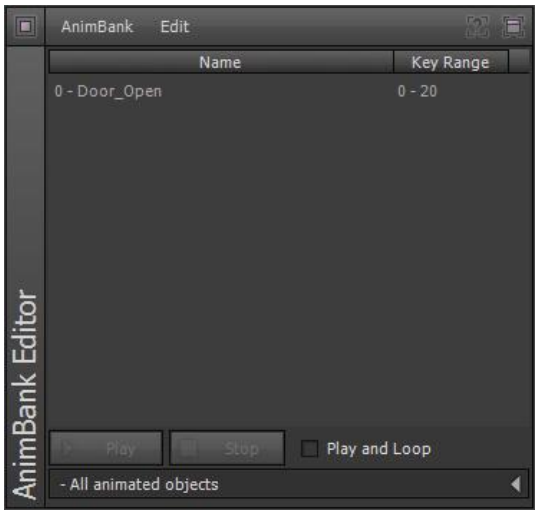
In this Chapter I'm going to be showing you how to use ShiVa's Animation Modules, AnimBank and AnimClip. I'll be starting with the AnimBank module, and then moving on to the AnimClip module.

AnimBank

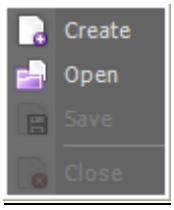
The AnimBank Editor module allows you to edit the contents of an AnimBank. An AnimBank is a set of animations called "AnimClips". These animations can be imported as-is with 3D Models from a DCC tool (such as 3DSMax, XSI, etc.) or can be built from scratch in the AnimClip Editor.

In the AnimBank, the "AnimClips" are indexed, which makes it possible to keep a consistency between all AnimBanks of the same type. For example, for all AnimBanks related to characters, it is a good idea to place the walk animation at index 0, run at index 1, jump at index 2 etc., so that the clips are easy to identify across AnimBanks. Also, spaces can be created between "AnimClips" simply by modifying their index. This may be useful if you wish to insert "AnimClips" at a later time, and still keep the consistency of the order.

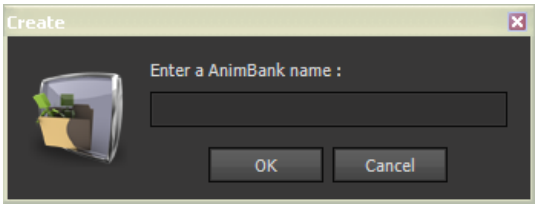
For each "AnimClip", the interval denotes two keys, the key for the beginning of the animation, and the key of the end of the animation. Knowing these keys will be of great help when using Scripts to play part, or all, of an "AnimClip".



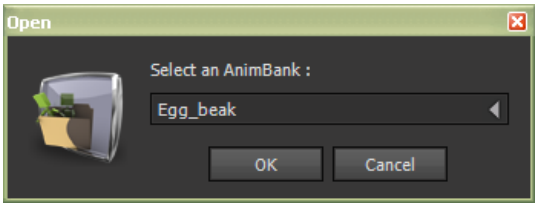
AnimBank Menu



Create: This option allows the creation of a new AnimBank. A dialog will appear asking for the name of the new AnimBank:



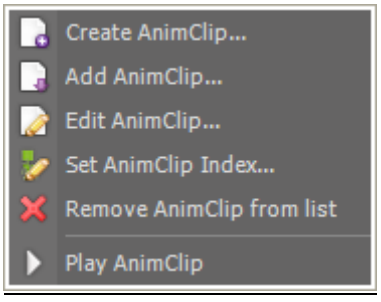
Open: This option allows any AnimBank that is available in the current project to be opened in the editor. A dialog will appear that has a drop-down selection of all of the AnimBanks available in the current project:



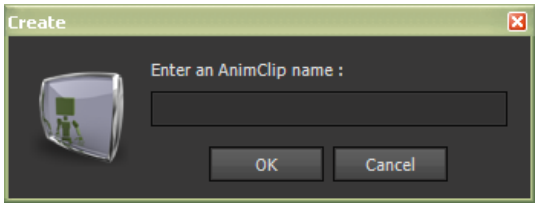
Save: This option allows the saving of the current AnimBank.

Close: This option closes the current AnimBank.

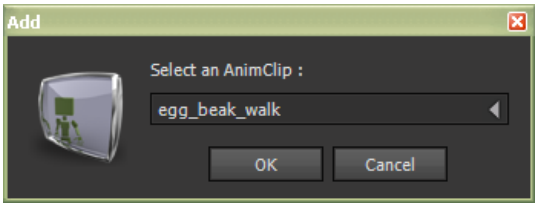
Editing Menu



Create AnimClip...: This option allows the creation, and addition, of an empty AnimClip into the current AnimBank. A dialog will appear asking for the name of the new AnimClip.

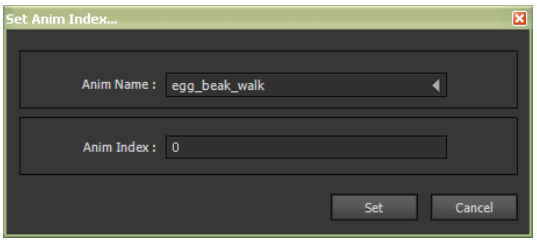


Add AnimClip...: This option allows the addition of any AnimClip that is available in the current project, into the current AnimBank. This acts like a drag'n'drop from the Data Explorer. A dialog will appear that has a drop-down selection of all of the AnimClips available in the current project.



Edit AnimClip...: This option opens the selected AnimClip in the AnimClip Editor to enable modification of the animation.

Set AnimClip Index...: This option allows the index of the selected AnimClip to be changed.
NOTE: If the new index is already assigned, you will be asked if you want to swap the indices of the two AnimClips.



Remove AnimClip from list: This option will remove the selected AnimClip from the current AnimBank, and replaces it with an empty slot.

Play AnimClip: This option allows the playing (or stopping if already playing) of the selected AnimClip. If "play and loop" is checked, then the AnimClip will be played until it is stopped, otherwise it will play just once.

To be able to see an animation run, a Scene or Model must be open in the Scene Viewer. The AnimBank Editor will suggest a list of Objects to which the AnimBank can be attached.

NOTE: You will only be able to play an AnimClip if it is in an AnimBank.
NOTE: Whilst an animation is playing, you can continue to work.

AnimClip

The AnimClip Editor module allows the creation, or modification, of the animations of Objects. AnimClips are the way to describe how Objects should move. They are made up of Channels which control the TRS (Translation Rotation Scale) of each part of the Object.

Basically, the AnimClip Editor allows the creation, and editing, of key frame animation. For example, if you have an Object animated by a Skeleton, you will have one channel for each node of the Skeleton.

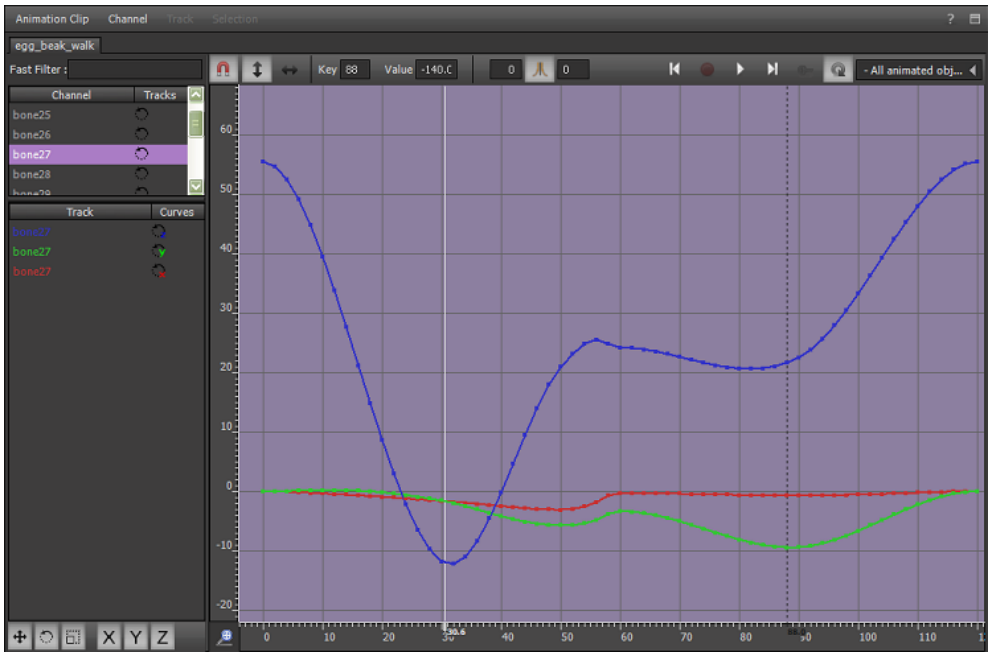
For a group of simple Objects, you will have one Channel for each Object in the group. Simple animated Objects (ie: those which don't have Skeletons) can be controlled solely by one Channel, which is selected in the 'Animation Controller' panel of the Attributes Editor.

NOTE: all AnimClips intended for use with the same Object must have the same set of Channels.

Each AnimClip Channel has Tracks for each transformation that it manages, and for each transformation (translation, rotation, scale), there are 3 Tracks for each axis. So, a Channel can have a maximum of 9 Tracks.

A Track is represented by a curve on a graph with the keyframe on the X-axis and the value on Y-axis. By default, the framerate is 60 frames per second (though this can be changed using the animation.setPlaybackSpeed() Function in Script).

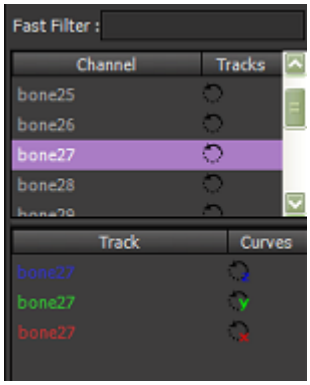
NOTE: for any rotational transformation, the values are expressed in degrees ranging between -180° and 180°.



The AnimClip Editor is made up of several distinct areas:

Menus
Toolbars
Selections

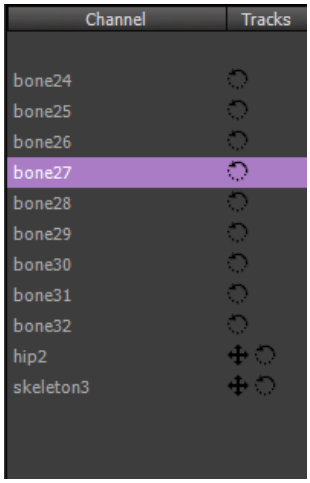
Over the next few pages I'll be describing the "Menus" and "Toolbars", but first I'll explain the "Selections" area, which can be found to the left of the AnimClip Editor:



Fast Filter

The "Fast Filter" is used to show Channels whose names contain the given string.

Channels

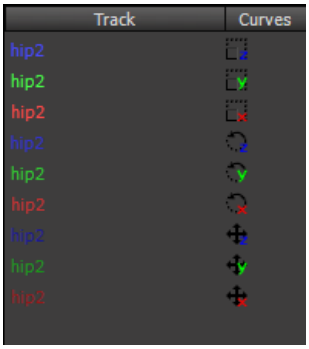


The "Channels" area is used to select the Channel of the animation that is to be displayed in the graph area. The "Channel" column displays the name of the relevant Channel, and the "Tracks" column displays the transformations that have been set for this Channel.

The symbols in the "Tracks" column denote the type of transformation(s) that have been applied (T = Translation, R = Rotation, S = Scale):

By selecting a Channel in this area, the corresponding Channel transformations are displayed in both the graph area, and in the "Tracks" area, as shown below:

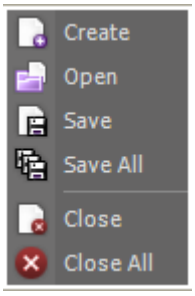
Tracks



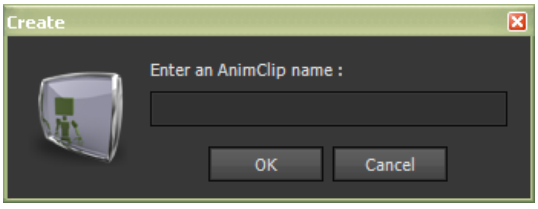
As you can see, the various axes are colour-coded, with Red denoting X, Green denoting Y, and Blue denoting Z. This is also carried over into the graph area.

By clicking on a Track, the corresponding Track will be highlighted in the graph area, allowing easy selection of Tracks for adjustments to be made.

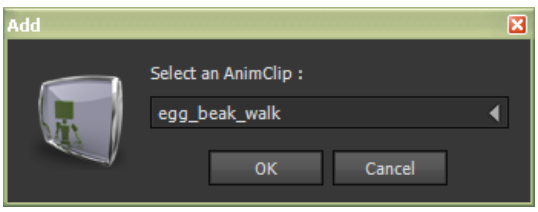
Animation Clip menu



Create: This option allows the creation of a new AnimClip. A dialog will appear asking for the name of the new AnimClip:



Open: This option opens any AnimClip that is available in the current project. A dialog will appear that has a drop-down selection of all of the AnimClips available in the current project:



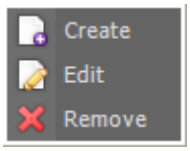
Save: This option saves the current AnimClip.

Save All: This option saves all opened AnimClips.

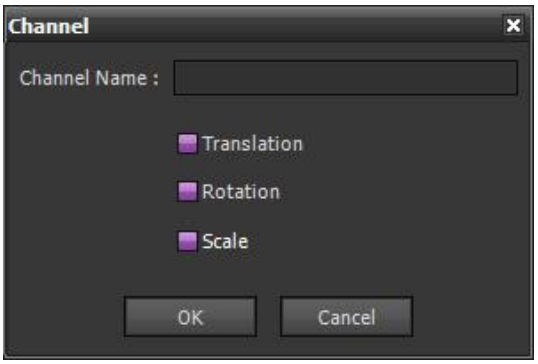
Close: This option closes the current AnimClip.

Close All: This option closes all opened AnimClips.

Channel menu



Create: This option allows the creation, and addition, of a new channel to the current AnimClip. In the dialog box that appears, a name can be assigned to the Channel. Take care to name the Channel according to the Objects that it will control.

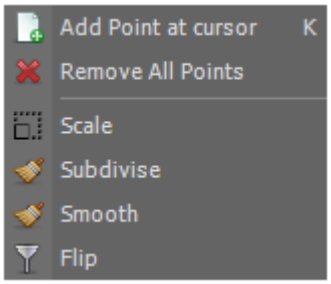


The three option buttons allow the definition of what type of transformation(s) will be performed (Translation, Rotation, Scale), and hence the number of Tracks available. This is useful if you want to restrict the Channel to specific types of transformation(s), or to reduce the number of visible Tracks.

Edit: This option allows the editing of the name of the selected Channel and the addition, or removal, of one, or more, of the transformation type(s).

Remove: This option allows the removal of the selected Channel from the current AnimClip.

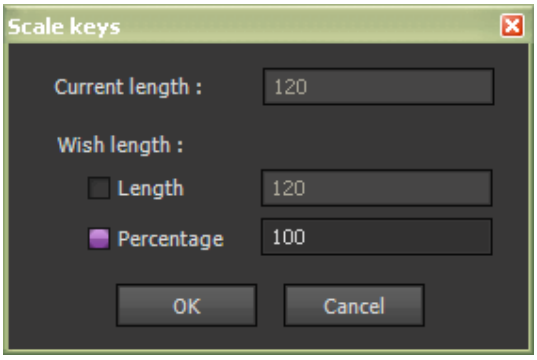
Track menu



Add Point at cursor: This option allows the addition of a new point on the curve of the selected Track at the position of the reading cursor.

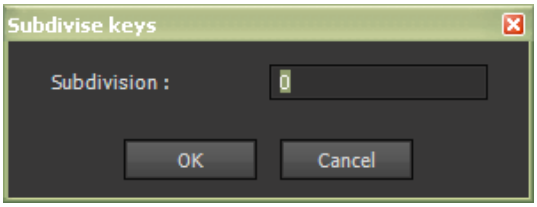
Remove All Points: This option allows the removal of all points from the selected Track.

Scale: This option allows the scaling of the curve of selected Track. A dialog will appear that allows the setting of the new range:



The scale factor depends on the ratio between the current key range and the wish key range. For example, current key range = 20 and wish key range = 40 give a scale factor of 200%. Either a numeric value, or a percentage, can be input as the new "Wish length".

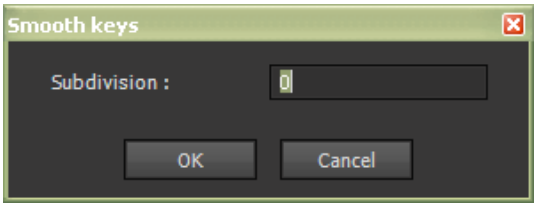
Subdivide: This option allows the subdivision of the curve of the selected Track, by adding keys at empty KeyFrame points. A dialog will appear that allows the setting of the number of keys to be added:



The Subdivision value entered in this dialog determines how many extra points will be inserted taking into account the number of points already in the curve i.e. if the curve already contains 6 points then any number up to 6 will add no extra points, numbers greater than 6 will add extra points based on:

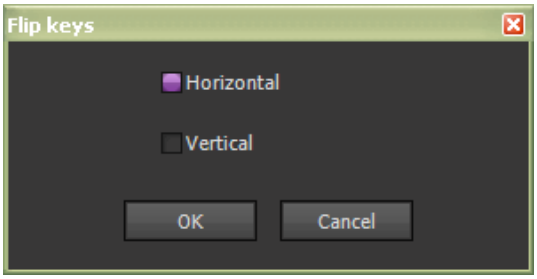
Subdivision – existing points = new points

Smooth: This option allows the smoothing of the curve of the selected Track, by adding keys at empty KeyFrame points. A dialog will appear that allows the setting of the number of keys to be added:

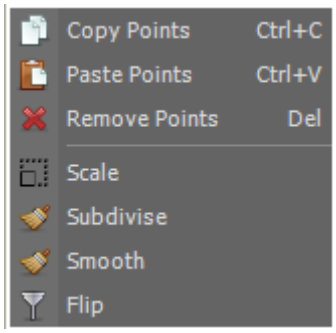


Smoothing works in a similar way to Subdivision.

Flip: This option allows the flipping of the curve of the selected Track either vertically or horizontally. A dialog will appear that allows the setting of the direction of the flip:



Selection menu



The Selection Menu is only enabled when points have been selected in the display by left-clicking and dragging a rectangle around the required points.

Copy Points: This option allows the copying of the positions of the selected points.

Paste Points: This option allows the pasting of the previously copied points at the position of the reading cursor. Any selected points in the range of the copy are removed.

Remove Points: This option allows the removal of any selected points.

Scale: This option is the same as Scale for the Track, but is only applied to the selected points.

Subdivide: This option is the same as Subdivide for the Track, but is only applied to the selected points.

Smooth: This option is the same as Smooth for the Track, but is only applied to the selected points.

Flip: This option is the same as Flip for the Track, but is only applied to the selected points.

Track toolbar

1. Translation: This option displays, or hides, the Tracks for translation animations.

2. Rotation: This option displays, or hides, the Tracks for rotation animations.

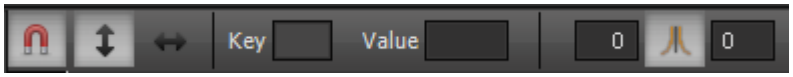
3. Scale: This option displays, or hides, the Tracks for scale animations.

4. X axis: This option displays, or hides, the Tracks for X axis animations.

5. Y axis: This option displays, or hides, the Tracks for Y axis animations.

6. Z axis: This option displays, or hides, the Tracks for Z axis animations.

Graph toolbar



1. Magnetic grid: This option activates, or deactivates, magnetism on the grid. When activated, dragged points will automatically snap to the grid.

2. Value tool: If this option is activated, the selected points can only be dragged vertically. Only the values of the selected point(s) will change.

3. Key tool: If this option is activated, the selected points can only be dragged horizontally. Only the KeyFrames of the selected point(s) will change.

NOTE: holding the Shift key while dragging points will activate the two tools.

4. Key: This is the KeyFrame of the main selected point.

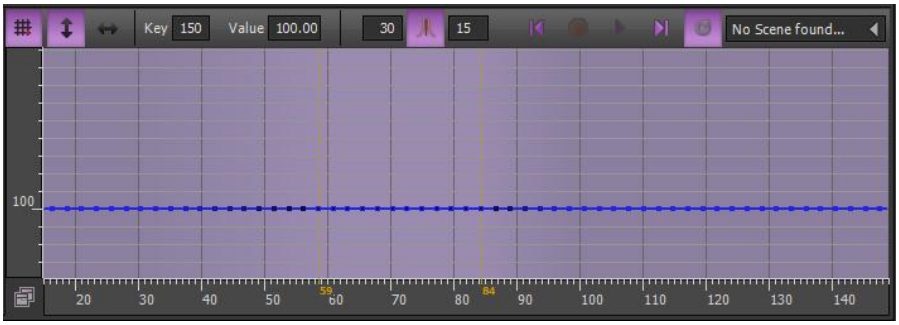
5. Value: This is the value of the main selected point. This can be edited, allowing the setting of the value of the main selected point. All of the other selected points will follow this modification. Points within the zone of influence will also be affected.

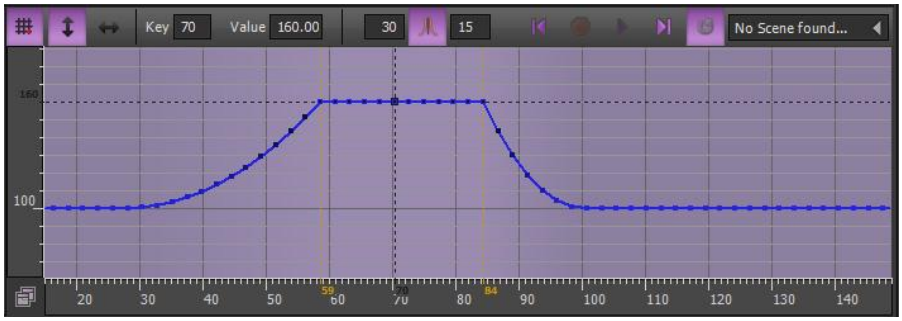
6. Left influence: This option allows the setting of the range of influenced frames to the left of the selection.

7. Influence: This option is only available when the value tool is active and enables, or disables, the zone of influence.

8. Right influence: This option allows the setting of the range of influenced frames to the right of the selection.

NOTE: the values of any points in the zone of influence will be affected by the modification of any values of any selected points. This operation uses a Gaussian-type equation.





1. Previous keyframe: This option allows the setting of the reading cursor to the first KeyFrame found before its current position.

2. Record: This option allows the addition, or settings, of a new point on the selected Track at the reading cursor position. The value of the new point is set according to the position of the selected Object.

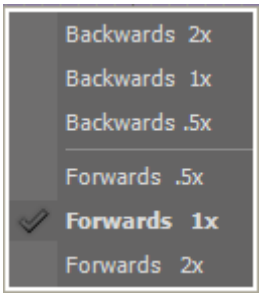
NOTE: If the left mouse button is held for a few seconds on 'Record', the following dialog will appear:

Clicking on 'Auto' will activate auto recording:

When in auto recording mode, any transformations carried out on the selected Object in the Scene Viewer, will modify the keys at the reading cursor position or, you can press 'K' to record the Object's position.

3. Play: This option allows the playing of the animation on the selected Object or on all Objects which have an AnimBank containing the current AnimClip. If certain keys are selected, then playing will loop within the bounds of the selection.

NOTE: If the left mouse button is held for a few seconds on 'Play', the following dialog will appear:



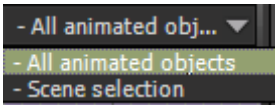
The options in this dialog allow the selection of the playback speed for the preview playback.

4. Next keyframe: This option allows the setting of the reading cursor to the first KeyFrame found after its current position.

5. Record: This option allows the adding/setting of a new point, at the current reading cursor position, on the selected track. The value of the new point is automatically set based on the position of the selected Object.

6. Loop: This option allows the activation, or deactivation of looping for preview playback.

7. Object selection: This option lists the Object candidates for which you can record positions or preview playbacks:



NOTE: the Objects in this list will all have an AnimBank containing the current AnimClip.

NOTE: if an Object has the correct AnimBank but doesn't react to the animation, check the **affected channel** in the 'Animation Controller' in the Attributes Editor!

Now I'll walk you through the demo for this Chapter. The one I'll be using is the "Simple Animation" demo, which is one of the samples in ShiVa.

This demo consists of the following files:

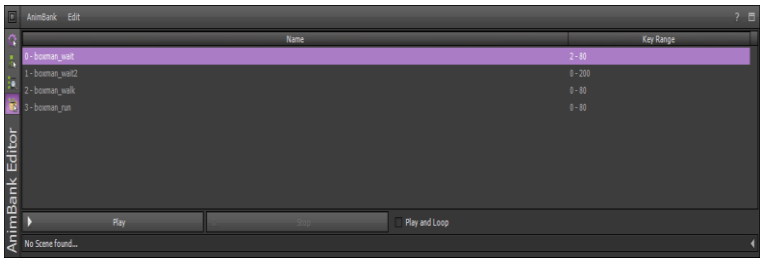
Games	-	SimpleAnimation
Models	-	StaticPointLight BoxMan
AIModels	-	SimpleAnimation_Character SimpleAnimation_Main
AnimBanks	-	perso
AnimClips	-	boxman_run boxman_wait boxman_wait2 boxman_walk
HUD	-	SimpleAnimation
Materials	-	BoxMan cornell_box_Material000 cornell_box_Material001 cornell_box_Material002 cornell_box_Material003 cornell_box_Material004

Meshes	-	cornell_box_Mesh000 cornell_box_Mesh001 cornell_box_Mesh002 perso_Mesh000
Scripts	-	SimpleAnimation_Character_Handler_onChangeAnim SimpleAnimation_Character_Handler_onEnterFrame SimpleAnimation_Character_Handler_onInit SimpleAnimation_Main_Handler_onChangeAnim SimpleAnimation_Main_Handler_onInit
Skeletons	-	perso_Skeleton000
Scenes	-	SimpleAnimation

Again, the ones highlighted in Red will be the files that I'll be dissecting here. Since this part of the Chapter is about Animation, I'll start with the AnimBank, and the AnimClips:

So, open up the AnimBank Editor, and we'll get started.

If you click on the "AnimBank" menu, and then "Open", you can select the "*perso*" AnimBank. Your AnimBank Editor should look like this:



As you can see, there are four AnimClips in this AnimBank:

0 – boxman_wait
1 – boxman_wait2
2 – boxman_walk
3 – boxman_run

If you open up the Scene Editor in another window, and select the "BoxMan" model, you will be able to view each of the animations by selecting them in the Editor, and clicking the "Play" button. Don't forget to check "Play and Loop", or you may miss the action!

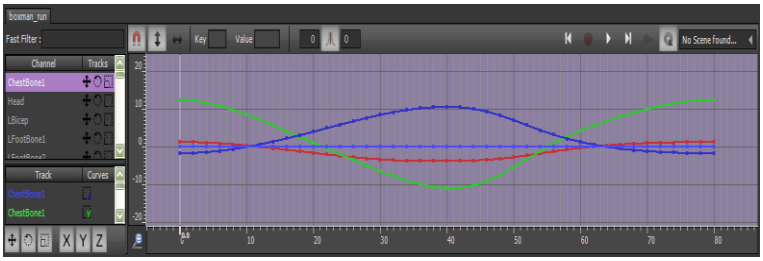
Also, you should note that the "Key Range" settings for the AnimClips are as follows:

boxman_wait	–	2 – 80
boxman_wait2	–	0 – 200
boxman_walk	–	0 – 80
boxman_run	–	0 – 80

The reason for this will become apparent when we delve more deeply into the various AnimClips.

Talking of which, now would be a good time to move on to the AnimClip Editor, as there's really not too much to look at in the AnimBank Editor, when it has already been created!

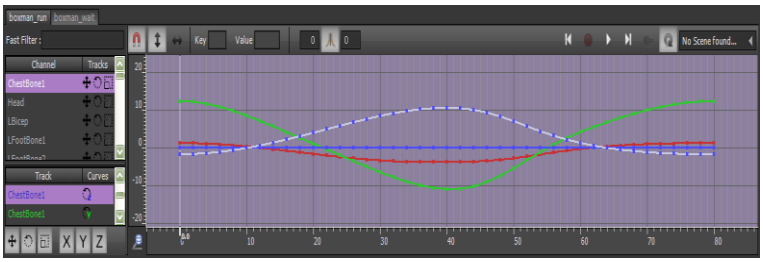
OK, open up the AnimClip Editor, click on the "Animation Clip" menu, and then on "Open", select the "boxman_run" AnimClip, and your AnimClip Editor should look like this:



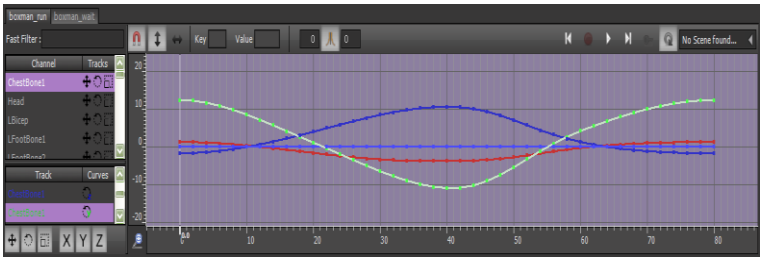
As you can see, there is an awful lot associated with this animation. For instance, if you scroll down in the "Channel" area, you will see that there are, in fact, over 50 Channels! I'll pick one of the more interesting ones, and leave you to look at the rest:

As you can see from the above image, this Channel ("ChestBone1" – i.e. the chest bone of "boxman") has flat "worms" for the Scale and Translation of the x, y and z axes, and these "worms" are all located on the "0" line of the graph. For this demo, there is no need to Scale or Translate the Character whilst it is moving, so this will be common across all Channels. This is because the Character stays in one place at all times, so all that we need to change is the Rotation. You can check this out by selecting each "Track", and seeing which "worm" is highlighted.

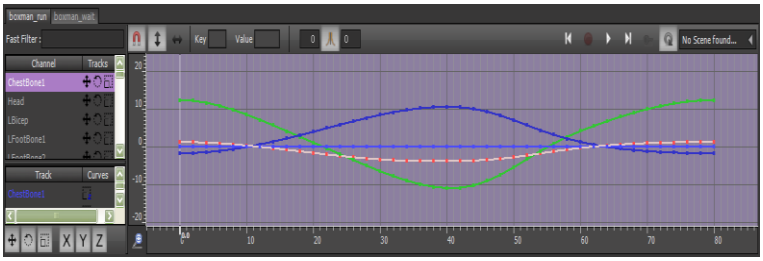
The Rotation, however, is somewhat different. The z axis Rotation has a blue "worm" that changes over time, between approx. -1 and +11, thus producing movements in this axis (the white line with blue dots below – the selected Channel turns white in the graph display):



The y axis is somewhat more pronounced and forms a negative parabolic curve about the 0 line:



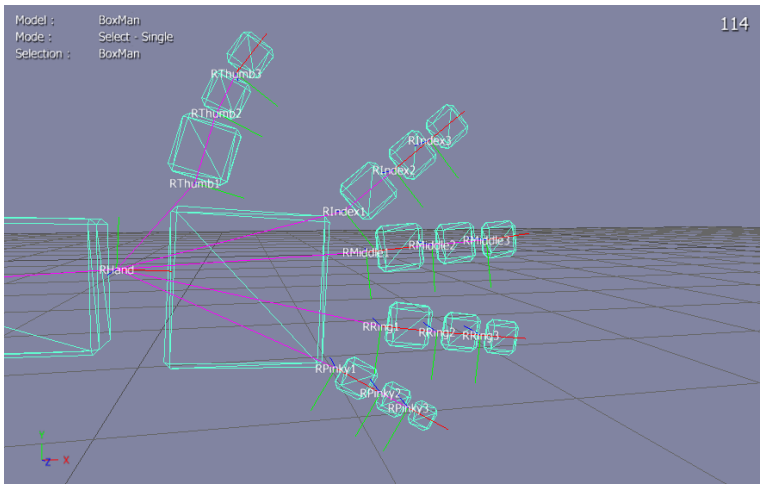
The x axis is the least pronounced of the lot, and forms a small curve that is slightly offset below 0:



As you can see from the above graphs, the "ChestBone1" Channel rotates, to a certain extent, about each axis, with the "z" and "y" axes somewhat mirroring each other. The "x" axis rotation is similar to the "y" axis, but a lot smaller. If you watch the AnimClip closely when it is playing, you can see how this Channel moves over time. It is extremely difficult to show this on paper, so I would suggest that you examine this AnimClip very closely, looking at the graphs for each Channel (that has a rotation, as quite a few don't – such as "LFootBone2"), and then watching that particular Channel in the running Animation. To see which Channel is which, you'll have to turn "on" the "Skeletons Joint Names" option in the "Display" menu of the Scene Viewer.

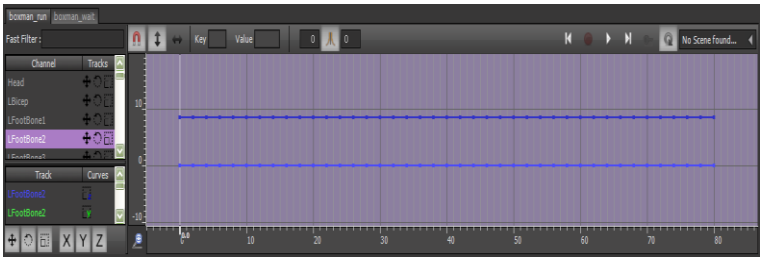
As I have previously stated, there are over 50 Channels to each AnimClip, with each Channel corresponding to a joint name of "BoxMan". If you look closely at "BoxMan", you will see that he (I'm assuming he's a "he", as his name is "BoxMAN"!) has been designed to pretty much mimic a human. For example, there are 3 bones in each finger (i.e. RIndex1, RIndex2 & RIndex3 for the right index finger). "BoxMan's" Skeleton is stored in ShiVa with the name "perso_Skeleton000".

Here is a "close-up" of the right hand of "BoxMan" (taken from the Scene Viewer, using a View Mode of "Wireframe", and both "Skeletons" and "Skeletons Joint Names" selected in the "Display" menu):



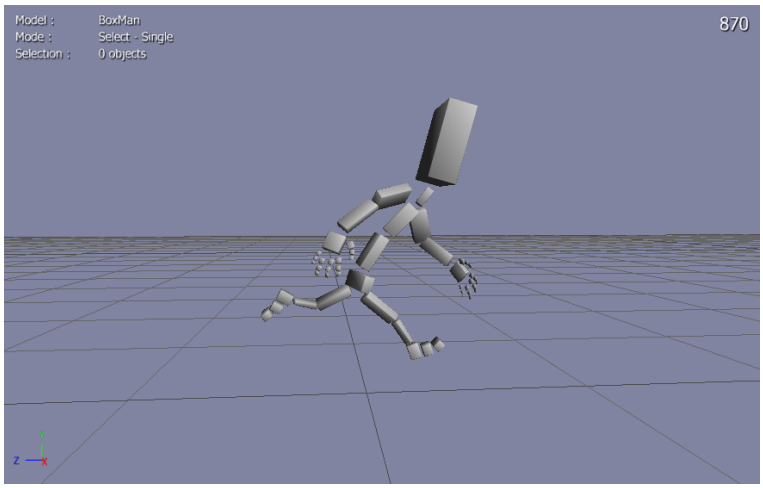
As you can see, the "Skeleton" is displayed using pink lines, and the "Skeletons Joints Names" are displayed using white text. The green lines protruding from the "Skeletons Joints Names" are placed so as to where the Joint is located, and are at right angles to the actual Skeleton. If you look closely at the right thumb (RThumb), you will see that the angle of the Skeleton changes slightly at each Joint, giving the impression that the thumb is slightly bent, whereas, the fingers are all pretty straight. Look at your own right hand, and you'll see that this pretty much follows how a normal person's hand looks (assuming that only "normal" people and not three-fingered aliens are reading this book of course!).

Just as an example of a single bone where there is very little movement in the AnimClip graph, consider "LFootBone2":

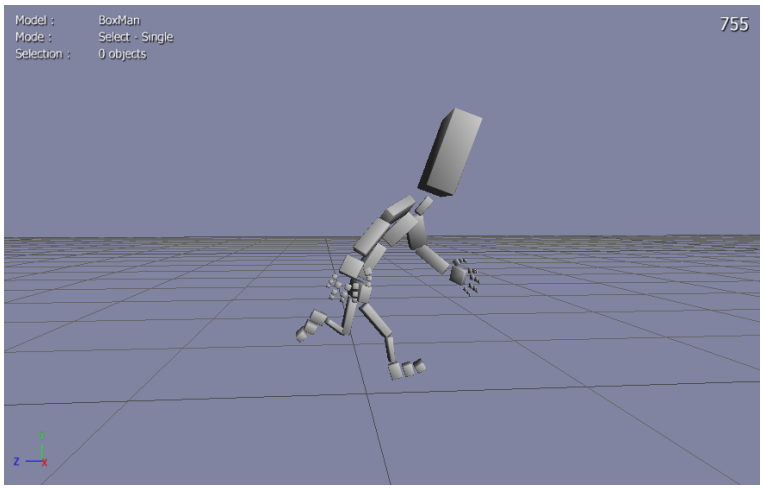


The only line that is not placed on the "0" line of the graph, is the dark blue line that sits at +8.5 (remember that these values are in degrees as we are talking about Rotations). This line relates to the Rotation about the "z" axis. As you can see, this is a straight line, meaning that it does not change throughout the Animation, but it does indicate that there is an initial Rotation about the "z" axis when the Animation starts.

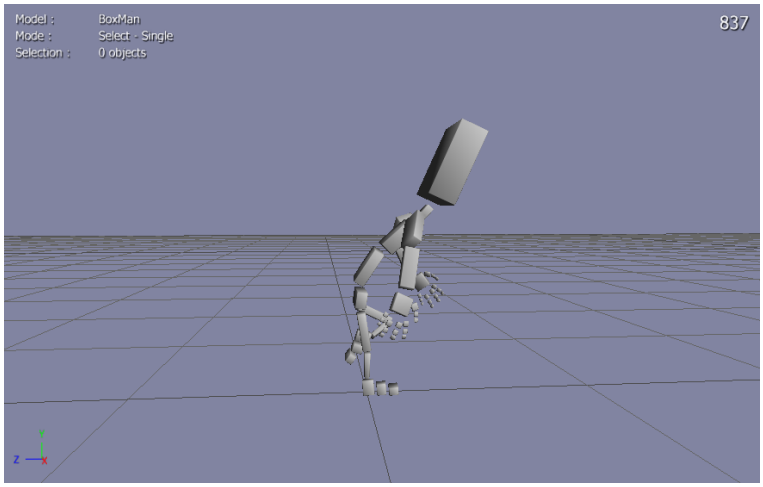
The last thing that I will show you is how the "Run" AnimClip changes over time. The following screenshots show this Animation from KeyFrame 0 to KeyFrame 80 (in steps of 10):



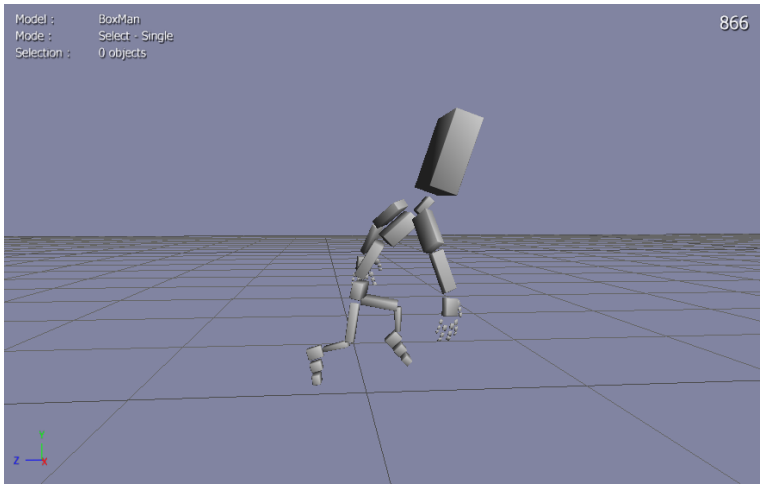
KeyFrame 0



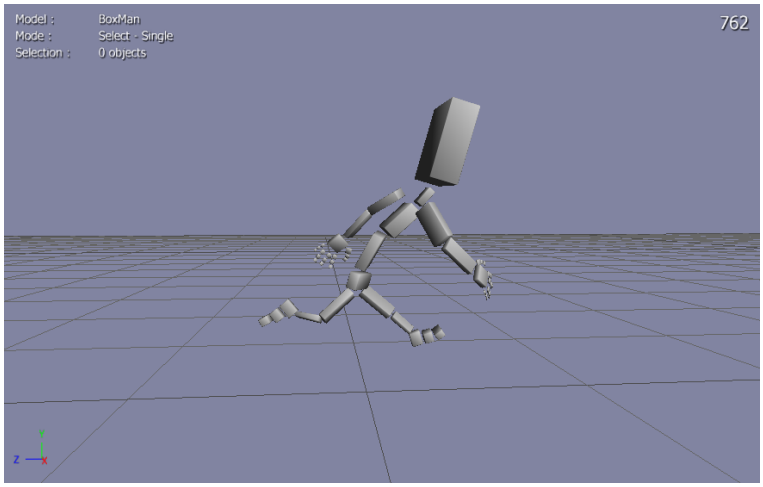
KeyFrame 10



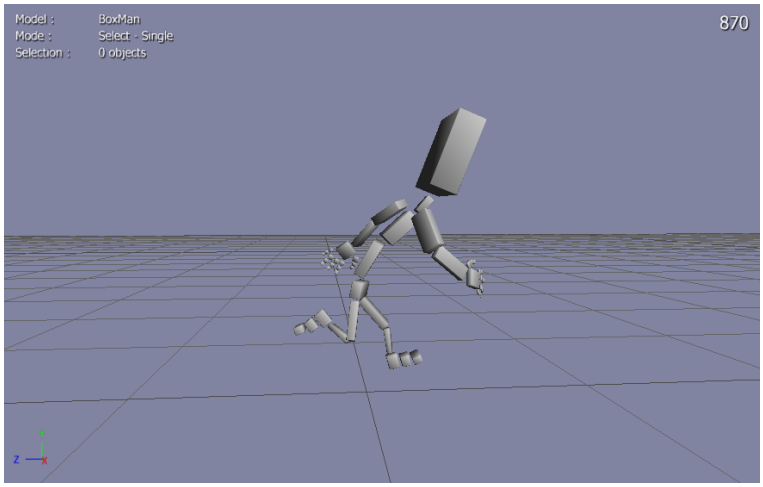
KeyFrame 20



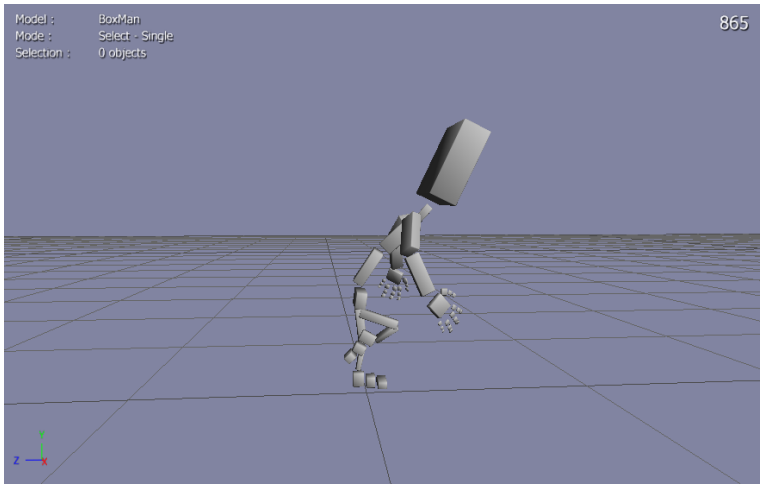
KeyFrame 30



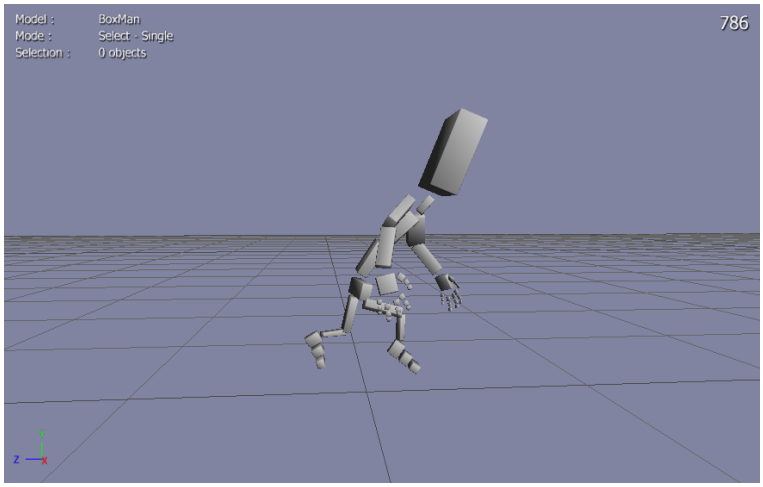
KeyFrame 40



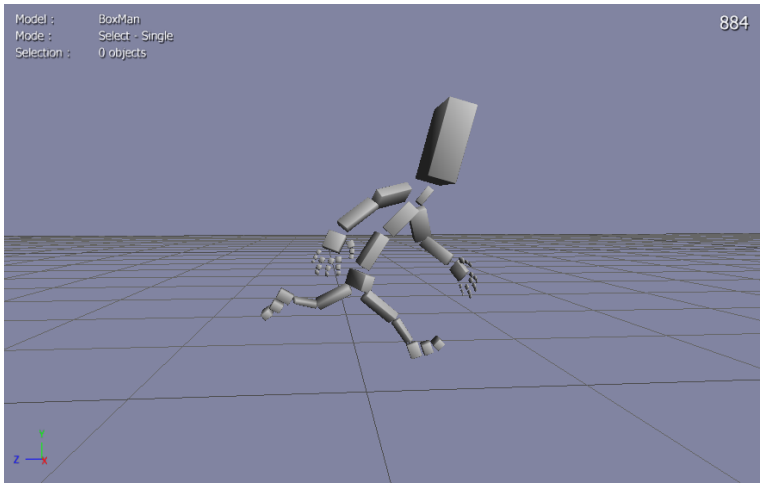
KeyFrame 50



KeyFrame 60



KeyFrame 70



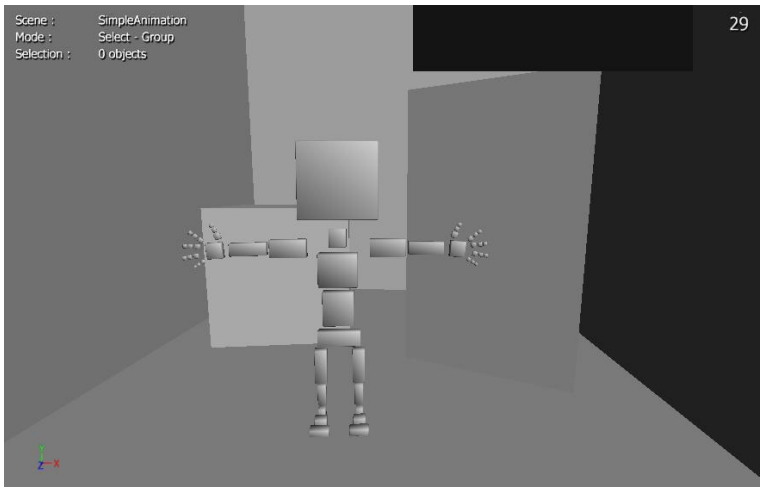
KeyFrame 80

Obviously, a lot happens between each KeyFrame, but I hope that the above screenshots give you some idea as to how the Animation changes.

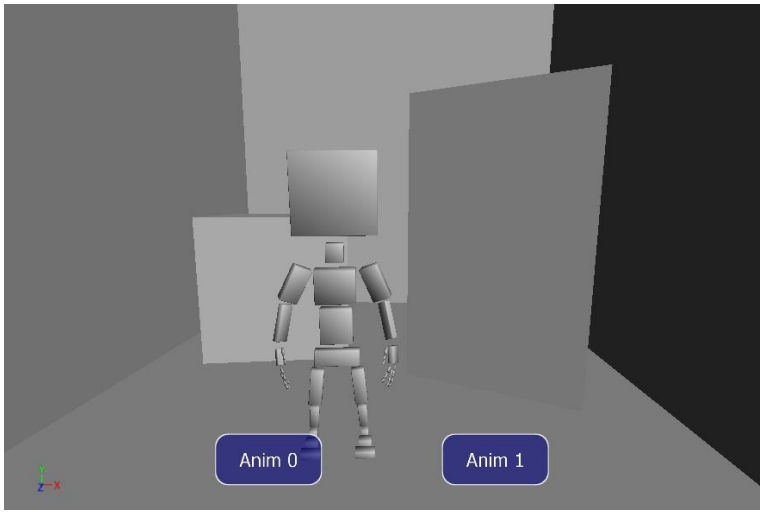
At this point, I would suggest that you go through each of the four AnimClips, and try to follow what is happening to each Channel. I know that it will take quite a bit of time, but it will be well worth it, as it will help you understand how an AnimClip (and Animation in general) works.

Well, that's pretty much all I can show you (at least in book form) about how AnimClips work! So, I'll move on to the next part of this demo, the HUD. If you have any further questions regarding AnimClips, or Animation in general, check out the Forum.

At this point, we should open up the Scene in the Scene Viewer. You should see something like this:



As you can tell, this is a standard ShiVa demo Scene, with the various boxes, the "cornell" boxes, and good old "BoxMan" standing in the middle. If you now click on the "Run" Button, you should see the following:



The only difference here is the two buttons, "Anim0", and "Anim1", which form the HUD for this demo.

To view the HUD, open up the HUD Editor, and select the "SimpleAnimation" HUD. You will now be presented with the following:

This is a pretty simple HUD, in that it consists of the two buttons mentioned above and two Actions, one for each button:

OnButtonAnim0

SendEventToUser (CurrentUser, "SimpleAnimation_Main", "onChangeAnim", 0)

OnButtonAnim1

SendEventToUser (CurrentUser, "SimpleAnimation_Main", "onChangeAnim", 1)

The only difference between the two Actions being the final parameter in each "SendEventToUser" call. Each Action calls the same Event "**onChangeAnim**" (in the "**SimpleAnimation_Main**" AIModel), with the final parameter determining which Animation to play.

If you now click on the "Anim 0" button, you will notice that nothing happens?? This is because Animation "0" is already playing (the "wait" Animation). If, however, you click on the "Anim 1" button, "BoxMan" will start to run towards you (though he runs on the spot, as this demo does not actually change "BoxMan's" position in the Scene).

Well, that's it for the HUD, so I'll now move on to the AIModels for this demo.

SimpleAnimation_Character

This AIModel consists of:

Variables

nCurFactor	-	Number	-	Initial Value – "0.000"
nDstFactor	-	Number	-	Initial Value – "0.000"

Handlers

onChangeAnim (nIndex)
onEnterFrame ( )
onInit ( )

onChangeAnim (nIndex)

A simple one-liner, which simply sets the value of the AIModel Variable "*nDstFactor*" to the value of the input parameter ("*nIndex*"):

```
this.nDstFactor ( nIndex )
```

onEnterFrame ( )

This one is slightly trickier, but most of it should be familiar to you by now:

```
local o = this.getObject ( )

if ( object.hasController ( o, object.kControllerTypeAnimation ) )
then
```

OK, we get a Handle to the current Object, and then check to see if it has an Animation Controller attached to it.

```
    local cur = this.nCurFactor ( )
    local dst = this.nDstFactor ( )
```

All that's happening here is that we are populating two LOCAL Variables with the values of the AIModel Variables.

```
    if ( cur ~= dst )
    then
```

Again, pretty straightforward. We are just checking to see if the value of "*cur*" is different to the value of "*dst*" (i.e. we haven't reached the end of the Animation).

```
        local dt = application.getLastFrameTime ( )
```

Assuming that "*cur*" and "*dst*" are different, we get the last Frame time in the standard way.

```
if ( dst > cur )
then
```

Next, we check to see if "dst" is greater than "cur" (i.e. we are part way through the Animation).

```
    cur = math.min ( cur + dt, 1 )
```

If it is, then we set "*cur*" to be the minimum of "cur + dt" and "1". This is done to ensure that "*cur*" contains the current position of the Animation in respect to time.

```
else
    cur = math.max ( cur – dt , 0 )
end
```

Otherwise, we set "*cur*" to be the maximum of "cur – dt" and "0". Again, to ensure that "cur" contains the correct position in respect to time.

Why do we find the minimum when "*cur*" is less than "*dst*", and the maximum when "*cur*" is greater than "*dst*"? This is done so that if "*cur*" becomes greater than "*dst*", then "*cur*" is set to a value of "1" (i.e. the end of the Animation). Bear in mind that the Animation (using "*cur*" and "*dst*") goes between "0" and "1", with "0" equating to the start of the Animation, and "1" equating to the end.

```
this.nCurFactor ( cur )
```

Next, we set the AIModel Variable "*nCurFactor*" to equal the value of "*cur*", so it can be used in the next Frame.

```
        animation.setPlaybackLevel ( o, 0, 1 – cur )
        animation.setPlaybackLevel ( o, 1, cur )
    end
end
```

Finally, we set the "PlaybackLevel" of the Animation as follows:

The first line sets the Blend Layer ("0") to equal the value of "1 – cur" (i.e. How much of the Animation is left).

The second line sets the other Blend Layer ("1") to equal the value of "*cur*" (i.e. How much of the Animation has already been completed).

These two lines ensure that the Animation is blended correctly and, as such, looks good!

onInit ( )

Another fairly complex looking Script but, in reality, it's not too hard!

```
local o = this.getObject ( )

if ( object.hasController ( o, object.kControllerTypeAnimation ) )
then
```

All standard stuff here.

```
    animation.changeClip ( o, 0, 0 )
```

This first line sets Blend Layer "0" for the Animation to be equal to the AnimClip at position "0" in the AnimBank.

```
    animation.setPlaybackMode ( o, 0, animation.kPlaybackModeLoopMirrored )
```

Now, we set the Playback Mode for Blend Layer "0" of the Animation to "LoopMirrored", which means that the Animation will be played in a Loop, and Mirrored.

```
    animation.setPlaybackLevel ( o, 0, 1 )
```

Next, we set the Playback Level for Blend Layer "0" to "1", which allows for smooth transitions between the Animations.

```
    animation.setPlaybackKeyFrameBegin ( o, 0, 0 )
    animation.setPlaybackKeyFrameEnd ( o, 0, 80 )
```

Finally, we set the Key Frames for the start and end of the Animation on Blend Layer "0" to "0", and "80" respectively (i.e. the first and last Key Frames of the Animation).

```
    animation.changeClip ( o, 1, 3 )
    animation.setPlaybackMode ( o, 1, animation.kPlaybackModeLoopMirrored )
    animation.setPlaybackLevel ( o, 1, 0 )
    animation.setPlaybackKeyFrameBegin ( o, 1, 0 )
    animation.setPlaybackKeyFrameEnd ( o, 1, 80 )
end
```

Now, we just repeat it all again for Blend Layer "1". The only differences being the first line, which sets the AnimClip at position "3" in the AnimBank, and the third line which sets the Playback Level to "0" (i.e. not playing).

That's all there is to it for the Character side of the Scripts! I'll quickly run through the "Main" AIModel, but there's really nothing that you haven't seen before:

Main

This AIModel consists of:

Handlers

onChangeAnim (nIndex)
onInit ( )

onChangeAnim (nIndex)

```
local s = application.getCurrentUserScene ( )

if ( s ~= nil )
then
    local o = scene.getTaggedObject ( s, "Character" )
```

All we are doing here is getting a Handle to the current Scene, and then a Handle to the Object in the Scene that has the "Character" Tag (i.e. "BoxMan")

```
    if ( o ~= nil )
    then
        object.sendEvent ( o, "SimpleAnimation_Character", "onChangeAnim", nIndex )
    end
end
```

Finally, all we do is send an Event to the "SimpleAnimation_Character" AIModel calling the "onChangeAnim" Handler, with the parameter of "*nIndex*".

onInit ( )

```
application.setCurrentUserScene ( "SimpleAnimation" )

hud.newTemplateInstance ( this.getUser ( ), "SimpleAnimation", "myHUD" )
```

These two lines simply set the current Scene to the "SimpleAnimation" Scene, and load up the HUD.

Well, that's about all for this Chapter. Hopefully, it's shown you how Animation can be used within ShiVa, especially with regards to blending Animations, and ensuring smooth transitions. In the next Chapter, I'll show you how to set up, and use, navigation Meshes.