

Tutorial „Cartoon eyes“

Written by Christoph Schinko

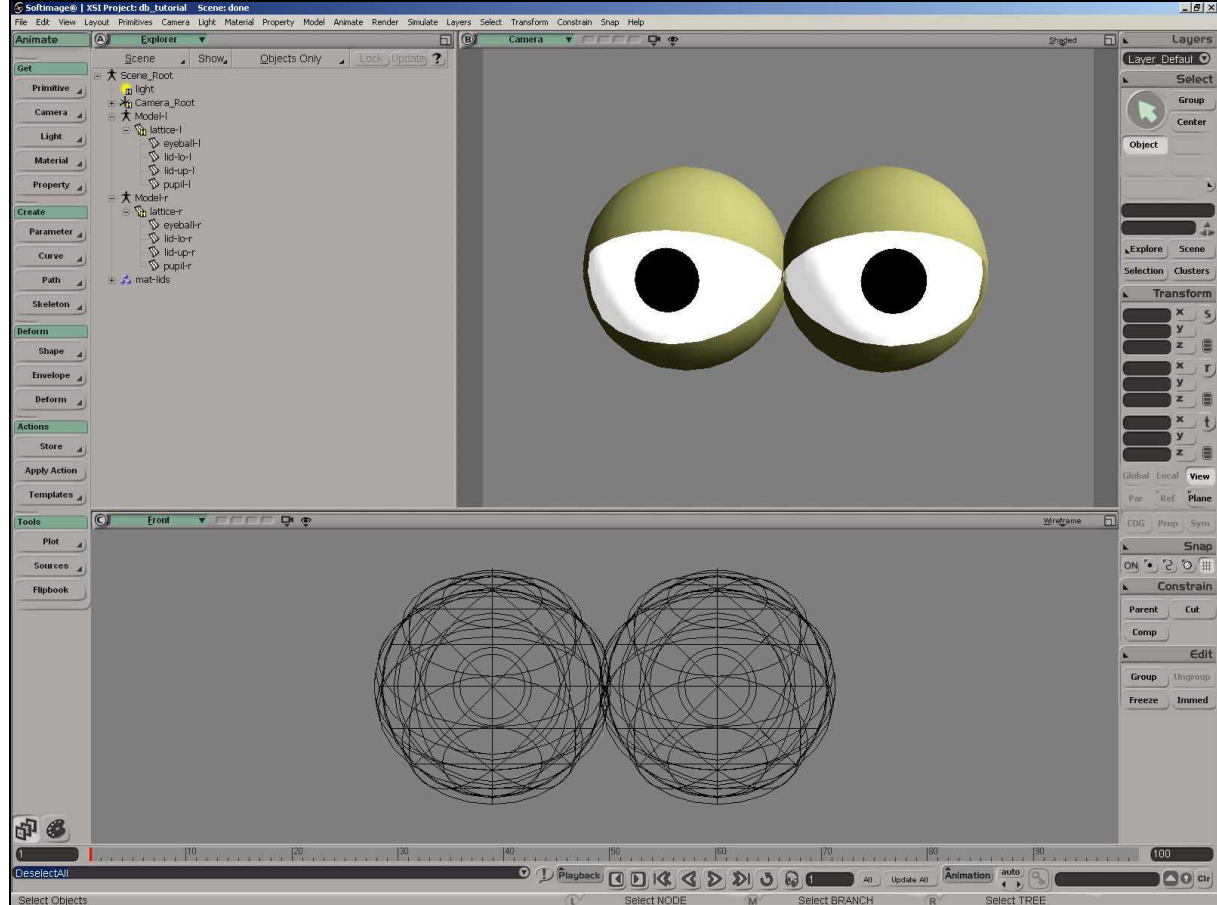
List of content:

- I. Creating the geometry
- II. Setting it up
- III. ‘fit clips to scene’ script
- IV. Creating the animation panel
- V. Summary

In this tutorial you will learn how to:

- create a pair of expressive eyes with very simple geometry
- set up lattices and how to animate them
- create and manage a library of eye expressions
- automate adjustments in the animation mixer via scripting
- rig your setup for easy animation

For the sake of simplicity our eyes will not have eyebrows or -lashes, they are not necessary for this exercise and may be added at your own will. This tutorial will NOT touch on XSI’stoon shader at all; it will simply deal with cartoony eyes.



Pic 1: overview

I. Creating the geometry

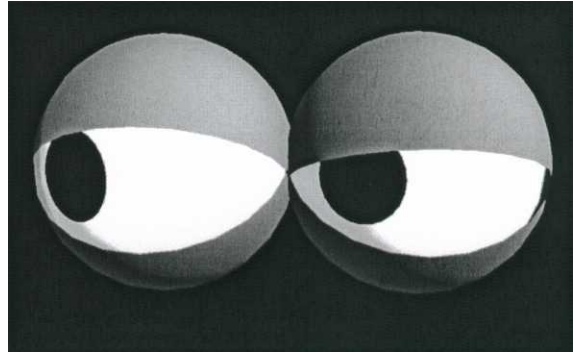
We're going to use simple NURBS-spheres to build up cartoon eyes. All we need for that is the eyeball, the pupil and eye lids. Very simple, so I didn't dare calling it 'modelling'.

1. **Get>Primitive>Surface>Sphere** (or when you hit Primitive, hit the shortcut „S“ twice) to create a Nurbs sphere and open its property editor. Name it „eyeball-l“, and leave the other settings at default.

In the **Extent** section, check out the **Start V** slider. Using it, we already have a control for dilating the pupil set up for us.

2. Rotate the sphere 90° in X, for the „eye“ to face us.
3. As we need to have a black pupil, create another sphere, and make it slightly smaller than the first, with a radius of about 3,9. Call it „pupil“ and give it a constant black material, with **Get>Material>Constant**. Remember that holding the **Ctrl-key** while dragging a colour slider, automatically changes all three values at once.
4. While we're at it, select the eyeball and give it a Phong material (**Get>Material>Phong**) with very white values (3 for colour, 4 for ambient). Here, by clicking on the RGB button below the colour box, XSI switches to the HLS mode, where in this case you only need to change the **luminosity (L)** value for our material.
5. Now we will create two half-spheres as the eye lids. Get another sphere, name it „lid-lo-l“, make it slightly bigger than default (~4.1), and set the **End V** slider to 100.
6. Same for the „lid-up-l“, just now it's the **Start V** slider that needs to be set to 80.
7. Give a material to the eye lids, if you like. For sharing a material between both lids, select them, and hit **Group** in the **MCP** (main command panel). With the group selected, get a material of your choice. This is useful so you only have to change one material for affecting both objects.

Now we already have quite some functionality here to control our eyes. Rotating the lids in X opens and closes them, and the Z rotation of the upper lid even adds some expression to the eye. The eye can look in any direction, using the X and Y rotation. And in the eyeball's ppg (property page), the size of the pupil can be changed with the Start V slider. If we want to make the eye look straight again, we have to set back its rotation values to 90,0,0. This can be confusing, so let's freeze the rotations on this object.



Pic 2: the eye geometry

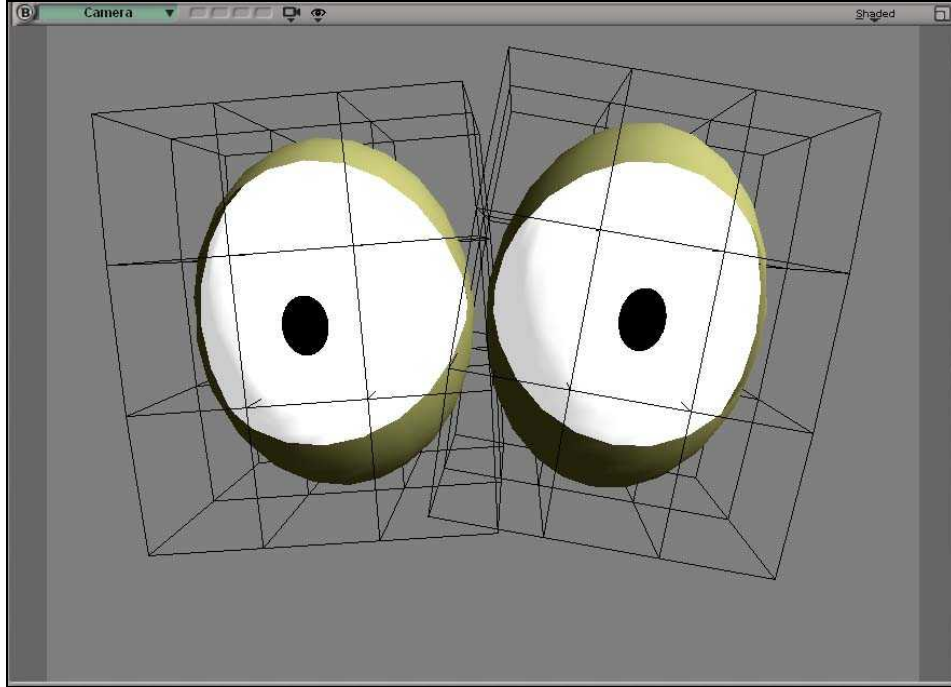
8. Select the eyeball and set it back to 90,0,0. With rotations still active, choose **Transform>Freeze Active Transform** above the translations. Now our default pose is at 0,0,0.

Now we'll make our eye a separate **model**. A model is like a scene within a scene that means that you can even have exactly the same name for two or more objects. Also, each model gets its own animation mixer, which makes it easier to handle and organize actions and such. There's way more cool things that you can do with models, but we don't need to know all that right now.

9. Select all your eye objects and hit **Create>Model>New** in the modelling toolbar. Just close the dialogue, or rename your model to „eye-1“ or so. Branch-select the model and translate it to x=4.
10. With the model still selected hit **Ctrl-D** or **Edit>Duplicate Single** to create the second eye. Move the new model to x=-4 and rename it (which also works with right-clicking in an explorer and selecting „rename“).

It is time to set our eyes up with lattice boxes and add some expression to them. We'll just touch on the basic shapes like anger, sadness, tired and some more. Those shapes will be saved out and brought into the animation mixer, where we'll be able to animate them beautifully.

11. Select all objects of one eye, (here this is a multi-selection, but you can of course select branches or trees) and choose **Get>Primitive>Lattice**. All selected objects will be deformed by that lattice, which ppg opens now up. Define a cage with 3,3,2 Subdivisions and set the interpolation to „curve“ for all axis.
12. Drag and drop the lattice over the model of the according eye for parenting. Now it will use the mixer of this model.
13. Do the same thing for the other eye. Lattice it and put it in a hierarchy suiting for you.

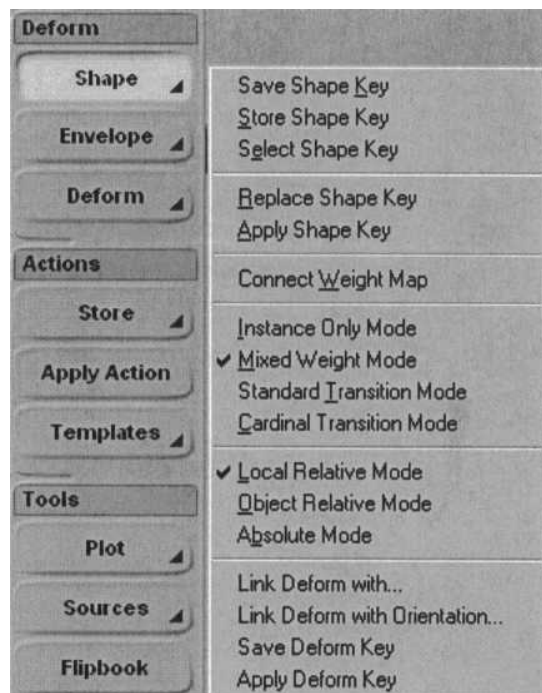


Pic 3: scared?

II. Setting it up

From here on, we'll mostly work in the animating module, where we'll save the shapes for the lattices and set up custom property pages for better controlling of the animation.

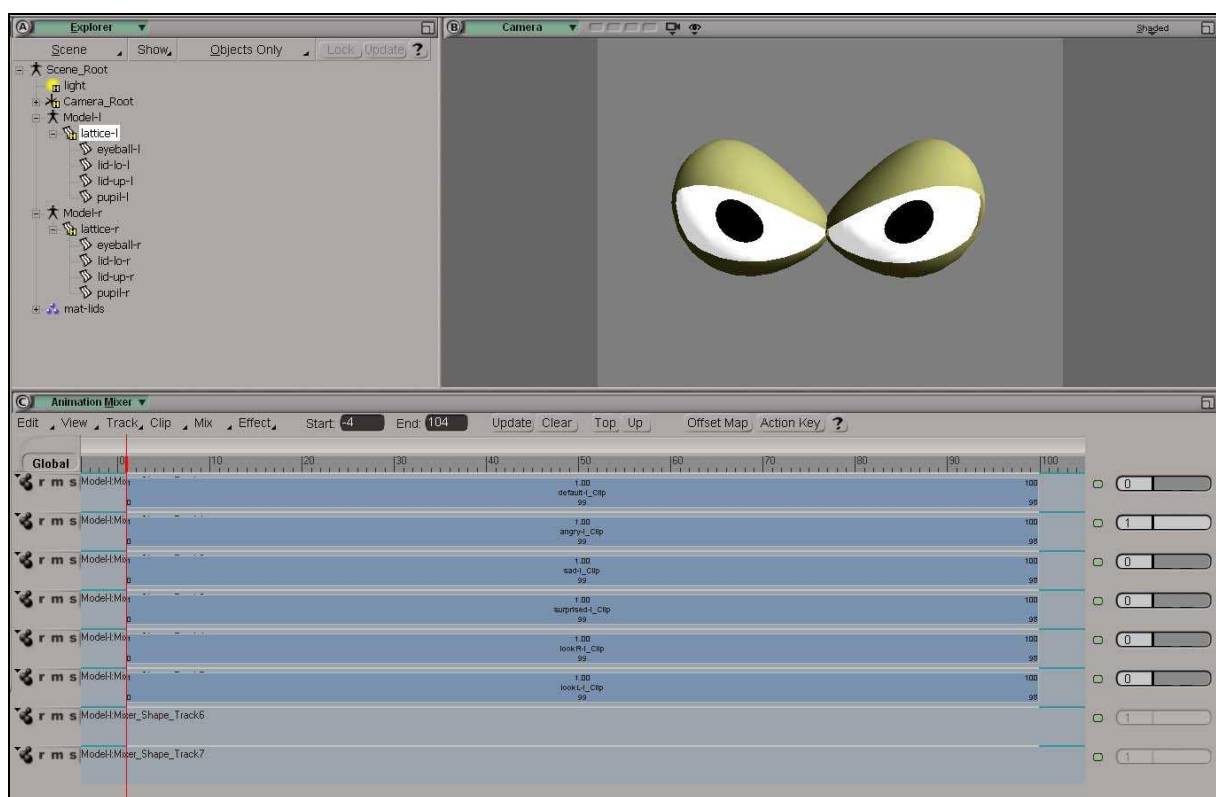
14. Make sure that **mixed weight mode** is active under **Deform>Shape**.
15. Selecting both lattice boxes, we'll keyframe the default shape for them. In the **animation module**, go **Deform>Shape>Store Shape Key**. This will create a shape source for each object, and as it is the first keyframe operation under these models, an animation mixer is created for each model. However, no clip will be put into the mixer. A ppg with the name of the shape clip opens and you are welcome to rename it.



Pic 4: shape options

In a viewport, open an animation mixer and an explorer in another. Let's check out what happened here. Select one of the lattices and hit **Update** in the animation mixer. You will see that there are two green, empty tracks. Green tracks are for animation clips, and there's blue for shape tracks and a sort of orange for audio tracks. We will create shape tracks in a second. Up in the explorer, select **Mixers only** from the selection tab. You'll see that there's a mixer under both of our models, and there would be one under the scene_root if we animated an object in its own space. Expand mixer to source and shape, where our first saved shape action is.

16. With the lattice boxes still selected, start to modify them to different shapes for eye expression. Make an angry, a sad and a surprised shape plus a look to the left and another to the right. And each time you are satisfied with one, save it as a shape source with **Deform>Shape>Store Shape Key**. Model them as you like, maybe make some sketches first and put in some asymmetry as well.



Pic 5: shape tracks in the animation mixer

17. For each of our eyes, select the lattice and hit update in the mixer again. Create shape tracks for our clips with right click on any track and choosing **Add Track>Shape** or hit Shift+S. Start dragging shape sources from the explorer into the shape tracks.

Don't worry about the length and placement of the clips yet. As we want to control the shapes with custom sliders, we will access them via the weight curve of the clip. And as they of course only work where there is a clip placed, and we probably will have scenes of different lengths to animate, we will automate our clip sizes for them to always fit to the actual scene length. We will do that with a small script that is hidden behind a button.

III. Fit clips to scene

18. Drag & drop all shape clips of both eyes onto the mixers, each clip on its separate track. Open the script editor by clicking on the „!“ . On the grey top half you can see all the operations you did logged. Below you can enter scripts and run them. First thing we want is to select all our clips, so that we don't have to place them all separately. Do **Edit>Clear History Log** to make things easier.



Pic 6: sexy script icon

19. Select a lattice and hit update in the animation mixer. Select all clips holding the Shift key. Each time another clip is selected, the command line is listed in the history log. It should look like this:

```
AddToSelection "Model-r.Mixer.Mixer_Shape_Track.lookL-r_Clip"  
AddToSelection "Model-r.Mixer.Mixer_Shape_Track1.lookR-r_Clip"  
AddToSelection "Model-r.Mixer.Mixer_Shape_Track2.surprised-r_Clip"
```

20. Copy/Paste all those „AddToSelection“ lines from the history log to the script editor. Select the second lattice, update the mixer again and also select all those clips. And add the logged results to your script.
21. We can see how the syntax of our shape clips is very similar to one another. XSI allows us to use wildcards, so we can actually replace the clip name with a ‘*’, and all existing shape clips will be selected. This is a very practical method, however, be aware that scenes may contain more than only the shape clips for the eyes! You can just copy/paste the commands for selecting each clip into your script, or come up with another way for only selecting our eye shape clips. For this exercise, we'll be fine with the dangerous way. This single line is enough to select all existing shape clips. The first ‘*’ stands for any model, the second one for any track and the third for any clip name.

```
SelectObj "*" .Mixer.Mixer_Shape_Track*.*"
```

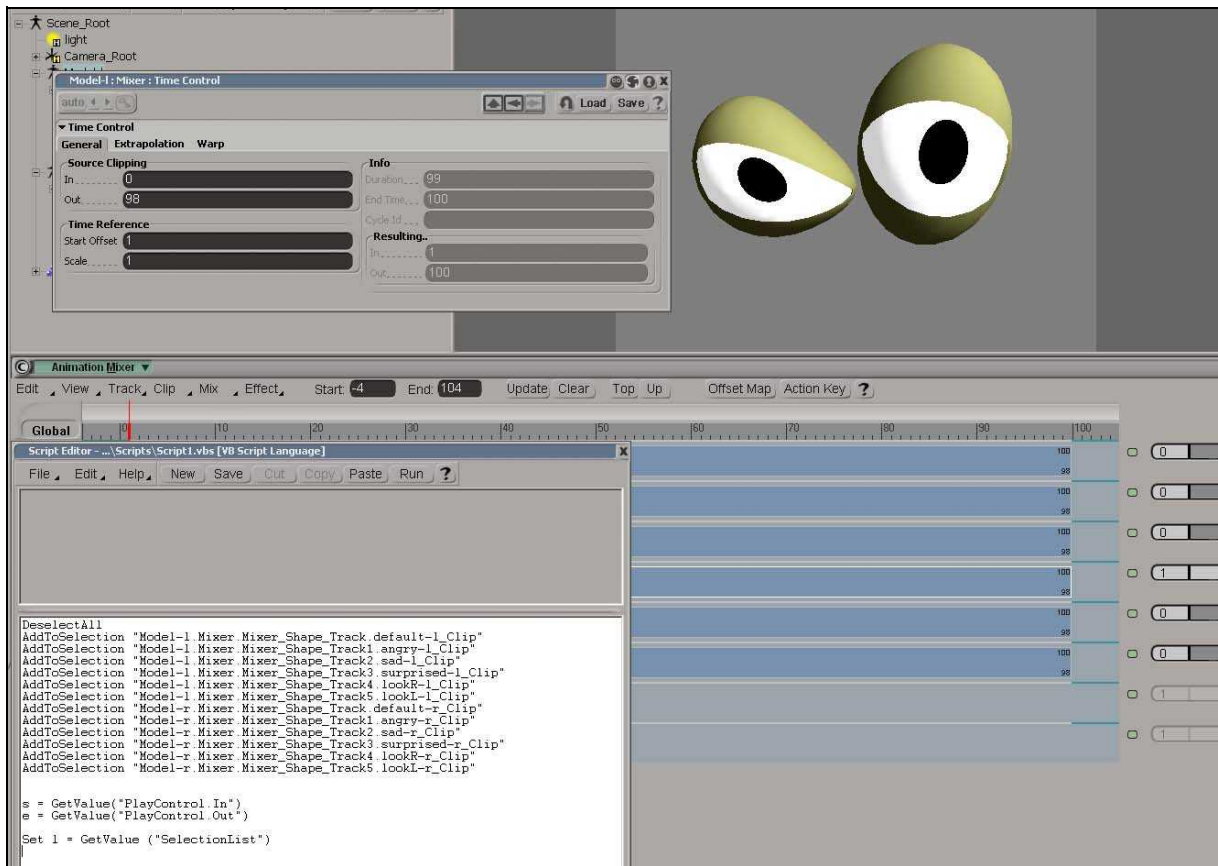
Should you choose to stick with the safer method, make sure that you have a `DeselectAll` in the first line of your script, or change the first command from `SelectObj` to `AddToSelection` to ensure that later commands are not executed on previously selected objects.

22. Next we need to do is find out how long our scene is. Changing the start and end frames of the scene, all information needed is displayed in the history log. Copy/Paste them into the script editor to use them. `"PlayControl.In"` and `"PlayControl.Out"`. With the `GetValue` function, we will extract their information, the frame numbers, to variables that we define. „S“ for start and „E“ for end (do not use „end“ as a variable, as it actually is a VBScript function and would end a routine).

```
s = GetValue("PlayControl.In")  
e = GetValue("PlayControl.Out")
```

23. The last variable we need to work with is the selection that we made in the top part of the script. The selected objects are stored in a variable called „SelectionList“, and with setting a variable of our own (l for „list“) to its value we can work with it better.

```
Set l = GetValue ("SelectionList")
```



Pic 7: the script editor

24. To see how we will be able to access the clip size via script, right click on a clip and choose **Time Properties** (or hit Ctrl+T). Here, the position and scaling of the clip and more are kept track of. Use the Start Offset for setting where the clip starts, again out of the history log. When the value is changed, something like this is logged (in one line):

```
SetValue "Model-r.Mixer.Mixer_Shape_Track.default-r_Clip.actionclip.timectrl.startoffset", 1.000
```

25. The part that we need from this command is the addressing of the start offset variable that we seek to set – „...actionclip.timectrl.startoffset“. Everything before that is the name of our clip. As the clipname will be provided from our l (list) variable, we have to tell XSI to use that instead of the one selected clip. To do that use **ELEM**, and connect it to the start offset string with a „&“.

```
SetValue elem&".actionclip.timectrl.startoffset", 19.000
```

26. To set it to the first frame of our scene, use the „s“ variable that we defined before. Simply replace it with the 19.000.

```
SetValue elem&".actionclip.timectrl.startoffset", s
```

27. The same thing is to be done to the Out value of the clipping. This is probably not the best way to do it, but it worked for me the easiest. Subtract the start variable from the end variable -1, then the clip will end with the end of the scene length.

```
SetValue elem&".actionclip.timectrl.clipout", e-s-1
```

28. As we have to do those two operations a bunch of times, we will use a loop to run through our selection list. For each element in our selection list „l“, do the two operations and go to the next element until you're done.

```
for each elem in l
SetValue elem&".actionclip.timectrl.startoffset", s
SetValue elem&".actionclip.timectrl.clipout", e-s-1
next
```

The whole script now looks like this, and hurray it works!

```
SelectObj ``*.Mixer.Mixer_Shape_Track*.*``

s = GetValue("PlayControl.In")
e = GetValue("PlayControl.Out")

Set l = GetValue ("SelectionList")

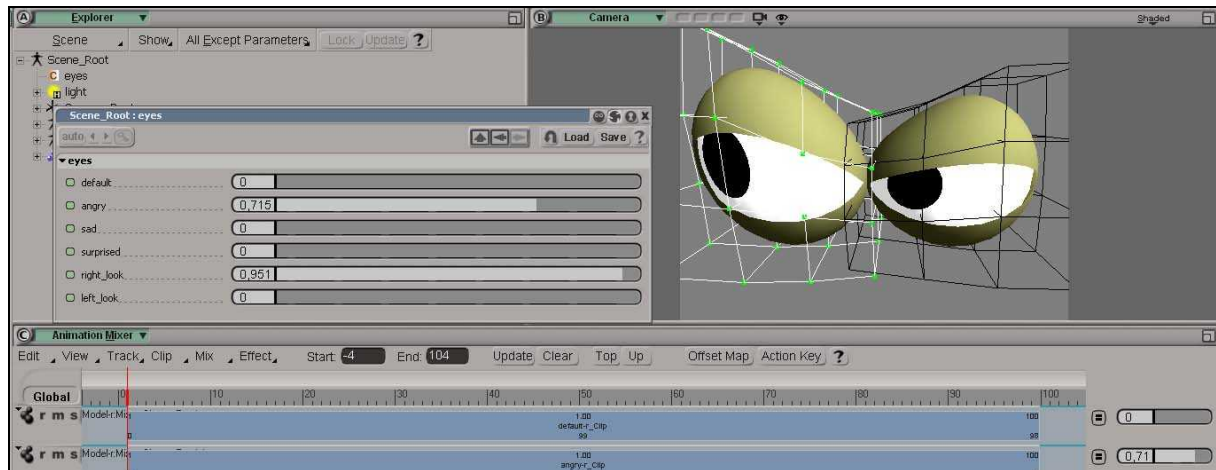
for each elem in l
SetValue elem&".actionclip.timectrl.startoffset", s
SetValue elem&".actionclip.timectrl.clipout", e-s-1
next
```

29. The last touch will be to put our script into a button to be able to access it more quickly. Click on the palette icon on the lower left side of the screen. Select the whole script in the script editor and just drag & drop it onto the grey area of the toolbar. A dialogue box pops up and lets you enter a name for your button, something like „fit to scene“. Leave the rest and hit **OK**. The button is placed on the toolbar and the script now runs under „Script1“ and is not showing every single command that we entered. Change the start- and end frame of the scene and see how fast it works.

IV. Creating the animation panel

It is time to put all the stuff that we will want to animate into a separate window with nice little sliders.

30. With nothing selected hit **Create>Parameter>New Custom Parameter Set** and name it in the upcoming dialogue. Open an explorer and make sure that „**All Except Parameters**“ is active as display filter. There is an orange icon with a „C“ in it right under the scene root. That is our new parameter set. Click on the icon and it opens up, still empty.
31. Hit **Create>Parameter>New Custom Parameter** to define the first slider. The name is very important here, as that is what will show up before the slider. All the settings are fine for us, we want the slider to go from 0 to 1 and have it animatable. UI Range is a very neat feature you should play with, it practically allows overshooting values in your own sliders. Create one custom parameter per shape, default shape included.



Pic 8: the slider panel

32. Let's link the weight curves of all our shape clips to the custom sliders. In the animation mixer, right click on the green key icon and choose **Set Expression**. An expression editor opens up, and the weight curve is already set as the affected element. Select the value that the expression is set on, 0 or 1. Here you will put in the slider name that you want that specific clip connected to. Hit **Objects** in the menu bar to open a transient explorer and click on eyes-sad, for example. The expression is entered in the editor, and with hitting **Apply** it is set. Do this for every shape clip for both eyes.

Done! Play with the sliders to see how they work. If you want, link the eye lid rotations or the size of the pupil to sliders aswell, until you can animate everything you desire from your own control window.

This tutorial was written by Christoph Schinko in July 2001, he is 3D artist and Softimage certified XSI instructor.