Maxscript Workshop 1

Full tutorial video support for Maxscript is available on BlackBoard (B8).

The “Listener” window is very helpful as it shows what Maxscript commands are currently being executed when you use the GUI. This is a good way to help figure out how to automate a process that you are doing by hand. There are exceptions though where it might not show what you need. Fortunately there is copious documentation in the “Maxscript Help” resource. Additionally there are some capabilities you might discover whilst perusing this documentation are only available through Maxscript and that are not accessible via the GUI.

Task 1 - Teapot Spiral (Warmup)

Try running the following code. You can either type it into a “New Script” and “Evaluate” it, or enter into the white window in the “Listener” window. If you choose the latter approach, to run multiple lines of script, select the lines and tap “Enter” key on the number pad.

```
resetMaxFile()
for k in 0.0 to 360.0 by 10.0 do (
   teapot position:[k,0,0] scale:[k/100,k/100,k/100] \
rotation:(angleAxis k [0,0,1])
)
```

Try tampering with it so you understand how it works.
  e.g. Create a script to construct and place 10 cubes on a circular path.

Task 2 - Planetary System

Create three spheres (e.g. Sun, Earth and Moon) then ctrl+click each each so all are selected. The order you click them in will be the order of they feature in the ObjectSet $[1..n]. Note, if only one items is selected then $ just represents that item (not an ObjectSet containing only a single item as one might expect).

The currently selected object, or selection of objects, is given by $
e.g.

```
if (classof $ == ObjectSet) then (
   messagebox ("There are "+($.count as string)+" objects selected")
) else (
   messagebox ("Just one object selected:\n"+($ as string))
)
```

Use the following script to animate your planetary system.

```
if (classof $ != ObjectSet) then (
   messagebox("Need at least two selected objects.")
) else (
   for obj_id in 2 to $.count do (
      $[obj_id].parent = $[obj_id-1]
   )
   animate on (
      for obj in $ do (
         at time animationRange.start (
            rotate obj (angleAxis 0 [0,0,1])
         )
         at time animationRange.end (
            rotate obj (angleAxis 360 [0,0,1])
         )
      )
   )
)
```

And be sure you are clear about how this script is working.

Task 3 - Autoworm

In the lecture I showed you how to skin and rig a worm in 3DSMax (like you did last week in Blender). Start by understanding this workflow (i.e. quickly do it yourself):

```
    1 - Make a Geometry->Cylinder
    2 - Make some appropriate Systems->Bones
    3 - Apply the Modifier->Skin to the cylinder
    4 - Add the bones to the modifier
    5 - Create an animation by keyframing the bones
```

Your task this week is to entirely automate the above process in a single Maxscript.

The first line of the script should be:

```
resetMaxFile(#noPrompt)
```

This will clear 3DSMax of any existing content. From here your script should complete the above steps to create and rig a cylinder and make animation. To give you some clues as to how to achieve this, here are some tips:

```
B1 = BoneSys.createBone [0,0,0] [10,0,0] [0,0,1]
```

Whilst you can create bones manually i.e. bone() these are real just geometric object placeholders. So that they have the properties we want (i.e. a fixed length and known start and end points, the `BoneSys` functions are very helpful. Note, you will still need to setup the hierarchy (e.g. `B2.parent = B1`).

```
mySkin = skin()
addmodifier myCylinder mySkin
```

Modifiers can be created just like any other object. The above commands create a new skin modifier object (mySkin) and attach it to a geometry object (myCylinder). However, the options within a modifiers are sometimes only available when the modifier's associated window is active. To make this happen you need to:

```
max modify mode
modPanel.setCurrentObject mySkin
```

You should then be able to do things like add a bone to the skin. e.g.

```
skinOps.addBone mySkin B1 1
```

Finally, to make keyframes automatically, you can use "`animate on`" (as in the planetary system example) and your script should **rotate** bones rather than trying to **move** them (when you manually drag a bone you are actually rotating it about its joint or "**pivot point**").

```
at time 10 rotate B2 (angleaxis -45 [0,1,0])
```