

MEL scripting

One of the great things about Maya is that, although it might not do everything you'd like it to straight out of the box, it provides a way for you the user to add your own functionality via MEL (Maya Embedded Language).

Disclaimer: there is much too much here for a 2-pager. Refer to Maya's built-in documentation for more info

There are a few main ways in which one uses MEL:

- running scripts created by other Maya users (check www.highend3d.com)
- using the script editor to record actions
- writing expressions
- writing your own external scripts

While you can make extremely complex scripts that add major functionality, you can also get a great deal of benefit (increases speed, efficiency, and ease) from relatively simple scripts.

Using the Script Editor

The script editor (Windows->General Editors->Script Editor) serves as a two-way communication tool between you and Maya. The top section is where Maya talks to you- look there for feedback. The bottom section is where you can give Maya MEL commands to execute. You can use the script editor to create MEL scripts for you, much like the action recorder in Photoshop. You can have it "echo" the MEL commands that correspond to actions you take in the scene by using the Script->Echo All Commands command. Now, anytime you do anything in Maya, the commands for that action will show up in the top half of the script editor.

You can highlight text and middle-mouse drag it to a shelf to create a button that will execute the code.

Note: Echoing commands is great, but can be misleading, as it often displays extra commands. If in doubt, check the MEL reference (Help->MEL command reference)

Running scripts from external files

Whether you download scripts or write your own, you'll generally be running them from external files (as opposed to from the script editor). To run an external script, copy the script file (*myScript.mel*) into your *scripts* directory (usually My Documents->maya->4.5->scripts). Having done that, you can type the name of the script in either the command line or the script editor.

If you've downloading scripts, that's all there is to it. If you want to write you own, though....

Writing scripts

While you can write scripts as long as you like in the script editor, you'll generally want to use an external editor, as code disappears from the script editor once it executes. So, the workflow looks like this:

- write a script in an external editor (Notepad or similar)
- save it out to your script directory as *Something.mel* Make sure to save it as plain text
- execute it in Maya to test it by typing its name at the command line, and, most likely...
- go back to your text editor to make changes

Note that for Maya to see any changes to a script, you have to *source* the script. This will make Maya re-load it into memory, and will reflect any changes you've made to what the script does. To source a script, use the File->Source script command from within the script editor. Do that every time you make changes to your script. Also, before running a script for the first time, you must use the rehash command to force Maya to re-examine its script directories and re-build the list of known scripts.

Structure of scripts

All scripts must contain at least one "global procedure" in order to be run. What that means is that you need to start a script with the following:

```
global proc myProcedure (any variables) {
```

and end it with a closing `}`. "*myProcedure*" is the name of the procedure, and what you type to get Maya to run it. Scripts are made up of a few kinds of elements, mainly variables, commands, control structure, and functions

- | | |
|---------------------------|--|
| Variables | are bits of information, and can be one of several different kinds (integer, float, string, array, etc). |
| Commands | are statements that either request information from Maya, or tell Maya to do something |
| Control Structures | are specialized commands that affect the order in which statements are executed |
| Procedures | are blocks of code, containing statements and variables that accomplish some task |

Custom GUIs

MEL not only allows you to make your own tools, but it also allows you to wrap them up into a custom GUI (Graphical User Interface) to make them easier to use. Making your own GUI consists of the following steps

- creating a window
- selecting a layout, or series of layouts
- creating controls and attaching them to the window
- attaching MEL commands to the controls
- showing the window

The last step is a bit counter-intuitive, but necessary. Though MEL has a window command to create a window, it doesn't display it until you tell it to. Creating a window looks like this....

```
window -title "myWindowTitle" windowName;
```

this specifies two things- the text that will appear in the top of the window (myTitle, in quotes), and the name by which Maya will know the window (windowName). Notice that the line ends with a semicolon (;). All MEL commands must end with a semicolon.

Once you've got a window, you need to specify a layout, which will control how control items are placed within the window. There are several different layout types available, here's an example.

```
rowColumnLayout -numberOfColumns 2;
```

...which will create a 2-wide grid (with as many rows as necessary) for control items to fit into. Now you're ready to add control items. There are a variety of different kinds- let's start with a button;

```
button -c "$temp = `polySphere`" -label "Make Sphere";
```

Will make a button that creates a polygon sphere. The -c tag specifies the command that is executed, and the -label tag gives the button its title. Notice that the command it executes when pressed is \$temp[0] = `polySphere`. The `polySphere` part creates a sphere, and since its enclosed in back-quotes (`), it *returns* the name of the sphere. The sphere's name is then set to a variable named temp (all variable names begin with \$). By assigning the sphere's name to a variable, we have a way of grabbing it later.

Now let's add some sliders to control the positioning of the sphere.

```
floatSliderGrp -label "Y position" -min -10 -max 10 -value 0 -step 1 posY
```

...will create a slider that takes float values and ranges from -10 to 10, starting at 0, and names it "posY". To get this to control something, we can use the connectControl command...

```
connectControl posY ($temp[0]+".ty");
```

This will connect the slider to the sphere's translate Y attribute. \$temp is the variable we created to hold the sphere we made. In order to attach the slider to the translate Y attribute, we have to tack on a ".ty" to the object name. Now, moving the slider will move the sphere. Okay, so now we've got a button that creates a sphere and a slider that lets us move the sphere's position along the Y axis. Let's finish up by showing the window, using the showWindow command...

```
showWindow windowName;
```

This will make the window actually appear in Maya, et voila!

This just scratches the surface of what MEL is capable of. To really explore the possibilities, be sure to look at the MEL pages in Maya's help documentation, as well as the MEL command reference (Help->MEL command reference) to see what the available commands are.