Introduction to the Maya API

by Mike Taylor

The notes for this section will be brief as the majority of this presentation is well covered in the introduction to the Maya API Developer's Manual.

Developing with Maya

There are two ways to extend Maya's functionality. The first is Maya's C++ API. This is the suggested way to write complicated extensions or ones that require a higher level of performance. For example, the C++ API is suitable for writing dependency nodes, tools, and shaders.

The second tool for extending Maya is MEL (Maya Embedded Language). MEL is a suitable for simple tasks and the creation of user interfaces. All of the user interface layout in Maya is done in MEL.

Writing a complicated extension to Maya typically requires both the API and MEL. The important thing is to try and find the correct balance between C++ and MEL. Developing with MEL is much faster if the task is simple enough to warrant it.

Two Core Architectures of Maya

When developing with Maya's API, there are two main units of functionality that can be added. These are dependency graph nodes and MEL commands. There are other ways to extend Maya via the API, but these two are the fundamental ones.

MEL Commands

Since the entire user interface of Maya is written in MEL, an excellent way to add a new feature is by writing a new MEL command. Once the new functionality is enclosed in a new MEL command, it is trivial to add a user interface via MEL using your new command.

Dependency Graph Nodes

The other core building block of Maya is the dependency graph. All objects in a scene and their data flow connections are represented by nodes and connections in the dependency graph. Nodes use data from their inputs to compute a suitable output. Dependency graph nodes come in different flavors. The types of nodes that can be written include the following:

- Geometry deformers
- Manipulators
- Locators (simple DAG objects)
- Dynamics fields
- Dynamics emitters
- Shaders (including hardware shaders)
- Shapes

The details of the dependency graph are covered later in the course notes.

Other Plug-ins

The API extends beyond nodes and commands. The following are a few other things that can be written in the API:

- Tools
- File translators
- IK solvers

Classes in the API

The following are the types of classes available in the API:

- Maya objects (MObject)
- Function sets (MFn…)
- Proxy objects (MPx…)
- Wrappers (M…)
- Iterators (MIt…)

The details of what these are and how they are used may be found in the first chapter of the Maya API Developer's Manual.

Top Ten Important Classes in the API

The following are ten of the most important classes in the API:

1. MObject
2. MPxCommand
3. MPxNode (and derived classes)
4. MGlobal
5. MDagPath
6. MItMesh*/MItSurface*
7. MSelectionList
8. MItDag/MItDependencyGraph
9. MPlug
10. MDGModifier

What follows is a brief description of each. Many are described in more detail later in these notes. If any of these classes seem unfamiliar, then consider taking some time to investigate.

MObject

An MObject is very much like a void star pointer. It is a handle to an object inside of Maya. It is up to Maya to create and destroy these objects. A good rule of thumb is to only use an MObject directly after it is received. One should not hold onto an MObject between calls to your plug-in.

MPxCommand

This is the base class to derive off of when writing a new MEL command. See the API class reference for more details.

MPxNode

This is the base class to derive off of when writing a new dependency graph node. One may derive off of MPxNode or any one of its subclasses. See the API class reference for more details.

MGlobal

MGlobal contains a general collection of high level routines. Among its contents are routines for setting and querying the current selection, routines for running MEL commands, and calls to control the error logging facilities.

MDagPath

An MDagPath is a reference to an object in the DAG. An MDagPath can be used instead of an MObject in most places where a DAG object is expected. Unlike an MObject, an MDagPath is a safe way to store a reference to a DAG object between calls to your plug-in. MDagPaths are important when objects are instanced because an MDagPath can point to a particular instance. Anyone doing much work with the DAG should use MDagPaths instead of MObjects in most cases.

MItMeshEdge, MItMeshPolygon, MItMeshVertex, MItSurfaceCV

These classes allow one to iterate over the components of an object and act upon those components. These iterators may be used on an entire object or a portion of the object that has been selected by the user.

MSelectionList

MSelectionList serves to store a list of objects in the scene. The list may contain whole objects or components of objects. This is a safe way to store a list of objects between calls to your plug-in.

MItDag/MItDependencyGraph

MItDag allows one to iterate over nodes in the DAG. MItDependencyGraph allows one to traverse the dependency graph from an arbitrary starting point. It is very complicated to traverse the graph without MItDependencyGraph.

MPlug

This class is critical when dealing with values or connections in the dependency graph. A plug is a connection point on a dependency node where information can flow into or out of the node. This class provides access to it.

MDGModifier/MDagModifier

MDGModifier is a utility class that is not absolutely necessary to get your work done. But, it is very useful and will make tasks considerably easier. It contains utility routines for creating nodes, deleting nodes, making connections, executing commands, and adding attributes to nodes. The best part of it is that this class implements undo and redo so that the programmer does not have to. MDagModifier is a sub class that also has routines for performing DAG operations.

There are lots of classes in the API and which ones you use will depend on what you are doing. But the ten listed above are generally important. Learning them is highly recommended!