

MAYA API – Lesson2 – The plug-in Structure

Here is the main structure for most scripted API plug-ins.

- **Top level overview of plugin structure:**
- Plugin class/classes implements core functionality.
 - Derives from an MPx class. Each type of plugin is implemented as python class that will be derived from.
 - Implements functionality by overriding interface functions. Maya gives us a template, and we override that template based on our needs.
 - Will contain any instance data, helper funcs, etc.
- Creator Function:
 - Returns a new instance of the class
 - Needs to be one per plugin class being authored.
- 'doIt' function:
 - What actually does the work in the instance of the plugin object.
- Registration/Deregistration:
 - If multiple plugins classes are being authored in the same Python module, they can share the same regis\deregis block
 - Tells Maya: about new plugins, their types, hooks needed to instance them.

Here is an example:

```
#===== Code Start =====
```

```
# License Information  
# Description  
# Usage
```

```
#####
```

```
import sys  
#~ import maya.cmds as mc  
import maya.OpenMaya as OpenMaya  
import maya.OpenMayaMPx as OpenMayaMPx  
#~ import maya.OpenMayaRender as OpenMayaRender  
#~ import maya.OpenMayaUI as OpenMayaUI
```

```
#####
```

```
kPluginNodeTypename = "My_Plugin_Name"  
kPluginNodeId = OpenMaya.MTypeId(0x8700B)  
#~ kPluginCmdName = "CMD_Name"  
#~ kPluginNodeClassify = "utility/color"  
#~ glRenderer = OpenMayaRender.MHardwareRenderer.theRenderer()  
#~ glFT = glRenderer.glFunctionTable()  
#~ print "Something here"
```

```

##### Node #####
# Node
class MyNode(OpenMayaMPx.MPxNode):
    # class variables
    input = OpenMaya.MObject()
    output = OpenMaya.MObject()
    def __init__(self):
        OpenMayaMPx.MPxNode.__init__(self)

# Node Creator
def nodeCreator():
    return OpenMayaMPx.asMPxPtr( animCube() )
##### Command #####
# Command
class scriptedCommand(OpenMayaMPx.MPxCommand):
    def __init__(self):
        print "scriptedCommand.__init__()"
        OpenMayaMPx.MPxCommand.__init__(self)

# Command Creator
def cmdCreator():
    return OpenMayaMPx.asMPxPtr( scriptedCommand() )
# Syntax creator
#~ def syntaxCreator():
    #~ syntax = OpenMaya.MSyntax()
    #~ syntax.addFlag(kPitchFlag, kPitchLongFlag, OpenMaya.MSyntax.kDouble)
    #~ syntax.addFlag(kRadiusFlag, kRadiusLongFlag, OpenMaya.MSyntax.kDouble)
    #~ return syntax
##### Initializer #####
# initializer
def nodeInitializer():
    # nothing to initialize
    pass

#####
# initialize the script plug-in
def initializePlugin(mobject):
    mplugin = OpenMayaMPx.MFnPlugin(mobject, "Autodesk", "1.0", "Any")
    try:
        mplugin.registerNode( kPluginNodeTypeName, kPluginNodeId, nodeCreator,
nodeInitializer )
        #~ mplugin.registerCommand( kPluginCmdName, cmdCreator )
    except:
        sys.stderr.write( "Failed to register node: %s" % kPluginNodeTypeName )
        raise
# uninitialize the script plug-in
def uninitializePlugin(mobject):
    mplugin = OpenMayaMPx.MFnPlugin(mobject)
    try:
        mplugin.deregisterNode( kPluginNodeId )

```

except:

```
sys.stderr.write( "Failed to deregister node: %s" % kPluginNodeTypeName )  
raise
```

#==== Code End =====

--

Sincerely,

©Copyright Farsheed Ashouri,

rodmena.com,

Lead Technical Director / Research & Development,

Pars Studios,

Tehran, Iran.