

Using exceptions in Python Maya plugins

How to work around a rather nasty bug in the way Maya handles Python plugin exceptions... Google turned up empty for me on this one, so I'm posting details here to save others the trouble.

Whilst spelunking around in the Maya API recently, I had the misfortune to stumble upon a rather nasty bug in the way Maya handles Python scripted plugins. After almost a day of wrestling with Maya I finally found a simple workaround, and thought I'd post it here in case any other poor buggers are stuck with the same problem.

First, the bug: if you're using `MPxCommand` in a Python Maya plugin, and you raise an exception in your `dolt()` method, unfortunately Maya will ignore the exception and will keep running any scripts regardless of the fact that an error occurred. So, for a simple example, let's make a command called `raiseException` that does nothing but raise an exception when you run it:

```
{geshibot lang="python"}
#####
# This command deliberately raises an exception when you call it to
# illustrate a bug in Maya
#####

import maya.OpenMaya as OpenMaya
import maya.OpenMayaMPx as OpenMayaMPx

class RaiseExceptionCmd(OpenMayaMPx.MPxCommand):
    """This class implements a custom MEL command."""

    def __init__(self):
        OpenMayaMPx.MPxCommand.__init__(self)

    def dolt(self, arglist):
        """This plugin deliberately raises an exception to illustrate a
        bug in Maya"""
        raise RuntimeError, 'An internal error occurred within this plugin'

def cmdCreator():
    return OpenMayaMPx.asMPxPtr( RaiseExceptionCmd() )

def initializePlugin(mobject):
    """Plugin initialisation - called whenever our plugin is loaded"""
    mplugin = OpenMayaMPx.MFnPlugin(mobject)
    try:
        mplugin.registerCommand('raiseException', cmdCreator )
    except:
        sys.stderr.write('Failed to register command: raiseException\n')
        raise

def uninitializePlugin(mobject):
    """Plugin cleanup - called whenever our plugin is unloaded"""
    mplugin = OpenMayaMPx.MFnPlugin(mobject)
    try:
        mplugin.deregisterCommand('raiseException')
    except:
        sys.stderr.write('Failed to unregister command: raiseException\n')
        raise
{/geshibot}
```

Which is exactly what the Maya docs tell us to do in the event of an error. Unfortunately, if we then called our command in the following simple MEL script, then the results are not what you'd expect:

```
{geshibot lang="javascript"}
print "Hello\n";
raiseException;
```

```
print "World\n";
{/geshibot}
```

Hello

```
#Traceback (most recent call last):
```

```
# File "raiseException.py", line 61, in
```

```
# raise RuntimeError, "An internal error occurred within this plugin"
```

```
# RuntimeError: An internal error occurred within this plugin #
```

World

Which is, as they say, a bit of an arse. Luckily for us, there is still an undocumented way to stop script execution. If our `dolt()` method returns `maya.OpenMaya.kUnknownParameter`, then Maya presumes that the plugin was passed incorrect flags, and stops. We can use this to our advantage - it means we need to catch any exception that our plugin may raise during execution, print it out, and then return `maya.OpenMaya.kUnknownParameter` instead. Or, in code form:

```
{geshibot lang="python"}
def dolt(self, arglist):
    """This method wraps around _dolt, catches any exceptions, prints
    them out to the script editor and then stops script execution
    correctly"""
    try:
        self._dolt(arglist)
    except:
        import traceback
        errorInfo = traceback.format_exc()

        #This is a bit of an awkward way to print things out, but it
        #ensures that the status line in Maya displays the last part
        #of the message (the useful part), rather than just "Traceback:"
        for line in errorInfo.split('\n'):
            if line:
                self.displayError(line)

        return OpenMaya.kUnknownParameter

def _dolt(self, arglist):
    """This is where we now do our actual plugin work"""
    raise RuntimeError, "An internal error occurred within this plugin"
{/geshibot}
```

This then traps errors and stops script execution correctly... As far as I can tell, this issue affects all versions of Maya (8.5 - 2010 at time of writing). I hope this was useful, and saved some of you some head scratching; if anyone knows of a better way to handle this, or knows how to make Maya respect Python exceptions then please let me know!