Python 2.0

Module 10

Autodesk

Autodesk Maya Python API Training

Kristine Middlemiss, *Senior Developer Consultant*
Autodesk Developer Network (ADN)

Autodesk

Day 9 Questions:

1. On "selList.getDependNode(0,self.blendShapeNode)", how come I have to put "0" before blendshapeNode? It was not on document
2. Is there a way to call a Python maya.cmds function call using the MDGModifier (or MDagModifier)? Or are we restricted to using MEL in our call to MDGModifier.commandToExecute()?
3. Any class that I can use to monitor user selection changes?
4. Why do we declare the compute function at all? Seems strange to declare it and then return an error code. Can't we just leave it undeclared and use the superclass's implementation of draw()?
5. How to implement access to the 'b' key during a custom context to get extra input from the user for a brush size for example (see the paint vertex color context). It seems MEvent only tracks modifiers keys from the keyboard. Also, MEvent doesn't seem to allow tracking mouse movement while in a context if the user is not pressing mouse buttons or keyboard modifiers.

Homework Review:

Session Agenda

- Looking at some of the issue in Python 1.0 (The Original)
- Changes made in 2.0
- Where we are in the transition

Python 1.0

Image courtesy of Johan Vikström, Shilo, Ool Digital, Mikros Image

Autodesk®

Python API 1.0

- Generated automatically from C++ headers
- Covers most (not all) C++ classes
- Close enough to C++ interface that they can share documentation

- So why do we need a new one?

Python API 1.0 - Problems

- MScriptUtil

- C++ API uses pointers and references to simple numeric types (int, float, etc) which won't work in Python.
- MScriptUtil provides pointer operations in Python to work around this.
- Cumbersome to use.
- Pointer semantics not very Pythonic.

MScriptUtil Example

```
xsu = om.MScriptUtil(0.0)
xptr = xsu.asDoublePtr()
ysu = om.MScriptUtil(0.0)
yptr = ysu.asDoublePtr()
zsu = om.MScriptUtil(0.0)
zptr = zsu.asDoublePtr()
fluidFn.getDimensions(xptr, yptr, zptr)
dim = (xsu.getDouble(xptr), ysu.getDouble(yptr),
zsu.getDouble(zptr))
```

Python API 1.0 - Problems

- “Dumbing down” of C++ API

- Gives better Python API, but not a lot better
- Makes C++ API worse
- End up with two mediocre APIs instead of two great APIs

Dumbed Down Example

MStatus getDimensions(double&x, double& y, double& z)

- Bad for Python
- Good for C++

double getXDimension(MStatus* st)
double getYDimension(MStatus* st)
double getZDimension(MStatus* st)

- Better for Python (not great)
- Worse for C++

Python API 1.0 - Problems

- Performance overhead.

- Automated tool cannot take method semantics into consideration.
- For simplest method calls (e.g. MAngle.setValue) more time spent converting parameters and return values than in executing the method itself.

Python API 1.0 - Problems

- C++ documentation difficult for Python developers to use.
- Loss of MStatus granularity: everything is a RuntimeError.
- ‘cvar’ access to static MObjects (e.g. OpenMayaMPx.cvar.MPxNode_state) is unintuitive.
- Tracking MPx* objects (e.g. MFnDependencyNode.userNode).

MPx* Objects Example

```
class myNode(ompx.MPxNode):
        nodeDict = {}

        def __init__(self):
                myNode.nodeDict[ompx.asHashable(self)] = self

        def __del__(self):
                del pyTestNode.nodeDict[ompx.asHashable(self)]
…
fn = om.MFnDependencyNode(nodeObj)
nodePtr = fn.userNode()
node = pyTestNode.nodeDict[ompx.asHashable(self)]
```

The New Python 2.0

Image courtesy of Johan Vikström, Shilo, Ool Digital, Mikros Image

Autodesk®

Python API 2.0

- Mostly hand-generated.
  - A few bits are automated. Hoping to increase that.
- Still calls out to the C++ API underneath.
  - Faster to code.
  - Combined Python/C++ testing.
  - C++ API mostly reflects Maya internal structure, so not much to be gained by bypassing it.

Python API 2.0 - Solutions

- Return output values as function result wherever possible.
- Return multiple values as tuples or lists.
- M*Array types become full Python sequences and can be used interchangeably with Python lists almost everywhere.
  Many get/set method pairs get replaced with object attributes.
  - Previously: MFnAttribute.setHidden(True)
  - Now: MFnAttribute.hidden = True
- Methods which have no result return a reference to the object itself so that calls can be chained.
  - obj.method1().method2()
- That gets rid of MScriptUtil!

MScriptUtil Example

```
xsu = om.MScriptUtil(0.0)
xptr = xsu.asDoublePtr()
ysu = om.MScriptUtil(0.0)
yptr = ysu.asDoublePtr()
zsu = om.MScriptUtil(0.0)
zptr = zsu.asDoublePtr()
fluidFn.getDimensions(xptr, yptr, zptr)
dim = (xsu.getDouble(xptr), ysu.getDouble(yptr),
zsu.getDouble(zptr))


dim = fluidFn.getDimensions()  # Returns (x, y, z)
```

Python API 2.0 - Solutions

- Python interfaces not limited to what's possible in C++.
- C++ interfaces not limited to what's possible in Python.
- Best of both worlds.

Dumbed Down Example

double getXDimension(MStatus* st)

double getYDimension(MStatus* st)

double getZDimension(MStatus* st)

C++: MStatus getDimensions(double&x, double& y, double z)

Python: getDimensions() ⟶ (x, y, z)

Python API 2.0 - Solutions

- Tighter interface code reduces overhead.

- MAngle.setValue() goes from 550 nS in API 1.0 to 130 nS in API 2.0
- Only affects interface code, not Maya's internal processing, so most noticeable on methods with short execution times and lots of parameters.

Python API 2.0 - Solutions

- C++ documentation no longer suitable, so API 2.0 gets its own.
  - Will be nicer for developers but does mean more effort on docs for us.
- Wider range of exception types.
- Normal access to static MObjects (e.g. MPxNode.state)
- Tracking of MPx* objects is done internally by Maya.

MPx* Objects Example

```
class myNode(ompx.MPxNode):
        nodeDict = {}

        def __init__(self):
                myNode.nodeDict[ompx.asHashable(self)] = self

        def __del__(self):
                del pyTestNode.nodeDict[ompx.asHashable(self)]
…
fn = om.MFnDependencyNode(nodeObj)
nodeDict = fn.userNode()
node = pyTestNode.nodeDict[ompx.asHashable(self)]

fn = om.MFnDependencyNode(nodeObj)
node = fn.userNode()
```

Python API 2.0 – What It Doesn't Do

- Does NOT provide OO interface to nodes, attributes, etc. You still use Function Sets.
- Does NOT provide seamless integration between API and Maya commands (e.g. commands that can return API objects)
- Doesn't fix flaws in the C++ API (for the most part).

How do you work with Python 2.0

- All of the new API modules are in maya.api.

  ```
  import maya.api.OpenMaya as om
  ```

- The module names are the same as in the old API

- Exception: Proxy classes (MPx****) do not have their own module but are located in the same module as their related classes, the same as in the C++ API.

New Maya Python API

- New and old API classes can be used within the same script, but their objects are not interchangeable.

```
import maya.api.OpenMaya as newOM
import maya.OpenMaya as oldOM

newAttrObj = newOM.MObject()
oldNodeObj = oldOM.MObject()
...

newAttrFn = newOM.MFnAttribute(newAttrObj)
oldNodeFn = oldOM.MFnDependencyNode(oldNodeObj)
print("Attribute name is %s.%s" % (oldNodeFn.name(), newAttrFn.name))
```

New Maya Python API

- This sample code here is not legal and does not work:

```
import maya.api.OpenMaya as newOM
import maya.OpenMaya as oldOM

newAttrObj = newOM.MObject()
oldNodeObj = oldOM.MObject()

...
 newPlug = newOM.MPlug(oldNodeObj, newAttrObj)
```

Python 2.0 Documentation

- You can download the Python 2.0 Documentation here:
  - www.autodesk.com/developmaya

Progress - Python 2.0

Image courtesy of Johan Vikström, Shilo, Ool Digital, Mikros Image

Python API 2.0 – Progress To Date

- Maya 2012: Includes everything necessary to get & set plug data and write custom commands (MPlug, MFn*Attribute, MPxCommand, etc)
- Maya 2013: Adds everything necessary to create custom nodes (MPxNode, MDatablock, MDataHandle, etc)
  - Bit of a slowdown from 2012 due to emphasis on building tools for automation and larger than usual amount of work in other areas of the API.
- May not bother with those portions of the API where Python's performance makes it unusable, such rendering, but we'll survey users first.

Autodesk