Game Developers Conference®

February 28 - March 4, 2011
Moscone Center, San Francisco
www.GDConf.com

Power Python Development for Maya

Jason Parks
Technical Artist
Sony Online Entertainment

UBM TechWeb

GDC

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Conclusion

Background

- SCEA (Sony Computer Entertainment America)
  - 12 years
- SOE (Sony Online Entertainment)
  - 2 years
- Maya and MotionBuilder
- Rigging and Mocap

Presentations

- 2005 Siggraph : Alias MasterClass “MEL for Artists”
- 2005 GDC : Helper Joints - Advanced Deformations on RunTime Characters
- 2006 GDC : Muscle Systems for Game Production
- 2007 Siggraph : Autodesk MasterClass “Python for MotionBuilder Artists”

Poll

- Use Maya
- How many script?
  - MEL?
    - still?
  - Python?
    - Maya? PyMEL?

Poll

- Object Oriented Programming (OOP) experience?
- API/SDK experience?

Poll

- Use a Script Editor?
  - –Notepad ++, UltraEdit, MEL Studio Pro, jEdit

OR

- Use IDE (programming environment)
  - –Eclipse, Wing, Visual Studio

Target Audience

- Maya scripting/programming
- Advanced Scripters
- Ideal experience
  - Lots of MEL
  - Maybe some Python and/or PyMEL

Takeaway

- Python rocks
  - Especially for Maya
- Superb Maya Development =
  - PyMEL
  - Eclipse as primary IDE
  - Wing for debugging in Realtime

Outline

- Background
- Poll
- Takeaway
  - **–Use Cases - Genetics Tool, Rig Build**
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Conclusion

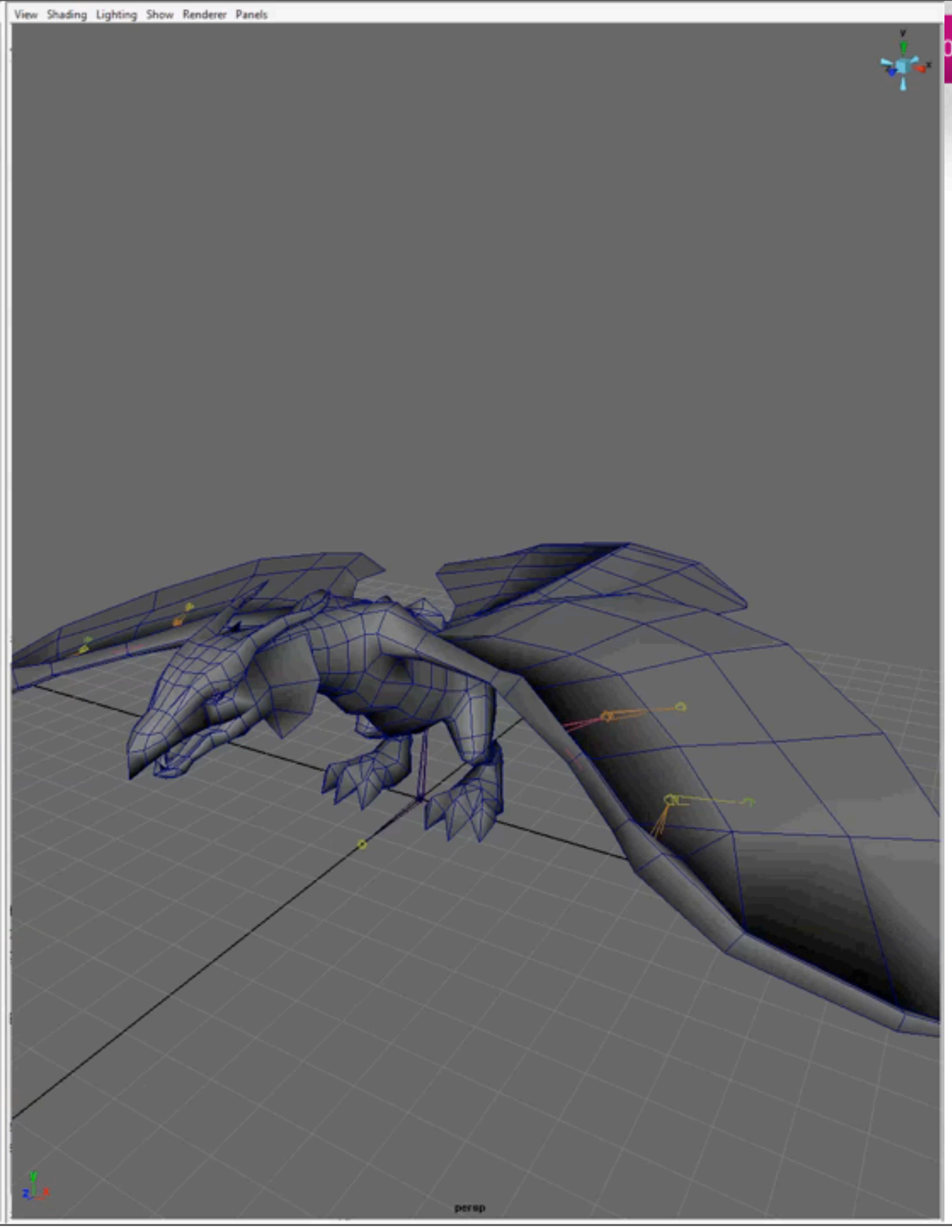

View Shading Lighting Show Renderer Panels
Verts: 8546 0 0
Edges: 16994 0 0
Faces: 8511 0 0
Tris: 15818 0 0
UVs: 10862 0 0
Backfaces: N/A
Smoothness: N/A
Instance: N/A
Display Layer: N/A
Distance From Camera: N/A
Selected Objects: 0
Character Tools
Asset hub
Wear
Genetics
EQ Next Genetics™
Preview :
Human
Ogre
Gender
Skeleton Scale Only (Within Body Type)
Wrap Deform Only (All Body Types)
Fitting Tools :
Wrap Tools
Attribute Editor
Channel Box / Layer Editor
Character Tools

File Edit History Command Panels Help

```
file -f -options "v=0"  -typ "mayaAscii" -o "Y:/FreeRealms/M
// File read in 0 seconds. //
// File read in 0 seconds. //
// File read in 0 seconds. //
// File read in 0 seconds. //
// Warning: line 0: Plug-in, "Mayatomr", is not loaded. //
All MentalRay nodes deleted and plug-in unloaded.
Units set to meters.
Timing set to ntsc/30fps
<<FR_startup sourced.>>
```

MEL | Python | Python | Python | Python | MEL | MEL

```
1  # character template script for
2  # wyvern
3  characterName = 'wyvern'
4
5  # check version
6  from soe_metarigging.utils import metaRigVersi
7  rigVersion = 356546
8  #metaRigVersion(rigVersion, __file__)
9
10 def wyvern_template():
11        # load python modules
12        from soe_rigging.FR_conformChar import
13        from soe_metarigging.FR_metarigging imp
14        from soe_metarigging.attachRig import F
15        from soe_metarigging.cogRig import FR_c
16        from soe_metarigging.finishRig import r
17        from soe_metarigging.rigClasses import
18        from soe_metarigging.quad.tailRig impor
19        from soe_metarigging.face import face_b
20        from soe_metarigging.align import world
21        from soe_libs.libUtilsMaya import jpHic
22        from soe_metarigging.metarigging import
23        from soe_metarigging.utils import impor
24        from pymel.all import *
25
```

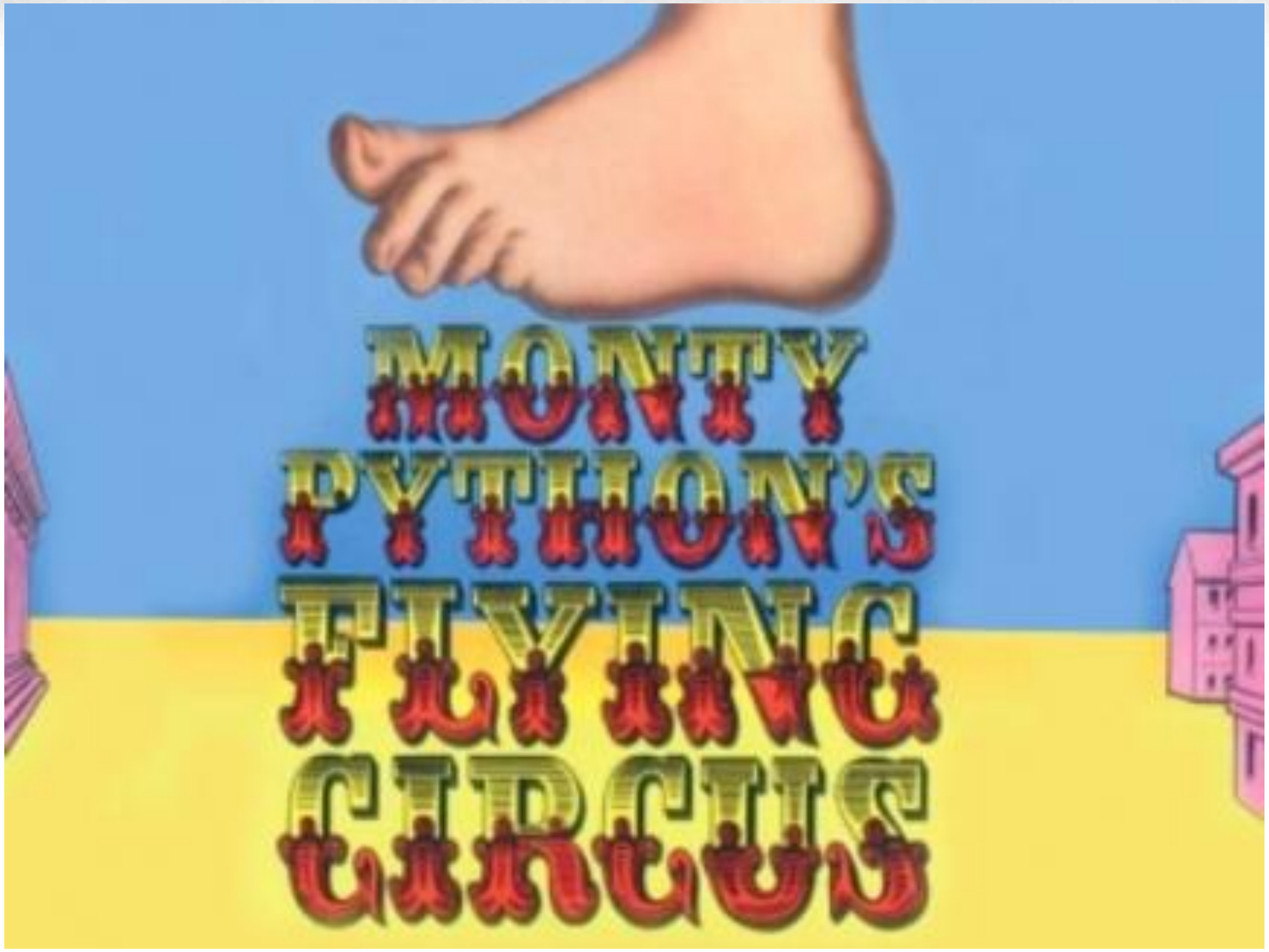


Outline

- Background
- Poll
- Takeaway
- **Maya Embedded Language (MEL) v. Python (maya.cmds)**
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Conclusion

MEL = 0 PYTHON = 1

- No Brainer
  - –MEL was fun, good start
    - First step
  - –Commit some weeks
  - –Second language is easier
    - Python


MONTY
PYTHON'S
FLYING
CIRCUS

Python Features

- String Methods
  - Dozens of cool ways to manipulate, slice strings
  - regex, regular expressions
  - Single, Double, Triple, and Quad quotes
  - Nice formatting

Python Features

- File System Package is complete
  - –shutil, os, os.path, path
- Lists, Dictionaries

```
//MEL
string $myList[] = ();
for ($i=0;$i<10;$i++){
   $myList[$i] = (“item” + $i);}
#Python
for i in range(0,10):
   myList.append(‘item%s’%i)
```

Python Features

- Functions are polymorphic
  - –Call many ways, variables can change type
  - –Arbitrarily nested returns
- Plenty of xml and SQL libraries
- List comprehensions
  - – fun python trick

Python Features

```
// MEL
string $joints[] = {};
int $i = 0;
for ($sel in `ls -sl`)
{
   if (objectType($sel) == “joint”)
   {
       $joints[$i] = $sel;
       $i = $i + 1;
   }
}

# Python
joints = [j for j in ls(sl=1) if objectType(j) == ‘joint’]
```

Python Features

- Logging module replaces all print and debug statements
- Compile to .pyc or .exe
- Tons of resources and tutorials

The KEY Python Feature

- Object Oriented Programming (OOP)
  - Classes and Instances
  - Data Abstraction and Encapsulation
  - Inheritance
  - Polymorphism

- Complicated at first, but scales to very powerful

Summary Python v. MEL

- String Methods
- Clean / readable
- Lists, arbitrary nesting
- Logging
- OOP / Classes

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
  - **Use Case - cvxporter**
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Conclusion

Use Case - cvxporter

- DirectX .x file exporter - Chad Vernon
  - Great example of vertex scraping
  - Lots of OpenMaya and Python wrapped API
  - Nice use of Classes for data containers
  - Pro-level code yet very applicable and easy to reverse engineer
  - http://www.chadvernon.com/blog/resources/cvxporter/

```
def getMeshInfo( self, path, xmesh ):
    """
    Gets attributes for current mesh.
    """
    xmesh.fnMesh.setObject( path )
    if xmesh.fnMesh.isIntermediateObject():
        return False

    instanceNumber = 0
    if path.isInstanced():
        instanceNumber = path.instanceNumber()

    xmesh.name = xmesh.fnMesh.name()

    weights = OpenMaya.MDoubleArray()
    numInfluences = 0
    if self.args['skinning']:
        # Get skin weights
        plugInMesh = xmesh.fnMesh.findPlug( 'inMesh' )
        try:
            itDg = OpenMaya.MItDependencyGraph( plugInMesh, OpenMaya.MFn.kSkinClusterFilter, Ope

            while not itDg.isDone():
                oNode = itDg.currentItem()
                fnSkinCluster = OpenMayaAnim.MFnSkinCluster( oNode )

                # Get components effected by deformer
                fnSet = OpenMaya.MFnSet( fnSkinCluster.deformerSet() )
                members = OpenMaya.MSelectionList()
                fnSet.getMembers( members, False )
                dagPath = OpenMaya.MDagPath()
                components = OpenMaya.MObject()
                members.getDagPath( 0, dagPath, components )

                # Get skin weights
                util = OpenMaya.MScriptUtil()
                util.createFromInt( 0 )
                pNumInfluences = util.asUintPtr()
                fnSkinCluster.getWeights( dagPath, components, weights, pNumInfluences )
```

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- **PyMEL v. maya.cmds**
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Use Cases

PYTHON = .5 PYMEL=1.0

- Get Serious, commit to PyMEL
  - –Autodesk blesses
  - –Not compiled
  - –Open source
    - Add to it
    - Contribute to it
  - –Free

Maya's languages

- MEL = Maya Embedded Language
- Maya Python (maya.cmds) = MEL wrapped in Python, still command based
- PyMEL = Pythonic Maya Langauge

Commands v. Objects

- Key difference

–maya.cmds is MEL in disguise (more white-spaces)

command -flag object

–PyMEL is OOP, based on Classes

object.function(flag)

Classes/OOP

- Class, Instance, Method
  - Create object of ‘Joint’ class

```
# creation of a joint object
myJoint = Joint()

# Perform a method on the instance object
myJoint.listRelatives()

# Result: ‘root’ #
```

Functions as Methods

- 'Pythonic' difference is ***the most important***
  - –All pertinent commands/functions are attached to a node as a method
  - –IDE w/ Code-completion lists all methods
  - –or use dir(node) to get the full list

```
dir(myJoint)
# or a nice list
for method in dir(myJoint): print method
```

Functions as Methods

- example, more readable

```
# maya.cmds
cmds.listRelatives(myJoint, parent=1)

# PyMEL object with method
myJoint.listRelatives(parent=1)

# or one of PyMEL's convenience functions (methods)
myJoint.getParent()

# Result: 'root' #
```

Readability (PEPs 8 & 20)

- *The Zen of Python*

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.

...
```

Functions as Methods

- example: animCurve
  - –PyMEL class derived from maya.OpenMayaAnim.MFnAnimCurve

```
# PyMEL
ac = AnimCurve('COG_translateX')
firstFrame = ac.getTime(0)
lastFrame = ac.getTime(ac.numKeys()-1)

# maya.cmds
last = cmds.keyframe('COG_translateY', q=1, kc=1)
firstFrame = cmds.keyframe('COG_translateY',index=(0,0), q=True)
lastFrame = cmds.keyframe('COG_translateY', index=(last-1,last-1),
q=True)
```

maya.cmds = MEL in disguise

```
# MEL
int $last = `keyframe -q -kc “COG_transalteY”`;
float $firstFrame[] = `keyframe -index 0 -q “COG_translateY”`;
float $lastFrame[] = `keyframe -index ($last-1`) -q “COG_translateY”`;
```

```
# maya.cmds
last = cmds.keyframe(‘COG_translateY’, q=1, kc=1)
firstFrame = cmds.keyframe(‘COG_translateY’,index=(0,0), q=True)
lastFrame = cmds.keyframe(‘COG_translateY’, index=(last-1,last-1),
q=True)
```

Attribute Access

- Super fast

```
# maya.cmds
transX = cmds.getAttr('%s.tx' % myJoint)

# PyMEL
transX = myJoint.tx.get()
```

API Hybridization

- PyNodes keep API dagPath
  - –PyMEL nodes are based on Maya's API dagPaths ("name-independent representation") so your node maintains its whereabouts no matter how your hierarchy gets re-ordered.
  - –Very handy for rigging
  - –No worries on bizarre shapeNode names

MEL call wrapper

- makes MEL calls act like a function instead of eval

```
# maya.cmds      import maya.mel as mm
mm.eval("skinWeightsIO -p \"" + path + "\" -m 0;")


# or
mm.eval('skinWeightsIO -p "%s" -m 0;' % path)


# PyMEL
mel.skinWeights(p=path, m=0)
```

PyMEL logging in Maya

- internal.plogging module
  - –Fixes Maya's Output bug for logging
  - –pymelLogger convenience object

```
# PyMEL
from pymel.internal.plogging import pymelLogger

pymelLogger.info('Regular user info here')
pymelLogger.warning('Colored output')
```

PyMEL logging in Maya

- loggingControl module
  - –Adds Logging Control menu to Maya
  - –easy access to all logging levels, streams, etc.

```
# PyMEL
from pymel.tools import loggingControl
loggingControl.initMenu()
```

imate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character SOE Global Creator Safe Muscle Logging Control Hel

Autodesk Maya 2008 Extension 2: ./untitled

ation Objects

shelfLayout1

mel mel
JP JP

File Edit History Command Panels Help

MEL Python Python Python logger Reload

```
1  from pymel.tools import loggingControl
2  loggingControl.initMenu()
3
4  from pymel.internal.plogging import pymelLogger
5  pymelLogger.setLevel(logging.DEBUG)
6  pymelLogger.setLevel(logging.INFO)
7
8  pymelLogger.debug('bug')
9  pymelLogger.info('test')
10 pymelLogger.warning('warn')
11 pymelLogger.error('lookout!')
12 pymelLogger.critical('crit')
13
```

Desktoppymel.log

Last Saved: 12/6/10 9:05:36 PM
File Path: ~/Desktoppymel.log

Desktoppymel.log

```
1 INFO:pymel:Added File-Handler to Logger 'pymel' at level NOTS
2
```

PyMEL UI

- uitypes module
  - Layout class
  - window class

```
# PyMEL
win = window(title="My Window")
layout = horizontalLayout()
for i in range(5):
    button(label="button %s" % i)
layout.redistribute()  # now will redistribute horizontally
win.show()
```

Autodesk Maya 2008 Extension 2: ./untitled

```
win = window(title="My Window")
layout = horizontalLayout(ratios=[1,0,2], spacing=10)
for i in range(5):
    button(label="button %s" % i)
layout.redistribute()    # now will redistribute horizontally
win.show()

layout.flip()     # flip the orientation
layout.reverse()  # reverse the order of the controls
layout.reset()

for ctrl in win.walkChildren():
        ctrl.setEnable(False)
```

MEL to python translator

- pymel.tools.mel2py
  - Converts entire scripts/directories
  - converts passed string

```
# PyMEL
import pymel.tools.mel2py as mel2py
mel2py.mel2pyStr('joint -n "root";')


# joint(n="root")
```

Performance

- Speed disadvantage
  - –wraps maya.cmds and maya.OpenMaya
    - Lots of conversion from string to/from MObject
  - –Shows itself during heavy vertex iteration

- Very small price to pay

Performance

- Addressing the issue
  - –Educate users on efficient practices
  - –Add features to avoid waisted conversions
  - –Profile and collect data
  - –Working/lobby with Autodesk

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
  - **–Use Case - Pythonized MetaNetwork**
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- Conclusion

Use Case - Pythonized MetaNetwork

- MetaRigging Concept - David Hunt/Seth Gibson
  - –"Modular Procedural Rigging" - GDC 2009
  - –Toolkit is open source MEL
  - –Convert to Python
  - –Convert to Class/Method structure
  - –Create new PyMEL/Maya node type 'MetaNetwork'
    - Sub-classing nt.Network using PyMEL's registerVirtualClass

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- **Eclipse IDE and PyDev**
- Wing IDE as Debugger
- Use Cases

Integrated Development Environment

- Commit to Serious Dev Software
- Need REAL features like:
  - auto-completion
  - code analysis
  - version control

Eclipse as IDE w/ PyDev

- Open-source is totally viable
- Eclipse is free
  - –open source
  - –very popular
  - –tons of add-ons, plug-in SDKs
    - PyDev plug-in turns Eclipse into Python powerhouse (pydev.org)

Eclipse: Mark Occurrences

- Best feature possible
- Fastest way to see what is happening to your variable

–(demo)

_sysMetaNetwork | mayaCore | test_mayaMetaRigging | cogRig | all

```
        mi()

    # widgetToggle control
    if not objExists(widgetGrip):
        widgetGrip = Joint(parent(createGripCube(gripScale, widgetGrip), moveAllGrip)[0])
        mi()


    # creates scriptJob for component manipulation
    widgetGrip.addAttr('selectJobNum', at='long', keyable=0)
    #scriptNodeCommand = '$jobNum = `scriptJob -compressUndo 1 -killWithScene -event "SelectionChanged" "python(\\"from soe_metarigg
    scriptNodeCommand = '$jobNum = `scriptJob -compressUndo 1 -killWithScene -event "SelectionChanged" "python(\\"from soe_metariggi

    #scriptNodeCommand = '$jobNum = `scriptJob -compressUndo 1 -killWithScene -event "SelectionChanged" "python(\\"from pymel.all imp
    #print "scriptNodeCommand is: ' % s'" % scriptNodeCommand
    sn = scriptNode(n='widgetToggleSN', st=1, bs=scriptNodeCommand)
    # execute once
    scriptNode(sn, eb=1)

    # add dagMenu procs
    moveAllGrip.addAttr('zooCmd0', dt='string')
    moveAllGrip.zooCmd0.set('turnCtrlToggleAll ^ python("from soe_metaNetwork import metaNetwork\\nvis = 1\\nmRoot = metaNetwork.Meta


    # hide and lock
    jpHideAndLock(rootChild_, 1, 1, 1, 1, 0, 0, 1, 1)
    jpHideAndLock(moveAllGrip, 0, 0, 1, 1, 0, 0, 1, 1)
    jpHideAndLock(widgetGrip, all=1)
    jpHideAndLock(rootJoint, 1, all=1)

    ############################################################
    ####################   META DATA   #########################
    ############################################################

    # TODO: Move these all into mayaMetaRigging
    metaRoot, mtAnimRig = FR_getMetaNodes(rigVersion, charName)

    # set special attrs
    moveAllGrip.side.set(3)
    moveAllGrip.attr('type').set(1)
    moveAllGrip.otherType.set('Center of Gravity')
    widgetGrip.side.set(3)
    widgetGrip.attr('type').set(18)
    widgetGrip.otherType.set('widgetToggle')
    connectToMeta(mtAnimRig, moveAllGrip, 'moveAllGrip')
    connectToMeta(mtAnimRig, widgetGrip, 'widgetGrip')

    return metaRoot, rootChild_, moveAllGrip

def moveAll_toggleVis():
    """ function to toggle visibility of controls.
        To be called via a scriptJob on moveAll_anm controller "
```

Eclipse: Code Completion

- Combine with Class based methods of PyMEL gives you nearly any possible action or query you would want to do on that object
- SUPER POWERFUL!

  –(demo)

cogRig | general | nodetypes | *test

```
'''
Created on Dec 27, 2010

@author: jason
'''
from pymel.all import *                          #@UnusedWildImport

myJoint = Joint(n='test')

myJoint.getPreferredAngle()
myJoint.getMinRotateDampZRange()
```

Eclipse: Tool Tips

- preview window into the full function
- jump to function

–(demo)

cogRig | general | nodetypes | *test

```
45    # update rigVersion w/ Time
46    rigVersion = int(rigVersion) + dateAsDecimal()
47
48    gripScale = mCore.getNodeFromSchema('gripScale')
49    moveAllGrip = mCore.getNodeFromSchema('moveAllGrip')
50    cogTurnGrip = mCore.getNodeFromSchema('cogTurnGrip')
51    cogOffset_ = mCore.getNodeFromSchema('cogOffset_')
52    cogGrip = mCore.getNodeFromSchema('cogGrip')
53    cogGimbal_ = mCore.getNodeFromSchema('cogGimbal_')
54
55    if not objExists(moveAllGrip): metaRoot, rootChild_, moveAllGrip = DEEP_createBase(rigVersion, charName)
56    else:
57        metaRoot = FR_getMetaNodes(rigVersion, charName)[0]
58        rootChild_ = mCore.getNodeFromSchema('rootChild_')
59
60    # check if probably run already
61    if objExists(cogTurnGrip) and PyNode(cogTurnGrip).getParent() == moveAllGrip:
62        mCore.logger.error("'%s' already parented to '%s', %s probably already run" % (cogTurnGrip, moveAllGrip, __file__))
63        return [False]
64    # turn Control
65    cogTurnGrip = Joint(parent(createGripCube(gripScale, cogTurnGrip), moveAllGrip)[0])
66
67    # offset Controls
68    cogOffset_ = group(em=1, p=cogTurnGrip, n=cogOffset_)
69
70    # try to position between hips
71    cogLoc = CA_schemaParser.ParseSchema().getWellFormed('cogLoc')
72    if objExists(cogLoc):
73        delete(cmds.pointConstraint(cogLoc, cogOffset_))
74    elif objExists("R_hip") and objExists("L_hip"):
75        delete(cmds.pointConstraint("R_hip", "L_hip", cogOffset_))
76    else:
77        delete(cmds.pointConstraint(cogJoint, cogOffset_))
78
79    cogGrip = Joint(parent(createGripCubeAt(cogOffset_, gripScale, cogGrip), cogOffset_)[0])
80    mi()
81
82    # extra gimbal
83    cogGimbal_ = group(em=1, p=cogGrip, n=cogGimbal_)
84    mi()
85
86    # constrain COG
87    pc = cmds.parentConstraint(cogGimbal_, cogJoint, mo=1, weight=1)
88    if pc.__class__ == list: pc = pc[0]
89    jpHideAndLockPCons(pc)
90
91    # add dagMenu procs
92    cogGrip.addAttr('zooCmd0', dt='string')
93    cogGrip.zooCmd0.set('turnCtrlToggle^python("turnCtrlShape = listCP(listMetaParent(SCENE.#).turnControl).getShape()\\nif turnCtrlShape.v.get(): turnC
94
95    # hide and lock
96    jpHideAndLockDetailed(cogTurnGrip, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1)
97    jpHideAndLock(cogOffset_, all=1)
98    jpHideAndLock(cogGrip, 0, 0, 1, 1, 0, 0, 1, 1)
99    jpHideAndLock(cogGimbal_, 1, 0, 1, 1, 1, 0, 1, 1)
100   jpHideAndLock(cogJoint, 1, 1, 1, 1, 0, 0, 1, 1)
101
102   ############################################################
103   ####################   META DATA   #####################
```

Outline

- objExists,parent (pymel.core.gen
- scriptNode (pymel.core.system)
- dateAsDecimal (soe_libs.libUtils)
- jpHideAndLock,mi,jpHideAndLoc
- * (soe_maya.all)
- FR_getMetaNodes,FR_getGripScal
- createGripCube,createGripCubeA
- FR_createCOGrig
- DEEP_createBase
- moveAll_toggleVis
- __main__

Eclipse: Auto importing

- ‘context-insenstive’ code completion
- bring up pythonpath and automatically import

—(demo)

cogRig | general | nodetypes | test | hands

```
'''
Created on Dec 27, 2010

@author: jason
'''



from pymel.all import *                    #@UnusedWildImport

myJoint = Joint(n='test')

myJoint.getPreferredAngle()
myJoint.getMinRotateDampZRange()

myJoint.setRotationOrder('XYZ')
```

Eclipse: Code Analysis

- Pydev finds
  - –Undefined/Unused variables/imports
  - –No ‘self’ token in class methods
  - –Mixed/bad indentation
  - –syntax errors
- All on the fly, while you are coding
  - –(demo)

cogRig | general | nodetypes | test

```
'''
Created on Dec 27, 2010

@author: jason
'''
from soe_metarigging import hands


hands.hands('joint1', 'joint2')


from pymel.all import *                    #@UnusedWildImport

myJoint = Joint(n='test')

myJoint.getPreferredAngle()
myJoint.getMinRotateDampZRange()

myJoint.setRotationOrder('XYZ')
```

Eclipse: Outliner

- Real Outliner
  - –Shows everything
  - –Hierarchy
  - –Icons
  - –“Bells and whistles”
  - –(demo)

cogRig | general | nodetypes | *test | mayaMetaNetwork

```
#-------------------------------------------------------------------//
from copy import copy #@UnusedImport

class Meta(object):
#class Meta:

    """
    def __init__(self, mtNode='', metaType='', metaParent='', common=0, metaClass='', **kwargs):

    def __repr__(self):

    def _createSystemRoot(self):

    def _create(self):
    def cast(self, mtNode=''):

    ###################

    def beyondVersion(self):

    def connectToMeta(self, node, attrName='', force=0, quiet=1):

    def connectChainToMeta(self, chain, attr_, force=0, quiet=1):

    def ex(self):

    def getMtNode(self):

    def getMetaChildOfType(self, metaType):

    def getMetaChildren(self):

    def getMetaChild(self, name=''):

    def getMetaParent(self):

    def getMetaParents(self):

    def setModified(self):

    def validate(self, name):

    @property
    def name(self):

class MetaSceneBrain(Meta):
    """

    def __init__(self, mtNode='', metaType='', metaParent='', common=0, metaClass='MetaSceneBrain', **kwargs):

    def createSystemRoot(self):

    ###################

    def verifyAttributes(self, version=1.0):

    def updateSceneMeta(self, fromVersion):
```

Outline

- copy (copy)
- * (pymel.all)
- * (soe_maya.mayaCore)
- datetime
- re
- Meta
  - __init__
    - self.mtNode
    - self.metaType
    - self.metaType
    - self.metaParent
    - self.common
    - self.metaClass
    - self.schemaObj
  - __repr__
  - _createSystemRoot
    - self.mtNode
    - self.metaParent
  - _create
    - self.mtNode
    - self.mtNode
    - self.mtNode
    - self.metaParent
  - cast
    - self.mtNode
    - self.mtNode
  - beyondVersion
  - connectToMeta
  - connectChainToMeta
    - self.mtNode
  - ex
  - getMtNode
  - getMetaChildOfType
  - getMetaChildren
  - getMetaChild
  - getMetaParent
  - getMetaParents
  - setModified
  - validate
  - name
- MetaSceneBrain
  - __init__
  - createSystemRoot
    - self.mtNode
    - self.metaType
  - verifyAttributes
    - self.version
  - updateSceneMeta
  - updateSceneBrain
  - getSceneBaseName
  - getAssetRootPath
  - findPCSkeletonNode
  - getSceneResourcePath
  - getWearClass
  - getWearTier
  - getWearSet
- MetaAssetHub
  - __init__
  - createSystemRoot
    - self.mtNode

Eclipse: Maya Editor

- Sends to Maya
- Embedded documentation
  - API
  - maya.cmds
  - PyMEL
  - http://www.creativecrash.com/maya/downloads/applications/syntax-scripting/c/eclipse-maya-editor

Eclipse: Real-time debugging

- Can remotely debug Maya with Eclipse
  - –Not ideal implementation
    - –Missing some key features that WingIDE has
    - –http://pydev.org/manual_adv_remote_debugger.html

Eclipse: Summary

- It's about the 'Environment'
- Aware of code base
- Jumps to functions
- Knows what methods are available
- It's a very fast way to program
- A professional's tool

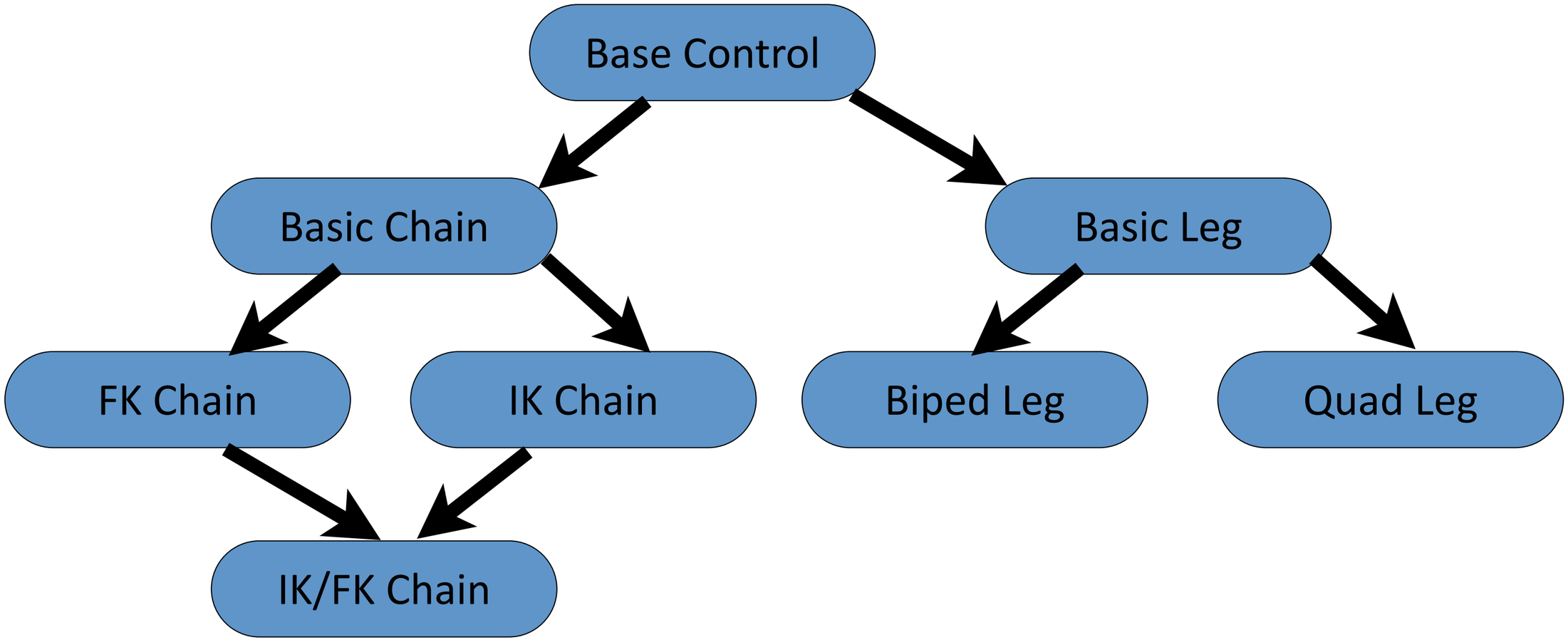
Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
  - **Use Case - RigClasses**
- Wing IDE as Debugger
- Conclusion

Use Case - RigClasses

- Turn rig components into inherited Class structure
  - –Start w/ basic rigging
    - FK Control
  - –Work way to more advanced
    - Simple FK Chain
    - IK Chain
    - Specific IK Chain

Use Case - RigClasses



Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- **Wing IDE as Debugger**
- Conclusion

Wing IDE

- Python-only IDE
- Nice features
- Though clunky
- Might just need Wing
- Can use both Wing and Eclipse
- $180 - pro version

Wing's Killer App

- Real-time debugging

  – Change your life as a Maya Python programmer

  – Impossible to describe how valuable a feature this is

Debugging the old way

- Sucks
- Lots of print statements
- UI prompts
- Way too slow

Wing: Real-time debugging

- Start Wing and Maya
- Establish connection
- Set breakpoint in script
- Call script from Maya
- Wing grabs the process and halts Maya
- (demo)

Autodesk Maya 2011 x64: untitled*

File Edit Modify Create Display Window Assets Lighting/Shading Texturing Render Toon Stereo Paint Effects SOE Global EQ Next Logging Control Perforce

Rendering

EQN_RigTemplates AnimationFriendlyRigging CW_RigTemplates Exporters FBIK_CTRL FR_RigTemplates Joints Misc Animation Rigging

Display Show Panels

File Edit History Command Panels Help

View Shading Lighting Show

persp
top
front
side
defaultLightSet
defaultObjectSet
brush1
defaultLightList1
defaultRenderUtility
defaultShaderList1
defaultTextureList1
defaultLayer
layerManager
dof1
dynController1
globalCacheControl
hardwareRenderGlc
defaultHardwareRe
hyperGraphInfo
hyperGraphLayout
ikSystem
lambert1
lightLinker1
lightList1
initialMaterialInfo
animLayersFilter
defaultRenderLayer
layersFilter
notAnimLayersFilter
relationshipPanel1L
relationshipPanel1R
renderingSetsFilter
renderLayerFilter
objectNameFilter4
objectScriptFilter10
objectTypeFilter74
objectTypeFilter75
objectTypeFilter76

unitTests Python Python Python Python Python Python

```
1 import demo
2 reload (demo)
3 demo.runDemo()
```

persp

1 2 4 6 8 10 12 14 16 18 20 22 24 1.00

Python

Select Tool: select an object

Wing IDE: SOEglobal_wing4.wpr: demo.py (SOEcommon)

File Edit Source Refactor Project Debug Testing Tools Window Help

CA_schemaParser.py (r/o) demo.py general.py (r/o) mayaCore.py (r/o) mayaMetaRigging.py

runDemo

```
1  '''
2  Created on Jan 4, 2011
3
4  @author: jparks
5  '''
6
7  from pymel.all import *
8
9  def gdcDemo():
10
11     eclipse = 'awesome'
12     wing = 'super debugger'
13
14     #cmds.sphere(n='test')
15
16     print eclipse, wing
17
18     pass
19
20
21 def runDemo():
22     #confirmDialog()
23     gdcDemo()
24
25     print 'done'
26
```

Call Stack Project Snippets Source Browser

MainThread() #0xcccc8990 (first to stop)

Stack Data

MainThread() #0xcccc8990 (first runDemo(): demo.py, line

Variable	Value
locals	{}
globals	<dict 0x41c18bd8L; len=0>

Search Search in Files Watch

MainThread() #0xcccc85 runDemo(): demo

Watching	Value: Cannot evaluate: d
refresh()	None
eclipse	<error evaluating>

Debug Probe Messages Modules OS Commands

MainThread() #0xcccc8990 (fi runDemo(): demo.py, lir

Cannot execute: debug process is busy Options

```
>>>
```

Indentation Source Assistant

Symbol: #confirmDialog()
Cannot determine type

Line 22 Col 15 - [User] Debugger: Debug process running; pid=5332 [3 modules loaded]

Wing: Maya debugger

- Break-points
- Watch variables
- Debug probe
  - –autocomplete
  - –see what methods are available at runtime
- Call stack
  - –see what functions led to current spot

Autodesk Maya 2011 x64: untitled*

File Edit Modify Create Display Window Assets Lighting/Shading Texturing Render Toon Stereo Paint Effects SOE Global EQ Next Logging Control Perforce

Rendering

EQN_RigTemplates | AnimationFriendlyRigging | CW_RigTemplates | Exporters | FBIK_CTRL | FR_RigTemplates | Joints | Misc | Animation | Rigging

Display Show Panels

File Edit History Command Panels Help

View Shading Lighting Show

persp
top
front
side
defaultLightSet
defaultObjectSet
brush1
defaultLightList1
defaultRenderUtility
defaultShaderList1
defaultTextureList1
defaultLayer
layerManager
dof1
dynController1
globalCacheControl
hardwareRenderGlc
defaultHardwareRe
hyperGraphInfo
hyperGraphLayout
ikSystem
lambert1
lightLinker1
lightList1
initialMaterialInfo
animLayersFilter
defaultRenderLayer
layersFilter
notAnimLayersFilter
relationshipPanel1L
relationshipPanel1R
renderingSetsFilter
renderLayerFilter
objectNameFilter4
objectScriptFilter10
objectTypeFilter74
objectTypeFilter75
objectTypeFilter76

unitTests | Python | Python | Python | Python | Python | Python

```
import demo
reload (demo)
demo.runDemo()
```

1 2 4 6 8 10 12 14 16 18 20 22 24 1.00

Python

Select Tool: select an object

Wing IDE: SOEglobal_wing4.wpr: demo.py (SOEcommon)

File Edit Source Refactor Project Debug Testing Tools Window Help

CA_schemaParser.py (r/o) | demo.py | general.py (r/o) | mayaCore.py (r/o) | mayaMetaRigging.py

gdcDemo

```
6
7   from pymel.all import *
8
9   def gdcDemo():
10
11      eclipse = 'awesome'
12      wing = 'super debugger'
13
14      #cmds.sphere(n='test')
15
16      print eclipse, wing
17
18      pass
19
20
21  def runDemo():
22      #confirmDialog()
23      gdcDemo()
24
25      print 'done'
26
```

Search | Search in Files | Watch

MainThread() #0x gdcDemo():

Watching	Value: Cannot eva
refresh()	None
eclipse	"awesome"

Debug Probe | Messages | Modules | OS Commands | Python Shell

MainThread() #0xcccc8990 (first to stop) gdcDemo(): demo.py, line 18

Cannot execute: debug process is busy | Options

```
>>>
```

Line 18 Col 8 - [User]

Call Stack | Project | Snippets | Source Browser

MainThread() #0xcccc8990 (first to stop)

Stack Data

MainThread() #0xcccc8990 (gdcDemo(): demo.py

Variable	Value
locals	{'eclipse': 'awesome', 'wing': '
eclipse	'awesome'
wing	'super debugger'
globals	<dict 0x41c18bd8L; len=0>

Indentation | Source Assistant

Symbol: pass

Cannot determine type

Wing: Maya debugger

- Stack data
  - –current state of data at that moment in code
  - –updates as you traverse up and down call stack
    - allows you to see value of variable as it passes through functions
  - –Ability to reverse engineer obscure classes via instantiated objects
  - –Can change/update in realtime

Autodesk Maya 2011 x64: untitled.ma*

```
select -r test ;
doDelete;
```

```
import demo
reload (demo)
demo.runDemo()
```

Wing IDE: SOEglobal_wing4.wpr: demo.py (SOEcommon)

```
from pymel.all import *
from fileIO import CA_schemaParser

def gdcDemo():

    eclipse = 'awesome'
    wing = 'super debugger'

    #cmds.sphere(n='test')

    print eclipse, wing

    parser = CA_schemaParser.ParseSchema()

    pass


def runDemo():
    #confirmDialog()
    gdcDemo()
```

```
>>> print eclipse
awesome
>>> joint(n='test')
>>> refresh
```

Symbol: CA_schemaParser
Likely type: module

Wing: Maya debugger

- Normally Maya process is frozen
- Refresh trick
  - Put ‘refresh()’ command in your Watch list in Wing
  - You can watch maya update as you step though line by line

Wing: Features

- Unit Testing Tool
  - –Convenient grouping
  - –Easy icons
  - –Clear results
  - –Run test in debug mode!

Wing: Summary

- Want professional coder's features
  - –debugging, break-points, stack-trace, etc.
- Wing's Real-time debugging is huge
  - –Single most import ability!
  - –Allows you to iterate quickly

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
  - **Use Case - RigClasses combined with Pythonized MetaNetwork**
- Conclusion

Use Case - RigClass Embedded MetaNetwork

- MetaNodes are brains for rig components
  - –Have connections to each part of that rig bit
- Connect all the MetaNodes to a central brain
  - –Nervous system for the entire rig
- Code to build rig component is an instantiated class
- Instance has tons of attributes and possibly plenty of utility methods to tweak, change the rig

Use Case - RigClass Embedded MetaNetwork

- Instance is gone as soon as build code is complete
- What if?
  - –Embed that instantiated object *into* the MetaNode for that rig component
  - –Anytime look at MetaNode, retrieve the instance (re-create it?)
  - – run a method on retrieved instance
    - Delete rig, change it, re-create it

Use Case - RigClass Embedded MetaNetwork

- Can it be done?
  - –Impossible in MEL
  - –Must use PyMEL and Classes
  - –Need environment aware IDE
  - –Must have Real-time debugging

Outline

- Background
- Poll
- Takeaway
- Maya Embedded Language (MEL) v. Python (maya.cmds)
- PyMEL v. maya.cmds
- Eclipse IDE and PyDev
- Wing IDE as Debugger
- **Conclusion**

Conclusion

- Top 3 important moments in DCC apps in last 12 years:
  - –Maya release
    - built with MEL
  - –Maya supports Python
    - really PyMEL's implementation
  - –Advanced Debugging
    - Eclipse is powerful
    - Wing's Real-time connection to Maya = godsend

Thanks

- Christian Akesson, @cakesson33
- Seth Gibson, @voMethod
- Eric Pavey, www.akeric.com
- Chad Vernon
- @riggingdojo

Questions?

- Twitter: @count_zer0
- Email: jparks@soe.sony.com

Links

- PyDev
  - http://pydev.org
- Akesson's Eclipse Setup page:
  - http://christianakesson.com/ArtSite/Blog/Entries/2010/11/13_Making_Eclipse_Soar.html
- PAIE
  - http://www.creativecrash.com/maya/downloads/scripts-plugins/animation/c/paie
- cvxporter
  - http://www.chadvernon.com/blog/resources/cvxporter/
- Wing connection
  - http://mayamel.tiddlyspot.com/#%5B%5BInteraction%20between%20Wing%20and%20Maya