Slicer4 and Python

Jean-Christophe Fillion-Robin & Julien Finet
Kitware Inc.
January 11th 2011

Demo

*

Kitware

Overview

- Embedded Python Interactor `(Ctrl-P)`
- Convenient API
  - slicer.app, slicer.mrmlScene, slicer.util, slicer.modules, ...
- Python modules
  - vtk, ctk, qt, numpy, scipy, ...
  - slicer libs (mrml, vtkTeem, ...)
  - slicer, slicer.cli, slicer.util, …
- slicerqt.py

Source vs Build – What goes where

- Python source script at different location:
  - QTScriptedModule/Scripts
  - Base/Python
  - Libs/MRMLDisplayableManager/Python
  - …
- All end up in: **`Slicer-build/bin/Python`**
- Thanks to: **`ctkMacroCompilePythonScript`**

Kitware

Slicer Python from the command line

- Command line options

```
./Slicer --no-main-window --python-
  script  path/to/script.py

./Slicer path/to/script.py
```

- Slicer python script

```
#! /path/to/Slicer
print “Hello Slicer”
```

Scripted loadable module (1/2)

- Where: **`QTScriptedModules/Scripts/`**
  Endoscopy.py, Editor.py, …

- Should contains two classes:
  Endoscopy
  EndoscopyWidget

- Slicer integration – C++ / python inteface
  qSlicerScriptedLoadableModuleFactory
  qSlicerScriptedLoadableModule
  qSlicerScriptedLoadableModuleWidget

Scripted loadable module (2/2)

- Endoscopy

```
def __init__(self, parent):
    parent.title = 'Endoscopy'
    parent.help = '…'
    …
```

- EndoscopyWidget

```
def __init__(self, parent = None):
    if not parent:
        # Slicelet mode

def setup(self):
    # Build UI
```

Python and UI integration

- Create UI using QtDesigner

- Load and instantiate from Python

```
f = qt.QFile('/path/to/helloSlicers.ui')
f.open(qt.QFile.ReadOnly)
loader = qt.QUiLoader()
widget = loader.load(f)
f.close()
widget.visible = True
```

Slicer customization

- ~/.slicerrc.py
  - Resource file in your user directory loaded at startup time.
  - Example: http://github.com/jcfr/SlicerPy

Coming next ...

- Tune the API
- Testing and Python
- ctkWorkflow / Python / CLI integration
- Performance improvement
- Integration of IPython
- Python code coverage
- Generation of python documentation
- Reload of python module at runtime ?
- 'slicer' python module ?