Using PyQt4 in Maya 2008

July, 2007

Published by:
Autodesk, Inc.
111 McInnis Parkway
San Rafael, CA 94903, USA

Introduction

Qt Designer is a powerful GUI layout and forms builder, which enables rapid development of high-performance user interfaces with native look and feel across all supported platforms. Qt Designer includes powerful features such as preview mode, automatic widget layout, support for custom widgets, an advanced property editor and more.

PyQt is a set of Python bindings for Trolltech's Qt application framework and runs on all platforms supported by Qt including Windows, Linux, and Mac OS X.

Using PyQt and Qt, specialized GUI can easily be created and hooked into Maya through Python on Windows and Linux.

QT interfaces running in Maya through PyQt on Mac OS X are not working in Maya 2008.

The Workflow

1. Design your GUI in Qt Designer.
2. Save your .ui file.

   - In the accompanying examples, *sample.ui* is the .ui file from Qt Designer.

3. Convert your .ui file to a .py file using pyuic4.bat (pyuic4.bat is a batch script that gets installed in your Python installation when you install PyQt4.)

   - In the accompanying examples, *sample.py* is the script generated from the *sample.ui* file with pyuic4.bat.
   - The command in Windows that converts the .ui file into its Python equivalent is:

   ```
   C:\Python25\pyuic4.bat sample.ui -o sample.py
   ```

4. Create a python script that calls the converted file, defines and hooks up your callbacks and starts *pumpThread.py*. (*pumpThread.py* is a module which processes Qt Events in a secondary thread. It can be found in the accompanying examples.)

   - This fourth step allows you to change your GUI later without losing your manual coding.

   - In the examples, *mysample.py* is the script you run to launch the GUI in Maya
     - In a Python tab in Maya the command is :

       import mysample; mysample.mysample()

- Another example, *mysample_J.py* shows how to hook up a Maya callback to the same *sample.py* file.

When using PyQt in Maya you work with four separate files:

- The **.ui** file which is made in Qt Designer.(the .ui file should be saved in case you wish to change your GUI design later on, but is not otherwise needed after it is converted.)
- The **.py** file which is the converted .ui file.
- A separate **.py** file you create, which starts up the pumpThread.py, instantiates the GUI, creates functions, and hooks these functions into the GUI. As long as you keep your widget's unique names the same between GUI iterations, your callback connections will not be broken.
- The **pumpThread.py** is a Maya-threaded Qt Events Processor. It refreshes the Qt GUI without the Maya GUI being affected.

The example files need to be in your local maya/scripts folder:

- Put *userSetup.py* and *pythonScripts.py* in your maya/scripts folder to implement the pythonScripts menu, which will appear in your Main Menus when you open Maya.
- Under the pythonScripts menu, you should be able to select any python script that you create. As long as your python script’s main function has the same name as the python script, it will bring up your GUI.
- Make a special folder named **zGUI** in your local maya/scripts folder to hold your Qt- and pyuic4.bat-generated files. The **zGUI** folder will then stay at the bottom of the pythonScripts menu and out of your way, so you won't accidentally select the wrong file.

Descriptions of the accompanying example files:

- makeStuff.ui: an original Qt Designer .ui file
- makeStuff.py: the .ui file converted to .py
- myMakeStuff.py: the separate .py file (see third file mentioned above). Also contains an example of checkBoxes changing the outcome of buttonClicks.
- sample.ui: an original Qt Designer .ui file
- sample.py: the .ui file converted to .py
- mysample.py: a separate .py file
- mysample_J.py: another seperate .py file
- frmConnect.py: a .ui file converted to .py
- mysampleStripped.py: One dialog with callback.
- mysampleStripped2.py: One dialog with different callback.
- dockwidgetsCall.py: Calls in an example MainWindow from PyQt4. (Edit this file and change the location of your dockwidgets example files.)
- pumpThread.py: Maya-threaded Qt Events Processor. This module is used by other scripts, may be unstable if executed separately.

You can also use PyQt4 within Maya to better understand PyQt at a lower level.

The following commands will create a standalone window labeled “Hello world!”

In a Maya Python tab, execute the following in succession:

```
##################################

import sys

import PyQt4 as qt

# Instantiate a QApplication, passing the arguments of the script to it:

app = qt.QApplication(sys.argv)

# Add a basic widget to this application
# The first argument is the text you want this QWidget to show, the second
# one is the parent widget. Since the "hello" is the only widget, the
# so-called "MainWidget", it does not have a parent.

hello = qt.QLabel("Hello world!", None)

# ... and that it should be shown.

hello.show()

##################################
```

CAUTION: running the above script twice in succession will kill Maya immediately.

This is because you can’t have more than one qt.QApplication(sys.argv) running at once. You get around this by using the *pumpThread.py* module, which checks to see if there is a pumpThread running before starting itself and the global QApplication(sys.argv), which subsequent GUIs can then reference.

Once your GUIs start getting more complex, you have to use the *global application instantiation* and *global windows* variables.

If you don't use both the *global application instantiation* and *global windows* variables**,** then the resultant GUI will appear and disappear very quickly as Python cleans up its local variables.

Running the following script, with the variables, displays a stable “Hello world!” label.

```
######################################
```

```
import sys

import PyQt4 as qt

app=None

win=None

def windo():

      global app

      global win

      app = qt.QApplication(sys.argv)

      win = qt.QLabel("Hello world!",None)

      win.show()

windo()
```

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

Notice that there is no app.exec_loop(). Maya performs this function.

If you want to use PyQt4 examples from elsewhere, you must add in these two global variables, and comment out or get rid of the app.exec_loop(). You may also need to import pumpThread and initializePumpThread as seen in the example scripts.