

How to Design a Graphical Interface with QT Creator

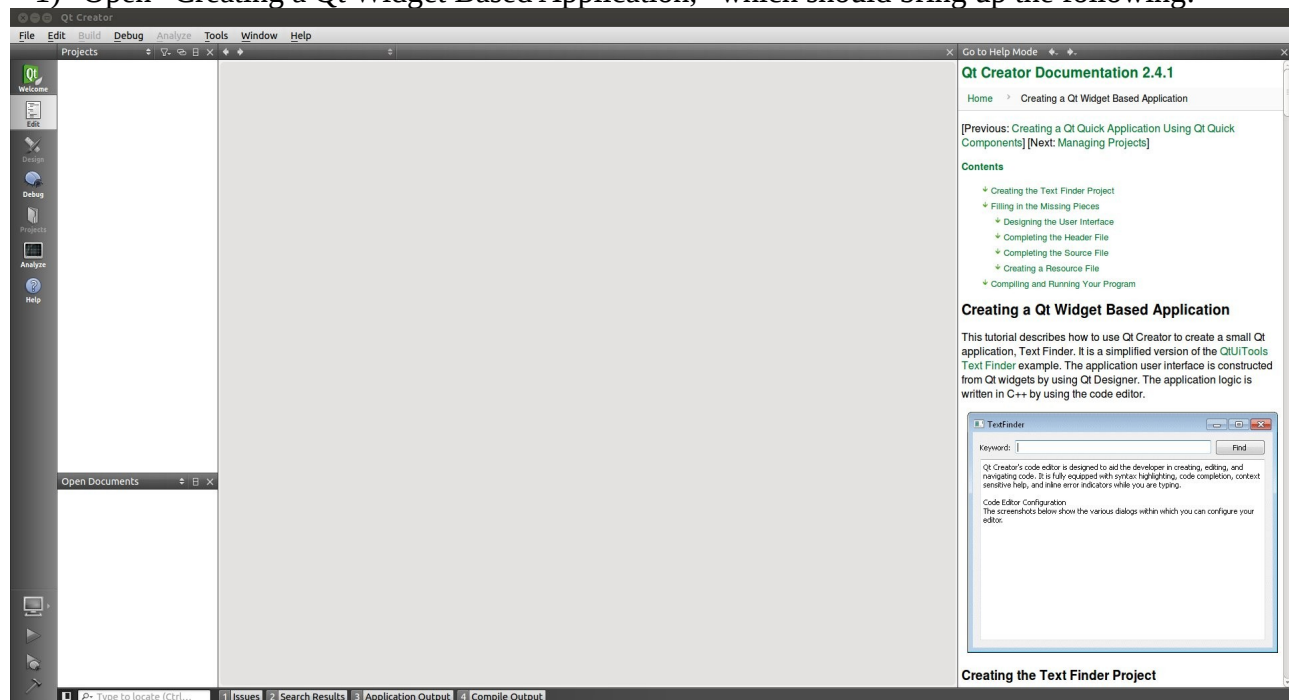
Wyatt Newman

October, 2014

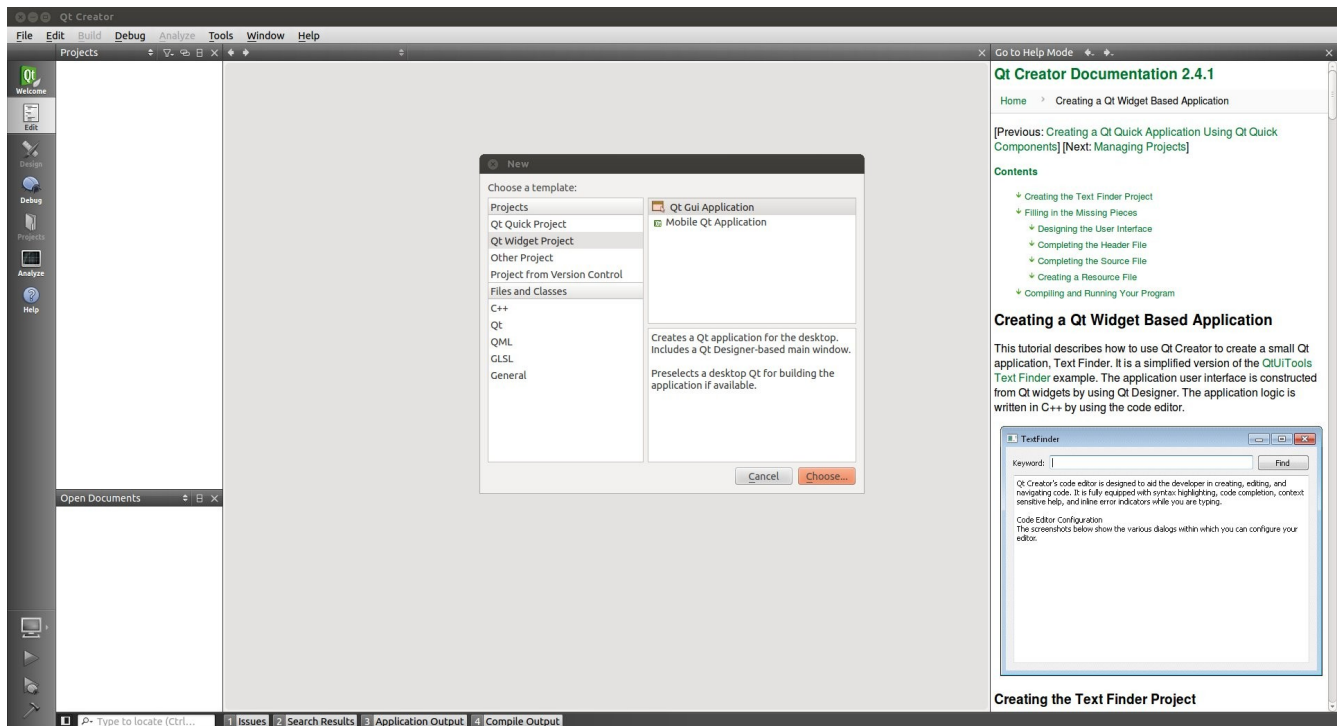
Designing a graphical interface is commonly separated into the design of the graphics and the design of the callback code that responds to “widgets” in the graphical interface. This document describes using the application “QT Creator” to design the graphical aspects of the interface. A separate document, “How to Create a Python GUI in ROS”, details how to write a ROS package with callbacks that connect to your graphical design.

To begin, start up QT Creator, which you should find under: Applications->Programming. Once this program has started up, do the following:

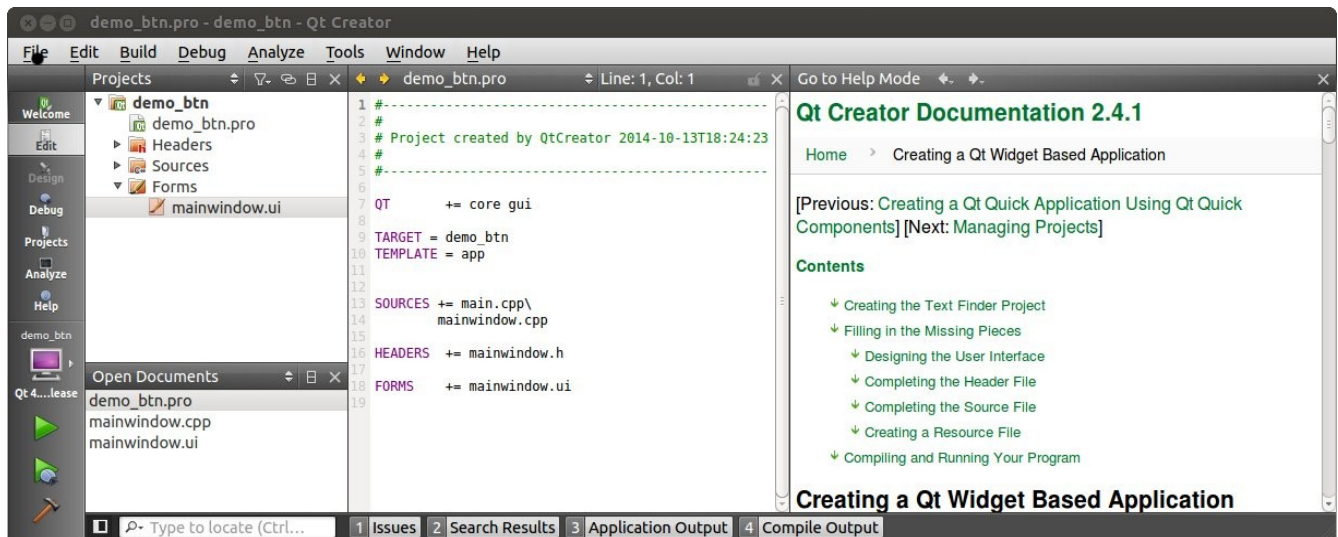
1) Open “Creating a Qt Widget Based Application,” which should bring up the following:



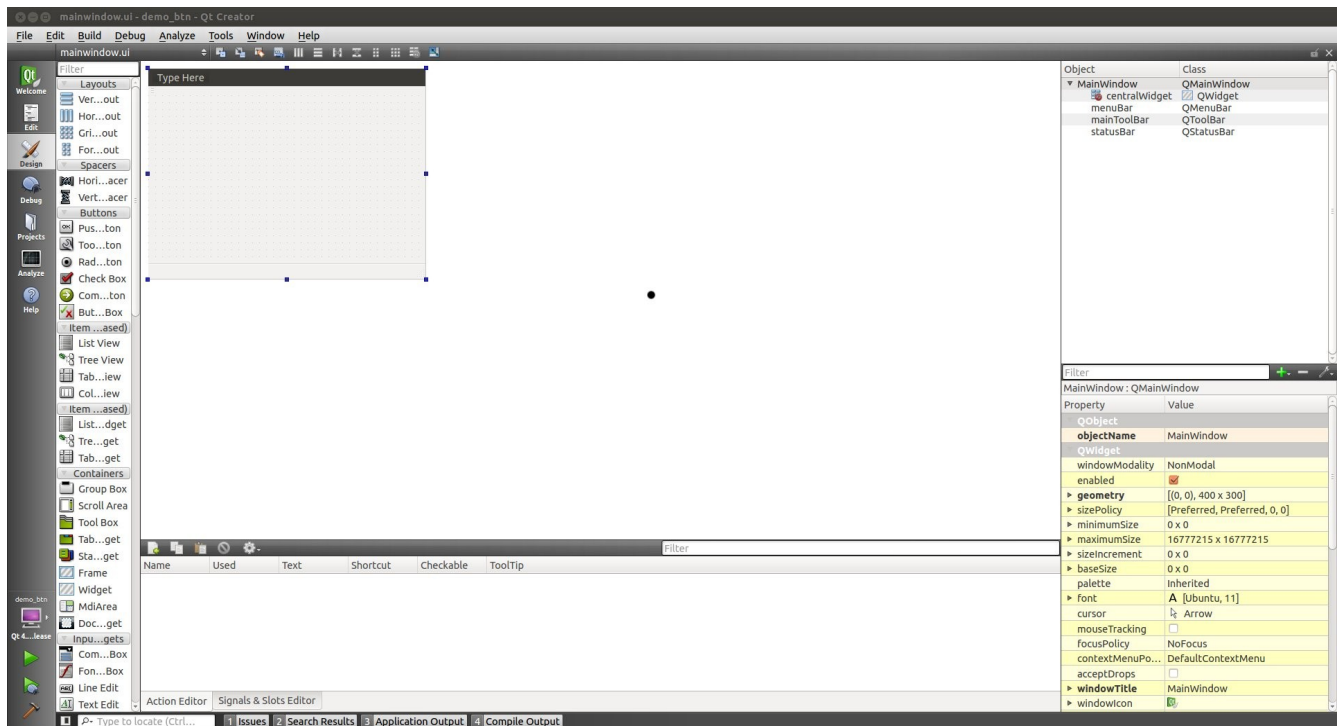
2) From the menu, choose: File/New File or Project..., which should bring up a pop-up titled “New.” In this pop-up, select “Qt Widget Project” and “Qt Gui Application”



3) You will be prompted to fill in a project name and directory. Do so, then accept the defaults (click “next” until done with the prompts). You should see a view something like the following:

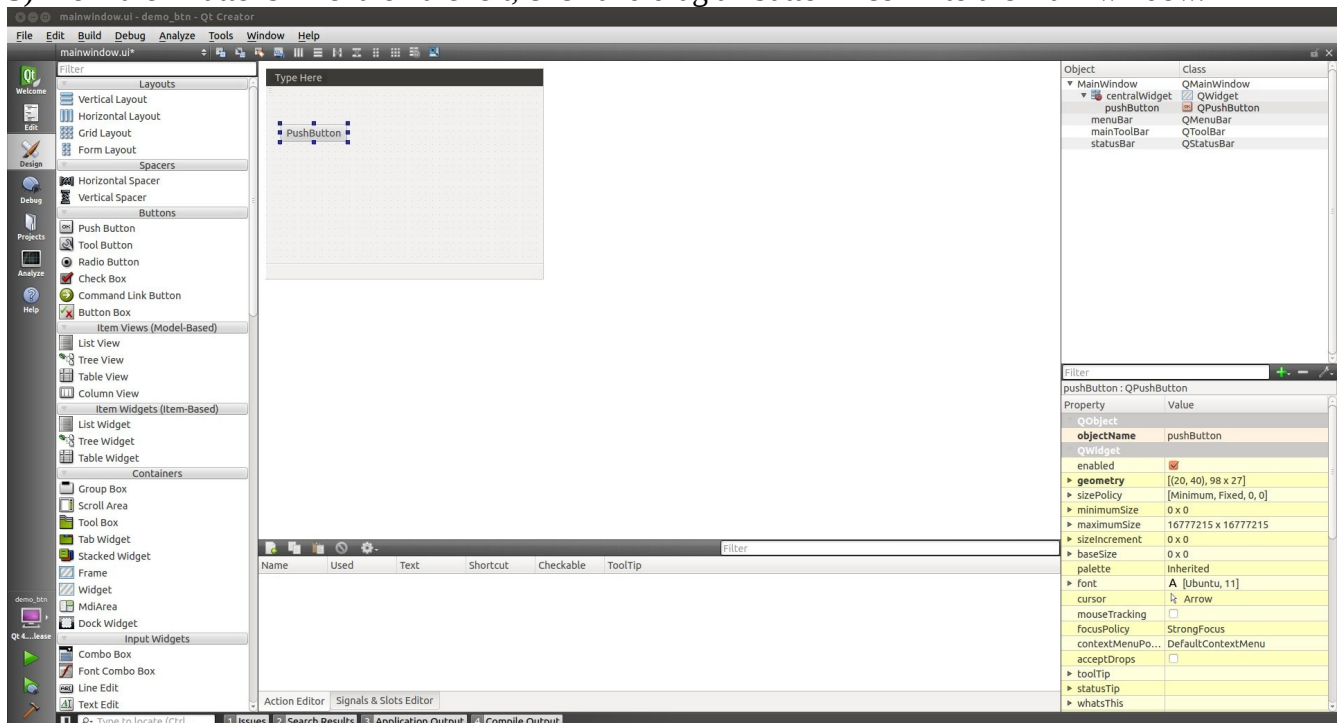


4) From the above view, within the “Projects” window, expand “Forms” and click on “mainwindow.ui”. Your view should switch to the scene below.



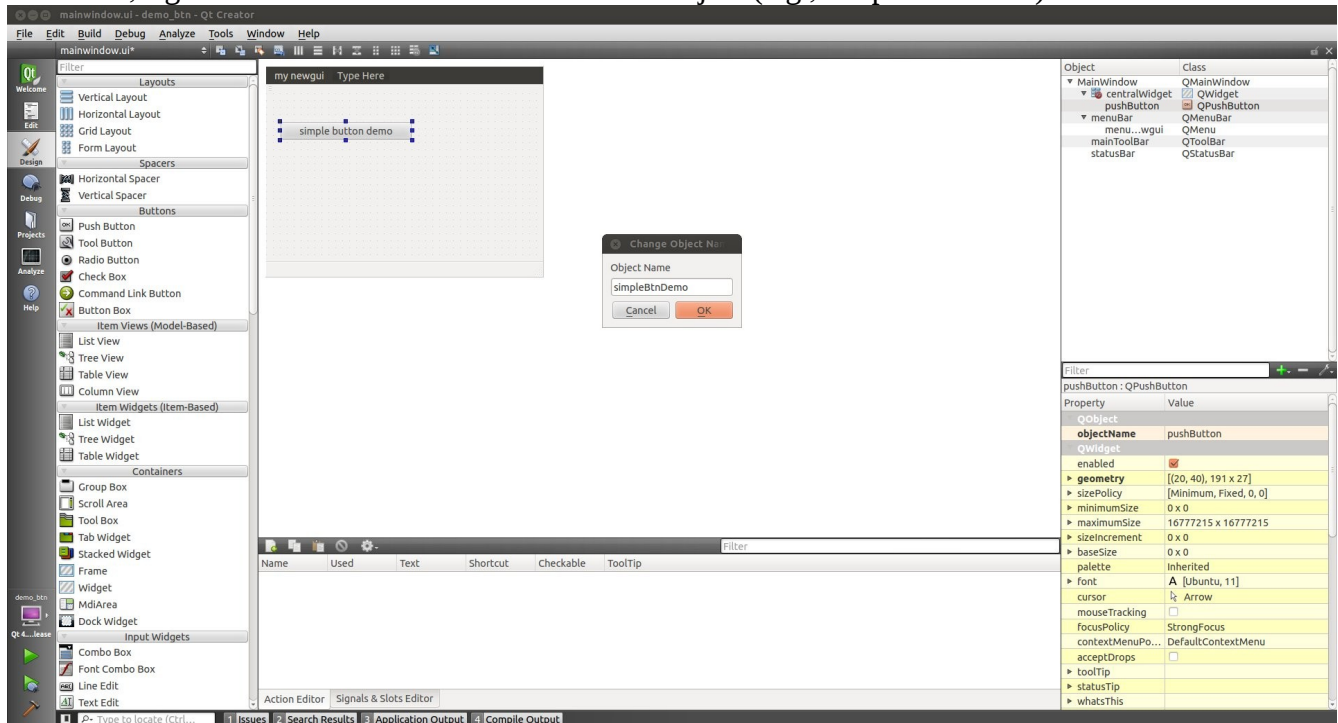
This will place you in a mode in which you can design the graphical aspects of your GUI.

5) From the “Buttons” menu on the left, click and drag a “button” icon into the main window.

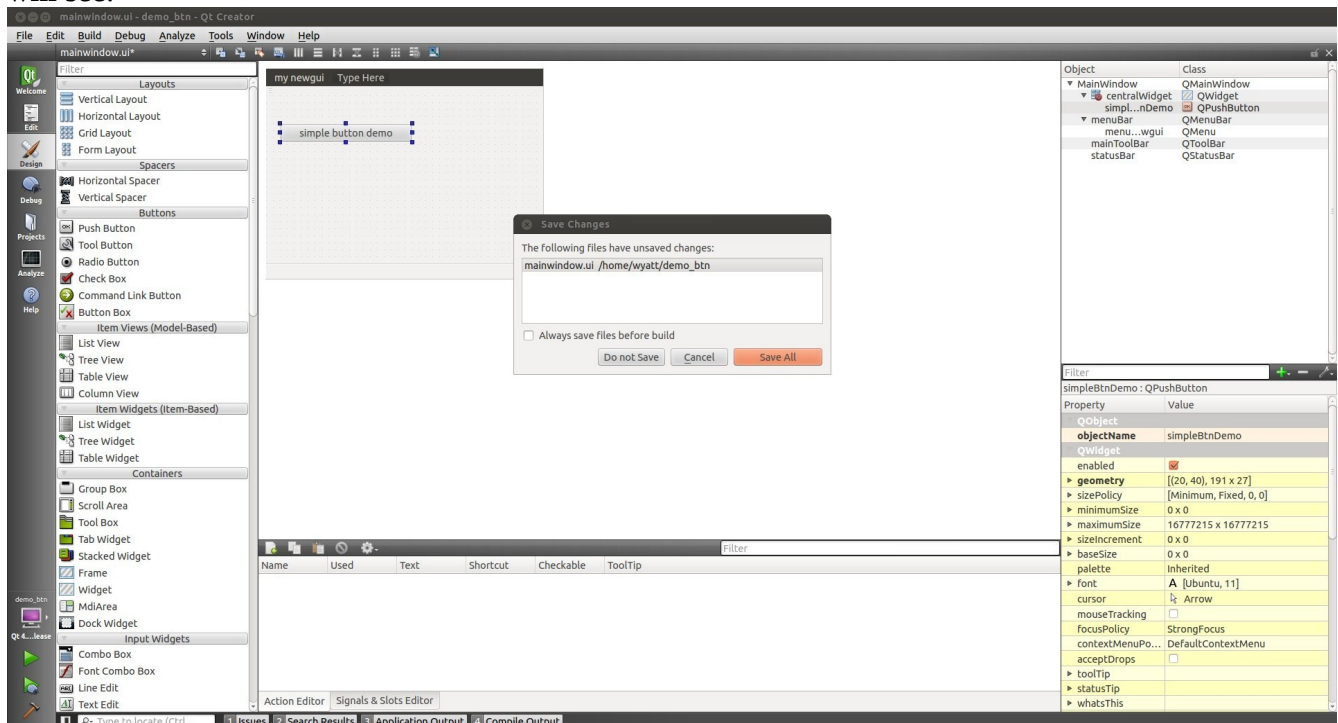


6) As desired, edit the title bar, and right-click on the button icon to edit its name. Resize the button, if necessary/desired.

If desired, right-click on the button and rename the object (e.g., simpleBtnDemo)

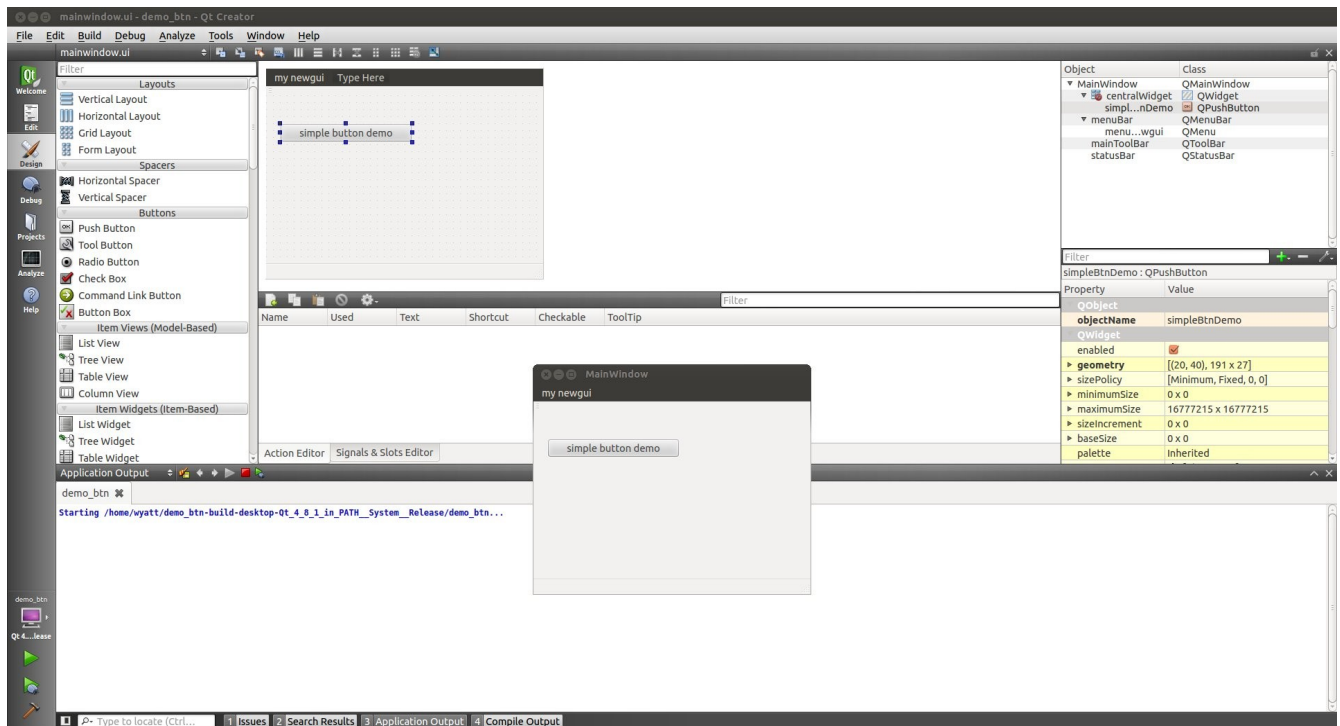


7) On the lower left, hover over the green triangle to see the hint “Run...”, then click this button. You will see:



8) Click “Save All”.

Your new GUI design should pop up.

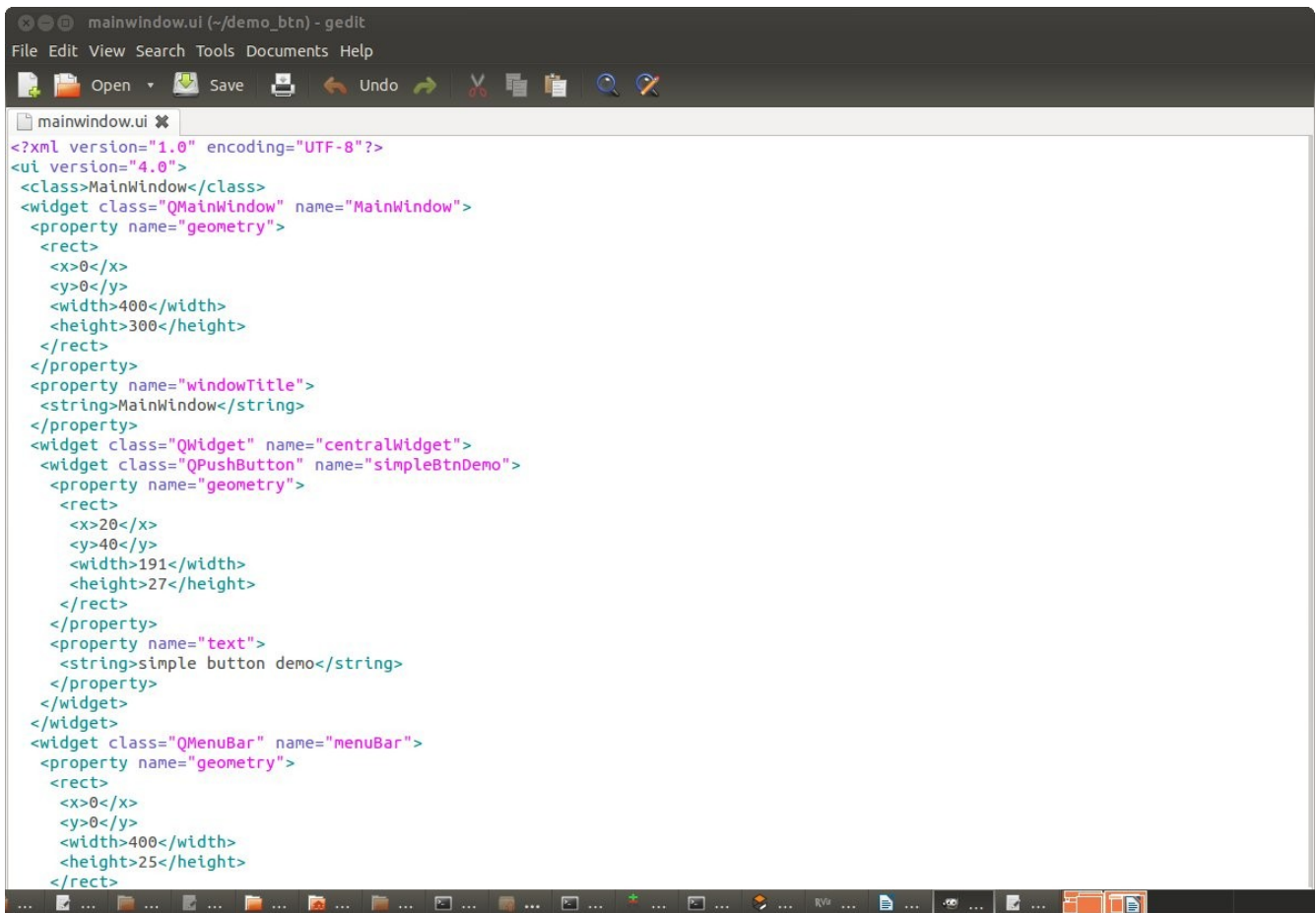


This completes this simple example of designing a graphical interface. You can close QT Creator.

Examining the UI file:

Navigate to the project folder (named in step 3 above). You will find a file by the name: mainwindow.ui. This is the file we will need to complete your GUI design.

If you open this file with an editor, you will see XML code, a screenshot of which follows:



```
mainwindow.ui (~/.demo_btn) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
mainwindow.ui
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>400</width>
        <height>300</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>MainWindow</string>
    </property>
    <widget class="QWidget" name="centralWidget">
      <widget class="QPushButton" name="simpleBtnDemo">
        <property name="geometry">
          <rect>
            <x>20</x>
            <y>40</y>
            <width>191</width>
            <height>27</height>
          </rect>
        </property>
        <property name="text">
          <string>simple button demo</string>
        </property>
      </widget>
    </widget>
    <widget class="QMenuBar" name="menuBar">
      <property name="geometry">
        <rect>
          <x>0</x>
          <y>0</y>
          <width>400</width>
          <height>25</height>
        </rect>
      </property>
    </widget>
  </widget>
</ui>
```

Note that there is a widget of type QPushButton with name “simpleBtnDemo”. (This is the name that was chosen in step 6).

You will need to refer to this name when writing callback functions that interact with this GUI.

The file “mainwindow.ui” should be copied to your /ui subdirectory in the GUI package that you will build. (See “How to Create a Python GUI in ROS”).

In this very elementary example, we have seen how to create a simple button. This is easily extended to alternative I/O and adding many more features. Each new widget added will get its own name in the *.ui file, and callback code should be written to respond to the functionality of these elements.