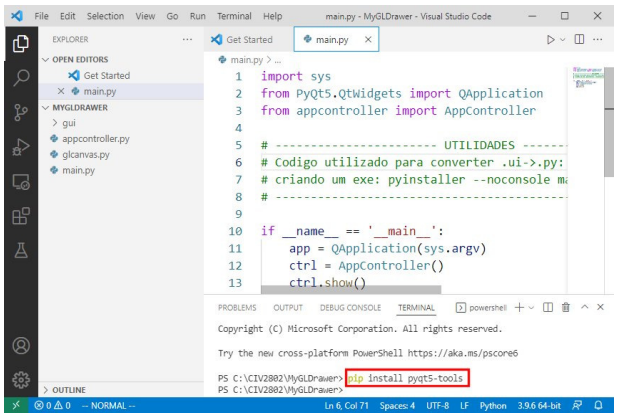
CIV 2802 – Sistemas Gráficos para Engenharia – PUC-Rio

Roteiro para criação de um programa simples para desenho 2D em Python usando Qt 5 e OpenGL

1. Instale o pacote pyqt5-tools

No Visual Studio Code:

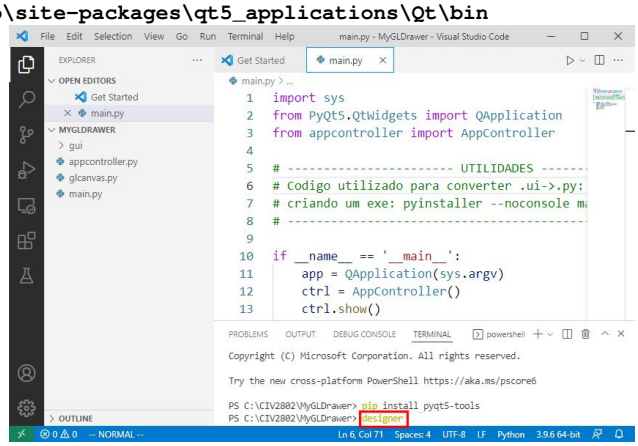
```
pip install pyqt5-tools
```



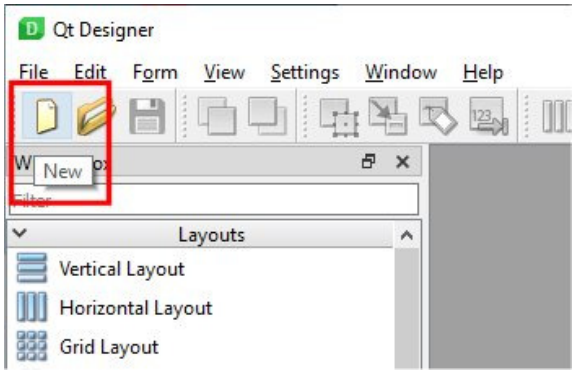
2. Abra o aplicativo Qt Designer

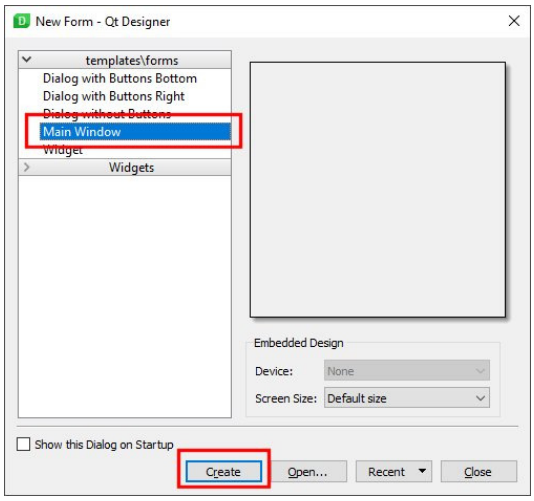
Antes adicione essa pasta no "path" do usuário:

```
C:\Users\<user>\AppData\Local\Programs\Python\Python39\
Lib\site-packages\qt5_applications\Qt\bin
```

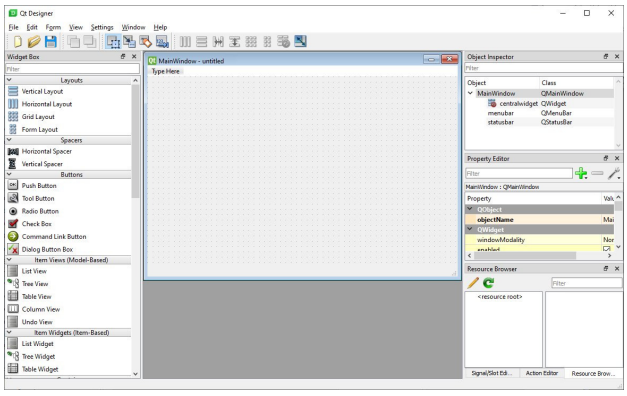


3. Execute o programa QtDesigner e crie uma nova Qt Main Window

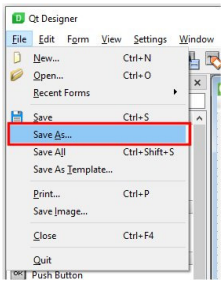


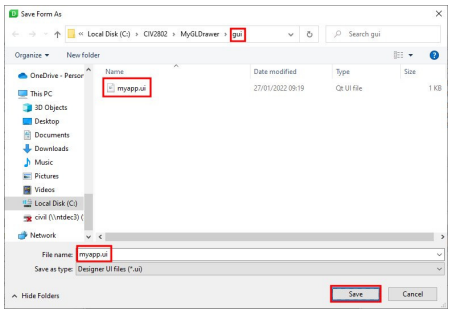


4. Pronto para criar interface gráfica

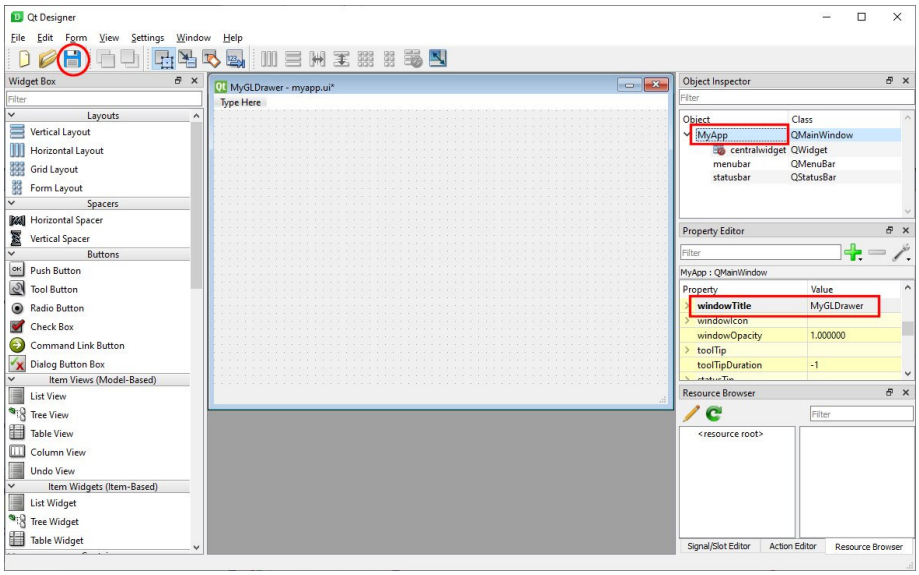


5. Salve arquivo com nome myapp.ui

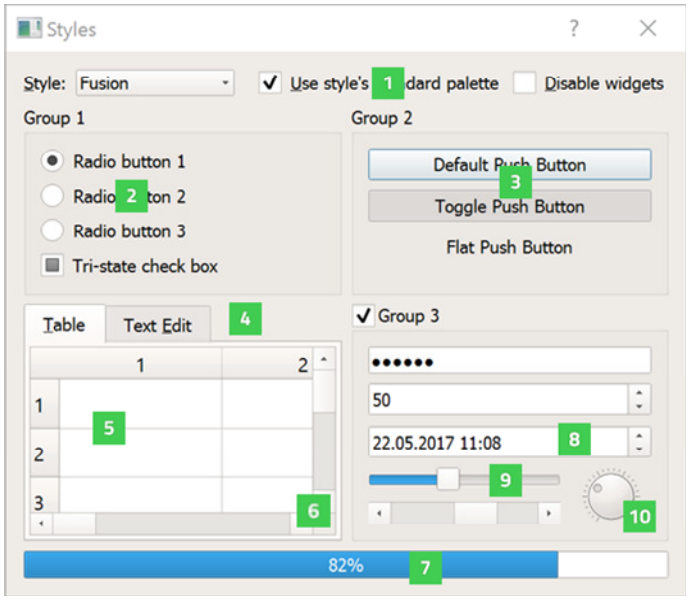




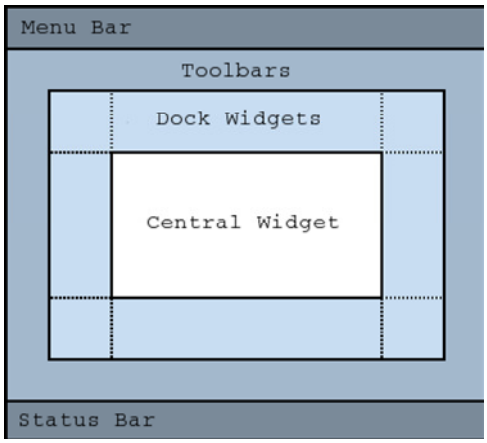
6. Muda o nome do objeto da janela principal e muda o seu título (salve)



QWidget

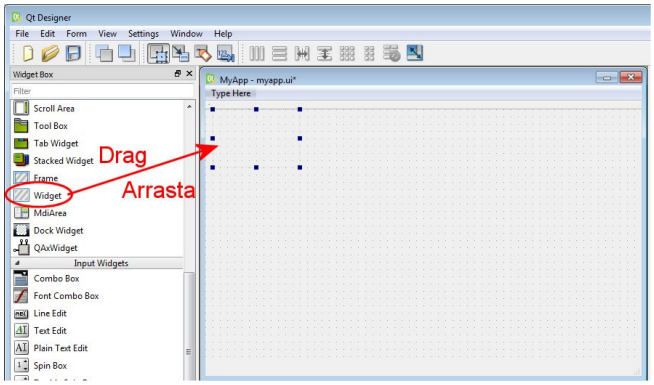


QMainWindow

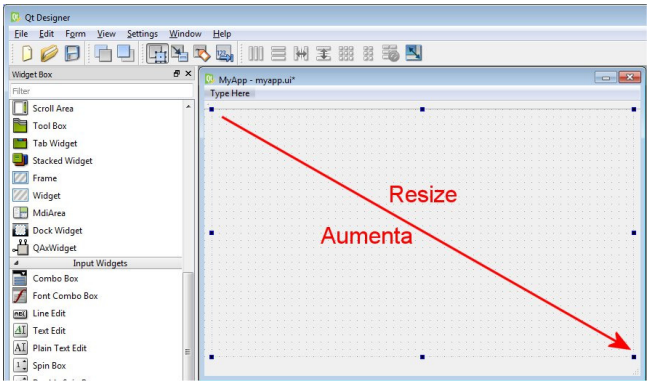


- QCheckBox (1) provides a checkbox with a text label.
- QRadioButton (2) provides a radio button with a text or pixmap label.
- QPushButton (3) provides a command button.
- QTabWidget (4) provides a stack of tabbed widgets.
- QTableWidget (5) provides a classic item-based table view.
- QScrollBar (6) provides a vertical or horizontal scroll bar.
- QProgressBar (7) provides a horizontal progress bar.
- QDateTimeEdit (8) provides a widget for editing dates and times.
- QSlider (9) provides a vertical or horizontal slider.
- QDial (10) provides a rounded range control (like a speedometer or potentiometer).

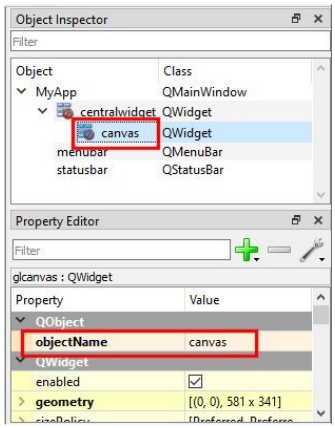
7. Crie um widget para o canvas



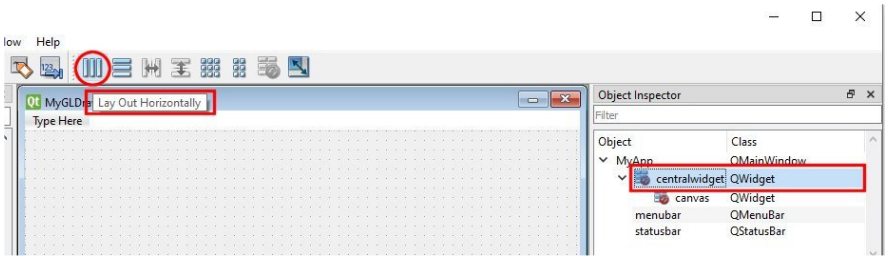
8. Faça o canvas ocupar toda a janela



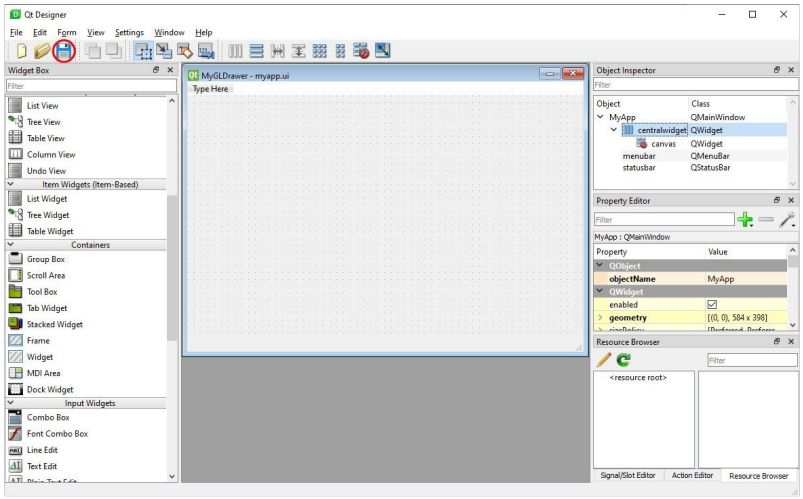
9. Mude o nome do objeto widget do canvas



10. Force um layout horizontal para o centralWidget



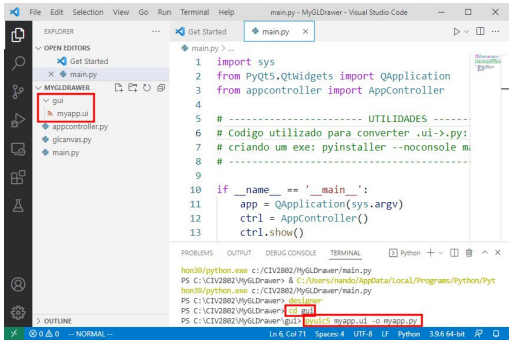
11. Salve o arquivo myapp.ui

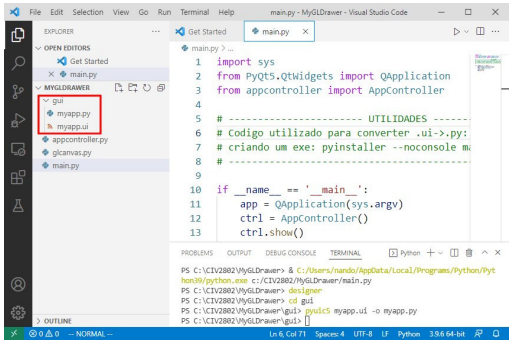



```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MyApp</class>
 <widget class="QMainWindow" name="MyApp">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>584</width>
    <height>398</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MyGLDrawer</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QHBoxLayout" name="horizontalLayout">
    <item>
     <widget class="QWidget" name="canvas" native="true"/>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>584</width>
     <height>21</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

12. Crie o arquivo myapp.py

```
cd gui
pyuic5 myapp.ui -o myapp.py
```





```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'myapp.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MyApp(object):
    def setupUi(self, MyApp):
        MyApp.setObjectName("MyApp")
        MyApp.resize(584, 398)
        self.centralwidget = QtWidgets.QWidget(MyApp)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.canvas = QtWidgets.QWidget(self.centralwidget)
        self.canvas.setObjectName("canvas")
        self.horizontalLayout.addWidget(self.canvas)
        MyApp.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MyApp)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 584, 21))
        self.menubar.setObjectName("menubar")
        MyApp.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MyApp)
        self.statusbar.setObjectName("statusbar")
        MyApp.setStatusBar(self.statusbar)

        self.retranslateUi(MyApp)
        QtCore.QMetaObject.connectSlotsByName(MyApp)

    def retranslateUi(self, MyApp):
        _translate = QtCore.QCoreApplication.translate
        MyApp.setWindowTitle(_translate("MyApp", "MyGLDrawer"))
```

(não modifique esse arquivo)

13. Arquivo appcontroller.py (classe AppController herda de QMainWindow e Ui_MyApp)

```
from PyQt5.QtWidgets import QMainWindow, QHBoxLayout
from gui.myapp import Ui_MyApp
from glcanvas import GLCanvas

class AppController(QMainWindow, Ui_MyApp):
    def __init__(self):
        super().__init__()
        super().setupUi(self)

        # Create an OpenGL canvas and associate to the
        # canvas widget
        self.glcanvas = GLCanvas()
        self.glcanvas.setParent(self.canvas)
        horizontalLayout = QHBoxLayout(self.canvas)
        horizontalLayout.setContentsMargins(0, 0, 0, 0)
        horizontalLayout.setSpacing(0)
        horizontalLayout.addWidget(self.glcanvas)
```

14. Arquivo glcanvas.py

```
from PyQt5 import QtOpenGL
from PyQt5.QtWidgets import *
from OpenGL.GL import *

class GLCanvas(QtOpenGL.QGLWidget):

    def __init__(self):
        super(GLCanvas, self).__init__()
        self.width = 0    # width of canvas (horizontal raster size)
        self.height = 0   # height of canvas (vertical raster size)
        self.left = 0.0   # left limit of clipping window (world)
        self.right = 0.0  # right limit of clipping window (world)
        self.bottom = 0.0 # bottom limit of clipping window (world)
        self.top = 0.0    # top limit of clipping window (world)

    def initializeGL(self):
        glClearColor(1.0, 1.0, 1.0, 1.0) # white color
        glClear(GL_COLOR_BUFFER_BIT)

    def resizeGL(self, _width, _height):
        # store GL canvas sizes in object properties
        self.width = _width
        self.height = _height
        # setup the viewport to canvas dimensions
        glViewport(0, 0, self.width, self.height)
        # reset the coordinate system
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        # establish the clipping window in world model coordinates
        # by setting up an orthographic projection
        self.left = 0.0
        self.right = float(self.width-1)
        self.bottom = 0.0
        self.top = float(self.height-1)
        glOrtho(self.left, self.right, self.bottom, self.top, -1.0, 1.0)
        # setup display in model coordinates
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()

    def paintGL(self):
        # clear the buffer with the current clear color
        glClear(GL_COLOR_BUFFER_BIT)
        # get clipping window sizes in world model coordenates
        wWorld = self.right-self.left # width of clipping window (world)
        hWorld = self.top-self.bottom # height of clipping window (world)
        # draw a triangle, centered at the clipping window, with RGB color
        # at the 3 vertices interpolating smoothly the color in the interior
        glShadeModel(GL_SMOOTH)
        xA = self.left   + wWorld / 3.0
        yA = self.bottom + hWorld / 3.0
        xB = self.left   + wWorld * (2.0 / 3.0)
        yB = self.bottom + hWorld / 3.0
        xC = self.left   + wWorld / 2.0
        yC = self.bottom + hWorld * (2.0 / 3.0)
        glBegin(GL_TRIANGLES)
        glColor3f(1.0, 0.0, 0.0) # red
        glVertex2f(xA, yA)
        glColor3f(0.0, 1.0, 0.0) # green
        glVertex2f(xB, yB)
        glColor3f(0.0, 0.0, 1.0) # blue
        glVertex2f(xC, yC)
        glEnd()
```

15. Arquivo main.py

```
import sys
from PyQt5.QtWidgets import QApplication
from appcontroller import AppController

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ctrl = AppController()
    ctrl.show()
    app.exec_()
```

16. Execução