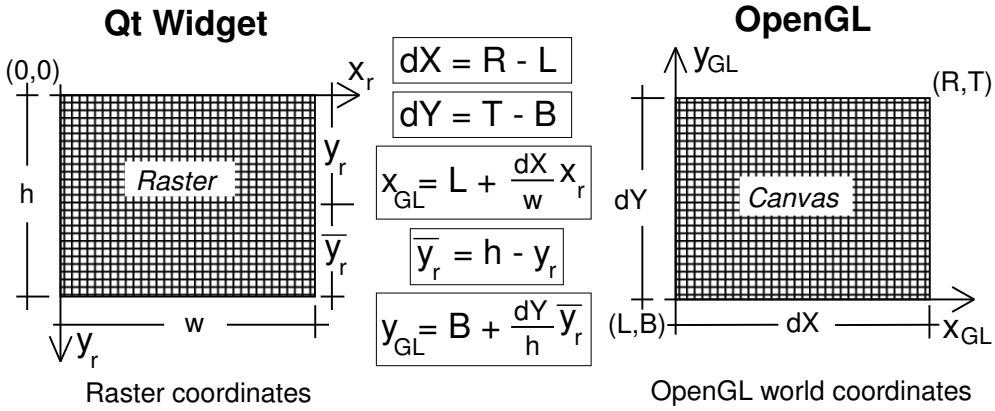
CIV 2802 – Sistemas Gráficos para Engenharia – PUC-Rio

Programa simples em Python para entendimento de eventos de mouse em canvas OpenGL usando Qt:

http://www.tecgraf.puc-rio.br/ftp_pub/lfm/CIV2802-CanvasMouseEvent-Python.zip



glcanvas.py

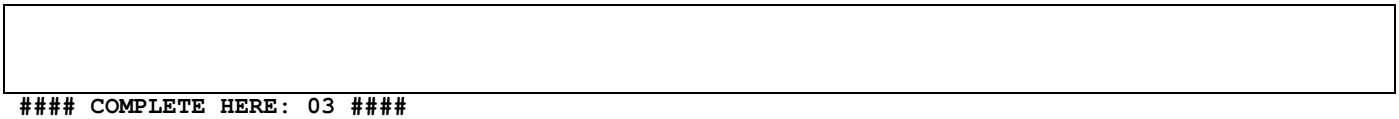
```
from PyQt5 import QtOpenGL, QtCore
from PyQt5.QtWidgets import *
from OpenGL.GL import *

class GLCanvas(QtOpenGL.QGLWidget):

    def __init__(self, _controller=[], _view=[]):
        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 = 12.0 # right limit of clipping window (world)
        self.bottom = 0.0 # bottom limit of clipping window (world)
        self.top = 10.0   # top limit of clipping window (world)

        # mouse button that was pressed
        # (QCore.Qt.LeftButton, QCore.Qt.RightButton QCore.Qt.MidButton)
        self.mouseButton = QtCore.Qt.NoButton
        self.mousebuttonPressed = False  # if true, mouse button is pressed
        self.pt0 = QtCore.QPoint(0,0)  # mouse position at button press event
        self.pt1 = QtCore.QPoint(0,0)  # current mouse position

    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # --------------------CANVAS PREDEFINED SLOTS-----------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------

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

        # set some graphics primitives attributes
        glEnable(GL_LINE_SMOOTH)
        glLineWidth(0.5)
        glPointSize(6.0)

    def resizeGL(self, _width, _height):
        # store GL canvas sizes in object properties
        self.width = _width
        self.height = _height
```

```
        # Setup the viewport to canvas dimesions
        glViewport(0, 0, self.width, self.height)

        # reset the coordinate system
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()

        # Establish the clipping volume by setting up an
        # orthographic projection
        glOrtho(self.left, self.right, self.bottom, self.top, -1.0, 1.0)

        # Setup display model in model coordinates
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()

    def paintGL(self):
        # Disregard display if first point and current point
        # are at the same position
        if (self.pt0.x() == self.pt1.x()) and (self.pt0.y() == self.pt1.y()):
            return

        # clear the buffer with the current color
        glClear(GL_COLOR_BUFFER_BIT)

        # convert first and current points from raster to world coordinates
        pt0W = self.convertRasterPtToWorldCoords(self.pt0)
        pt1W = self.convertRasterPtToWorldCoords(self.pt1)

        # draw a red line from button press point to current point
        glColor3f(1.0, 0.0, 0.0)  # red
        glBegin(GL_LINE_STRIP)
        glVertex2f(pt0W.x(), pt0W.y())
        glVertex2f(pt1W.x(), pt1W.y())
        glEnd()

        # draw black marks at first and current points
        glColor3f(0.0, 0.0, 0.0)  # black
        glBegin(GL_POINTS)
        glVertex2f(pt0W.x(), pt0W.y())
        glVertex2f(pt1W.x(), pt1W.y())
        glEnd()

    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # --------FUNCTION TO CONVERT POINT FROM RASTER TO WORLD COORDS--------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------

    def convertRasterPtToWorldCoords(self, _pt):
        dX = self.right - self.left  # World window horizontal size
        dY = self.top - self.bottom  # World window vertical size

        # Origin of canvas raster coordinates it at the left-top corner,
        # while origin of GL canvas floating point coordinates is at
        # left-bottom corner.
        # mX is the distance of point to left universe window limit
        # mY is the distance of point to bottom universe window limit
#### COMPLETE HERE: 01 ####
```

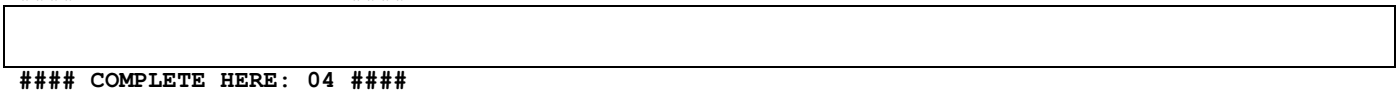
```
#### COMPLETE HERE: 01 ####
        return QtCore.QPointF(x, y)
```

```
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ----------------------MOUSE EVENT SLOTS---------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------
    # ------------------------------------------------------------------

    def mousePressEvent(self, event):

        # Ignore event if not mouse left button
        self.mouseButton = event.button()
        if self.mouseButton != QtCore.Qt.LeftButton:
            return

        # Set flag for mouse button presses and
        # get current button pressed mouse position
#### COMPLETE HERE: 02 ####
```

```
#### COMPLETE HERE: 02 ####

    def mouseMoveEvent(self, event):

        # Get current mouse position if mouse button is pressed
#### COMPLETE HERE: 03 ####
```



```
#### COMPLETE HERE: 03 ####

        # force redisplay
        self.update()

    def mouseReleaseEvent(self, event):

        # Reset flag for mouse button presses and
        # get current button pressed mouse position
#### COMPLETE HERE: 04 ####
```



```
#### COMPLETE HERE: 04 ####

        # force redisplay
        self.update()
```