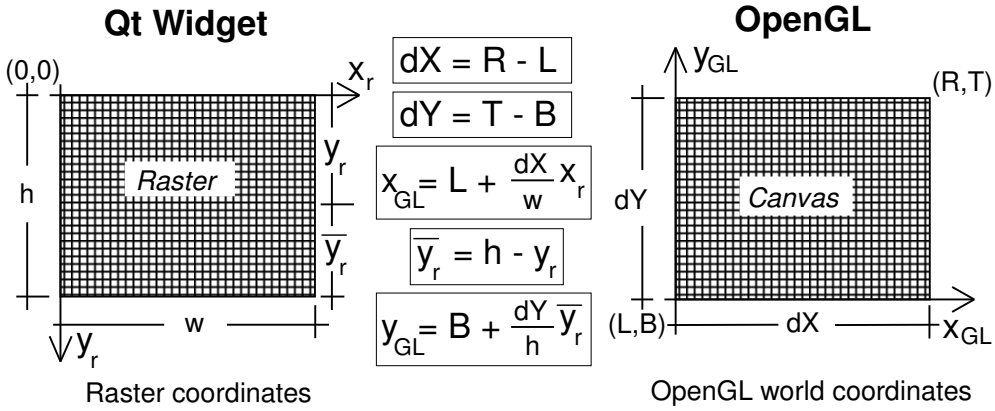
CIV 2802 – Sistemas Gráficos para Engenharia – PUC-Rio

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

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



glcanvas.h

```
#ifndef GLCANVAS_H
#define GLCANVAS_H

#include <QMouseEvent>
#include <QtOpenGL/QGLWidget>

class GLCanvas : public QGLWidget
{
  Q_OBJECT // must include this if you use Qt Signals & Slots

 public:
  GLCanvas(QWidget *parent = 0);
  ~GLCanvas();

 protected:
  // Canvas predefined slots
  void initializeGL();
  void resizeGL(int _width, int _height);
  void paintGL();

 protected:
  int  m_w; // width: GL canvas horizontal size
  int  m_h; // height: GL canvas vertical size
  GLdouble m_left;     // left limit of object space window
  GLdouble m_right;    // right limit of object space window
  GLdouble m_bottom;   // bottom limit of object space window
  GLdouble m_top;      // top limit of object space window

  // Mouse point properties
  bool m_buttonPressed;     // if true, mouse button is pressed
  // QPoint is a Qt class that defines a point in a plane
  // i.e. the position of the mouse on the canvas.
  QPoint m_pt0;             // mouse position at button press event
  QPoint m_pt1;             // current mouse position

  // QPointF is a Qt class that defines a point in floating
  // point coordinates.
  QPointF convertPtCoordsToUniverse(QPoint _pt);

 public slots:
  // Mouse events slots
  void mousePressEvent(QMouseEvent* event);
  void mouseMoveEvent(QMouseEvent* event);
  void mouseReleaseEvent(QMouseEvent *event);
};

#endif // GLCANVAS_H
```

glcanvas.cpp

```
#include "glcanvas.h"

//------------------------------------------------------------------------
GLCanvas::GLCanvas(QWidget *parent)
: QGLWidget(parent)
{
 m_pt0.setX(0);
 m_pt0.setY(0);
 m_pt1.setY(0);
 m_pt1.setY(0);
 m_buttonPressed = false;
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
GLCanvas::~GLCanvas()
{
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
void
GLCanvas::initializeGL()
{
 // set white as background color and clear window
 glClearColor(1.0, 1.0, 1.0, 1.0);
 glClear(GL_COLOR_BUFFER_BIT);

 // enable smooth line display
 glEnable(GL_LINE_SMOOTH);
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
void
GLCanvas::resizeGL(int _width, int _height)
{
 // store GL canvas sizes in object properties
 m_w = _width;
 m_h = _height;

 // setup the viewport to canvas dimensions
 glViewport(0, 0, (GLint)m_w, (GLint)m_h);

 // setup world limit coordinates based on canvas sizes
 m_left = 0.0;
 m_right = (double)m_w/10.0;
 m_bottom = 0.0;
 m_top = (double)m_h/10.0;

 // reset the coordinate system
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();

 // establish the clipping volume by setting up an
 // orthographic projection
 glOrtho(m_left, m_right, m_bottom, m_top, -1.0, 1.0);

 // setup display in model coordinates
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
void
GLCanvas::paintGL()
{
 // clear the buffer with the current clear color
 glClear(GL_COLOR_BUFFER_BIT);
```

```
 // draw a red line from button press point to current point
 glColor3f(1.0, 0.0, 0.0);  // red
 QPointF pt0_U = convertPtCoordsToUniverse(m_pt0);
 QPointF pt1_U = convertPtCoordsToUniverse(m_pt1);
 glBegin(GL_LINE_STRIP);
 glVertex2f(pt0_U.x(), pt0_U.y());
 glVertex2f(pt1_U.x(), pt1_U.y());
 glEnd();
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
QPointF
GLCanvas::convertPtCoordsToUniverse(QPoint _pt)
{
 double dX = m_right - m_left; // universe window horizontal size
 double dY = m_top - m_bottom; // universe 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 QPointF(x,y);
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
//                       PUBLIC SLOTS
//------------------------------------------------------------------------

//------------------------------------------------------------------------
void
GLCanvas::mousePressEvent(QMouseEvent* event)
{
/*** COMPLETE HERE 02 ***/


/*** COMPLETE HERE 02 ***/
}

//------------------------------------------------------------------------
void
GLCanvas::mouseMoveEvent(QMouseEvent* event)
{
/*** COMPLETE HERE 03 ***/



/*** COMPLETE HERE 03 ***/
 update();
}
//------------------------------------------------------------------------

//------------------------------------------------------------------------
void
GLCanvas::mouseReleaseEvent(QMouseEvent *event)
{
/*** COMPLETE HERE 04 ***/


/*** COMPLETE HERE 04 ***/
 update();
}
//------------------------------------------------------------------------
```