

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)**

Лабораторная работа №6
дисциплина «Компьютерная графика »
по теме «Знакомство с библиотекой OpenGL»

Выполнил: студент группы ВТ-31
Проверил:

Макаров Д.С.
Осипов О.В.

Белгород 2019

Лабораторная работа №6

«Знакомство с библиотекой OpenGL»

Цель работы: изучение функций OpenGL для построения трёхмерных объектов в среде Qt Creator..

Вариант 9

Требования к программе

1. Графические объекты должны быть изображены на экране в виде набора закрашенных полигонов. В памяти объекты необходимо хранить в виде массива многоугольников. На сцене должно быть не менее 10 различных объектов.
2. Предоставить пользователю возможность перемещать, поворачивать, масштабировать объекты сцены с использованием клавиш и мыши, а также изменять положение камеры (наблюдателя).

Ход работы

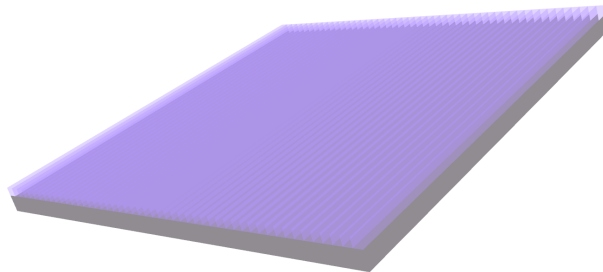


Рис. 1: Пример работы программы

Приложение

Содержимое файла main.py

```
import sys
import math

from PyQt5.QtCore import Qt, QObject, pyqtSignal, QPoint
from PyQt5.QtGui import QColor, QMatrix4x4
from PyQt5.QtWidgets import QApplication, QMessageBox
from PyQt5.QtOpenGL import QGL, QGLFormat, QGLWidget
from PIL.Image import open

from blockmap import p_map
try:
    from OpenGL import GL, GLU
except ImportError:
    app = QApplication(sys.argv)
    QMessageBox.critical(None, "OpenGL samplebuffers",
        "PyOpenGL must be installed to run this example.")
    sys.exit(1)

class GLWidget(QGLWidget):
    GL_MULTISAMPLE = 0x809D
    rot = 0.0
    def __init__(self, parent):
        super(GLWidget, self).__init__(QGLFormat(QGL.SampleBuffers), parent)
        self.setWindowTitle("OpenGL")
        self.globalScale = 0.1
        self.rotationMatrix = QMatrix4x4()
        self.centerMatrix = QMatrix4x4()
        self.pos = QPoint(0,0)
        self.z_offset = -10

    def changeRotationMatrix(self, dx, dy):
        self.rotationMatrix.rotate(-dx, 0, 1, 0)
        self.rotationMatrix.rotate(-dy, 1, 0, 0)

    def centredScene(self, count_x, count_y, count_z):
        self.centerMatrix.setToIdentity()
        self.centerMatrix.translate(-count_x/2, 0, 0)
        self.centerMatrix.translate(0, -count_y/2, 0)
        self.centerMatrix.translate(0, 0, -count_z/2)

    def mouseMoveEvent(self, event):
        newPos = QPoint(event.pos())
        dx = newPos.x() - self.pos.x()
        dy = newPos.y() - self.pos.y()
        self.changeRotationMatrix(dx / 2, dy / 2)
        self.pos = newPos
        self.resetModelView()
        self.repaint()

    def mousePressEvent(self, event):
        self.pos = event.pos()
        self.repaint()

    def resetProjection(self):
        GL.glMatrixMode(GL.GL_PROJECTION)
        GL.glLoadIdentity()
```

```

GLU.gluPerspective(30.0, self.width() / self.height(), 0.1, 20)

def resetModelView(self):
    GL.glMatrixMode(GL.GL_MODELVIEW)
    GL.glLoadIdentity()
    GL.glTranslatef(0, 0, self.z_offset)
    GL.glMultMatrixf(self.rotationMatrix.transposed().data())
    GL.glScalef(self.globalScale, self.globalScale, self.globalScale)
    GL.glMultMatrixf(self.centerMatrix.data())

def wheelEvent(self, event):
    numPixels = QPoint(event.pixelDelta())
    numDegrees = QPoint(event.angleDelta() / 8)
    if (not numPixels.isNull()):
        self.globalScale = self.globalScale -
            ↪ (event.pixelDelta().x()+event.pixelDelta().y())/600
    elif (not numDegrees.isNull()):
        self.globalScale = self.globalScale - ((numDegrees.x()+numDegrees.y()) / 15)/600
    self.resetProjection()
    self.resetModelView()
    self.repaint()

def initializeGL(self):
    GL.glClearColor(1.0, 1.0, 1.0, 0.0)
    GL.glEnable(GL.GL_DEPTH_TEST)
    GL.glEnable(GL.GL_NORMALIZE)
    GL.glShadeModel(GL.GL_SMOOTH)
    #GL.glEnable(GL.GL_MULTISAMPLE)
    #обрезка внутренних
    GL.glEnable(GL.GL_CULL_FACE)
    GL.glMatrixMode(GL.GL_PROJECTION)
    #альфа канал
    GL.glEnable(GL.GL_BLEND)
    GL.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA)
    #отрисовка внутренней стороны полигона
    self.makeObject()

def resizeGL(self, w, h):
    GL.glViewport(0, 0, w, h)
    self.resetProjection()

def paintGL(self):
    GL.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT)
    GL.glMatrixMode(GL.GL_MODELVIEW)
    GL.glEnable(GLWidget.GL_MULTISAMPLE)
    self.makeObject()

def makeObject(self):
    for index_z, z in enumerate(p_map):
        for index_y, y in enumerate(z):
            for index_x, x in enumerate(y):
                self.geometry(x, index_x, index_y, index_z)
    self.centredScene(len(p_map[0][0]), len(p_map[0]), len(p_map))

def geometry(self, Block, x, y, z):
    block_color = QColor(
        Block.material.color_red,
        Block.material.color_green,
        Block.material.color_blue,
        Block.material.color_alpha
    )

```

```

)

self.qglColor(block_color)
GL.glBegin(GL.GL_QUADS)
#нижний полигон
GL.glVertex3d(x, y, z)
GL.glVertex3d(1+x, y, z)
GL.glVertex3d(1+x, y, 1+z)
GL.glVertex3d(x, y, 1+z)
#GL.glEnd()

self.qglColor(block_color)
#GL.glBegin(GL.GL_QUADS)
#фронтальный полигон
GL.glVertex3d(x, y, z)
GL.glVertex3d(x, y+1, z)
GL.glVertex3d(x+1, y+1, z)
GL.glVertex3d(x+1, y, z)
#GL.glEnd()

self.qglColor(block_color)
#GL.glBegin(GL.GL_QUADS)
#верхний полигон
GL.glVertex3d(x, y+1, z)
GL.glVertex3d(x, y+1, 1+z)
GL.glVertex3d(1+x, y+1, 1+z)
GL.glVertex3d(1+x, y+1, z)
#GL.glEnd()

self.qglColor(block_color)
#GL.glBegin(GL.GL_QUADS)
#задний полигон
GL.glVertex3d(x, y, z+1)
GL.glVertex3d(x+1, y, z+1)
GL.glVertex3d(x+1, y+1, z+1)
GL.glVertex3d(x, y+1, z+1)
#GL.glEnd()

self.qglColor(block_color)
#GL.glBegin(GL.GL_QUADS)
#боковой левый полигон
GL.glVertex3d(x, y, z)
GL.glVertex3d(x, y, z+1)
GL.glVertex3d(x, y+1, z+1)
GL.glVertex3d(x, y+1, z)
#GL.glEnd()

self.qglColor(block_color)
#GL.glBegin(GL.GL_QUADS)
#боковой правый полигон
GL.glVertex3d(x+1, y, z)
GL.glVertex3d(x+1, y+1, z)
GL.glVertex3d(x+1, y+1, z+1)
GL.glVertex3d(x+1, y, z+1)
GL.glEnd()

if __name__ == '__main__':

    app = QApplication(sys.argv)

```

```
my_format = QGLFormat.defaultFormat()
my_format.setSampleBuffers(True)
QGLFormat.setDefaultFormat(my_format)
```

```
widget = GLWidget(None)
```

```
widget.resize(640, 480)
```

```
widget.show()
```

```
sys.exit(app.exec_())
```