Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторные работы №3-6 по курсу “Компьютерная Графика”

Студент: А. Т. Бахарев

Преподаватель: Г. С. Филиппов

Группа: М8О-306Б

Дата: 25.12.2018

Оценка:

Подпись:

# Лабораторные работы №3-6

**Тема:** Основы построения фотореалистичных изображений, Ознакомление с технологией OpenGL, Создание шейдерных анимационных эффектов в OpenGL

**Задача:** Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником с использованием OpenGL. Создать шейдерные анимационные эффекты. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

**Вариант:** Прямой эллиптический цилиндр

Исходный код

**main.cpp**  
#include <QtWidgets>

#include <QApplication>

#include <QMainWindow>

#include <MyWidget.hpp>

#include <QSurfaceFormat>

#include <QSlider>

#include <QVBoxLayout>

#include <QHBoxLayout>

#include <QGroupBox>

#include <QTimer>

#include <QLabel>

**template** <**typename** Str, **typename** Parent>

**auto**

make\_label(**const** Str &s, Parent p)

{

**auto** label = **new** QLabel{s, p};

**auto** vbox = **new** QHBoxLayout{};

vbox->addWidget(label);

vbox->addWidget(p);

**auto** gbox = **new** QGroupBox{};

gbox->setLayout(vbox);

**return** gbox;

}

**int** main(**int** argc, **char** \*argv[])

{

QApplication app(argc, argv);

QSurfaceFormat format;

format.setDepthBufferSize(24);

QSurfaceFormat::setDefaultFormat(format);

app.setApplicationName("");

app.setApplicationVersion("");

**auto** widget = **new** MyWidget{};

// slider for setting aprroximation

**auto** approxSlider = **new** QSlider{Qt::Horizontal};

approxSlider->setTickPosition(QSlider::NoTicks);

approxSlider->setValue(4);

approxSlider->setMinimum(4);

approxSlider->setMaximum(256);

approxSlider->setSingleStep(1);

// slider for setting "brightness" of diffuse light

**auto** brightSlider = **new** QSlider{Qt::Horizontal};

brightSlider->setTickPosition(QSlider::NoTicks);

brightSlider->setValue(50);

brightSlider->setMinimum(0);

brightSlider->setMaximum(256);

brightSlider->setSingleStep(1);

// timer for changing ambient part of light

**auto** ambientTimer = **new** QTimer{};

QObject::connect(ambientTimer, &QTimer::timeout, widget, &MyWidget::onAmbientChange);

ambientTimer->start(100);

// slider for setting shiness of material

**auto** shineSlider = **new** QSlider{Qt::Horizontal};

shineSlider->setTickPosition(QSlider::NoTicks);

shineSlider->setValue(50);

shineSlider->setMinimum(0);

shineSlider->setMaximum(100);

shineSlider->setSingleStep(1);

// connecting sliders to widget

QObject::connect(approxSlider, &QSlider::valueChanged, widget, &MyWidget::onApproxChange);

QObject::connect(brightSlider, &QSlider::valueChanged, widget, &MyWidget::onBrightChange);

QObject::connect(shineSlider, &QSlider::valueChanged, widget, &MyWidget::onShineChange);

**auto** qvbox = **new** QVBoxLayout{};

qvbox->addWidget(widget, 9);

// labels for sliders

qvbox->addWidget(make\_label(QObject::tr("Approximation slider"), approxSlider));

qvbox->addWidget(make\_label(QObject::tr("Brightness slider"), brightSlider));

qvbox->addWidget(make\_label(QObject::tr("Shineness slider"), shineSlider));

**auto** groupbox = **new** QGroupBox{QObject::tr("")};

groupbox->setLayout(qvbox);

QMainWindow window{};

window.setCentralWidget(groupbox);

window.resize(QSize(800, 600));

window.show();

**return** app.exec();

}

**MyWidget.hpp**

#ifndef MYWIDGET\_HPP

#define MYWIDGET\_HPP

#include <QOpenGLWidget>

#include <QOpenGLFunctions>

#include <QMatrix4x4>

#include <QQuaternion>

#include <QVector2D>

#include <QBasicTimer>

#include <QOpenGLShaderProgram>

#include <QOpenGLTexture>

#include <QTransform>

#include <utility>

#include <GeometryEngine.hpp>

#include <Light.hpp>

**class** GeometryEngine;

**class** MyWidget : **public** QOpenGLWidget, **protected** QOpenGLFunctions

{

Q\_OBJECT

**public**:

**explicit** MyWidget(QWidget \*parent = **nullptr**);

~MyWidget();

**public slots**:

**void** onApproxChange(**int** value);

**void** onBrightChange(**int** value);

**void** onAmbientChange();

**void** onShineChange(**int** value);

**protected**:

**void** keyPressEvent(QKeyEvent \*event) override;

**void** mousePressEvent(QMouseEvent \*e) override;

**void** mouseReleaseEvent(QMouseEvent \*e) override;

**void** timerEvent(QTimerEvent \*e) override;

**void** initializeGL() override;

**void** resizeGL(**int** width, **int** height) override;

**void** paintGL() override;

**virtual** **void** initShaders();

**private**:

QBasicTimer timer;

QOpenGLShaderProgram program;

GeometryEngine \*geometries;

QOpenGLTexture \*texture;

QMatrix4x4 projection;

QVector2D mousePressPosition;

QVector3D rotationAxis;

qreal angularSpeed;

QQuaternion rotation;

qreal zoom\_scale;

**int** approx\_scale;

Light light;

QVector3D materialSpecularColor;

GLfloat materialShininess;

};

#endif // MYWIDGET\_HPP

**GeometryEngine.hpp**

#ifndef GEOMETRYENGINE\_HPP

#define GEOMETRYENGINE\_HPP

#include <QOpenGLFunctions>

#include <QOpenGLShaderProgram>

#include <QOpenGLBuffer>

#include <vector>

#include <Light.hpp>

**class** GeometryEngine : **protected** QOpenGLFunctions

{

**public**:

GeometryEngine(**int** radius = 1, **int** height = 2, **unsigned** **int** curr\_approx\_points = 4);

**virtual** ~GeometryEngine();

**void** drawGeometry(QOpenGLShaderProgram \*program, **unsigned** **int** approx\_points,

**const** Light &light, **const** QVector3D &materialSpecularColor,

GLfloat materialShininess);

**private**:

**void** initGeometry();

**void** initVertices();

**void** triangulizeBotTop();

**unsigned** **int** triangulize(**unsigned** **int** start\_vertice,

**unsigned** **int** center,

**unsigned** **int** end\_vertice\_ind,

**unsigned** **int** vertex\_ind,

**bool** is\_wall);

**int** radius;

**int** height;

**unsigned** **int** curr\_approx\_points;

**unsigned** **int** indeces\_amount;

QOpenGLBuffer indexBuf;

QOpenGLBuffer arrayBuf;

std::vector<QVector3D> vertices;

std::vector<GLuint> indeces;

};

#endif // GEOMETRYENGINE\_HPP

**Light.hpp**

#ifndef LIGHT\_HPP

#define LIGHT\_HPP

#include <QVector3D>

#include <QOpenGLFunctions>

**struct** Light

{

Light() = **default**;

Light(**const** QVector3D &p, **const** QVector3D &i, GLfloat at, GLfloat am)

: position(p), intensities(i), attenuation(at), ambientCoeff(am){};

~Light() = **default**;

QVector3D position;

QVector3D intensities;

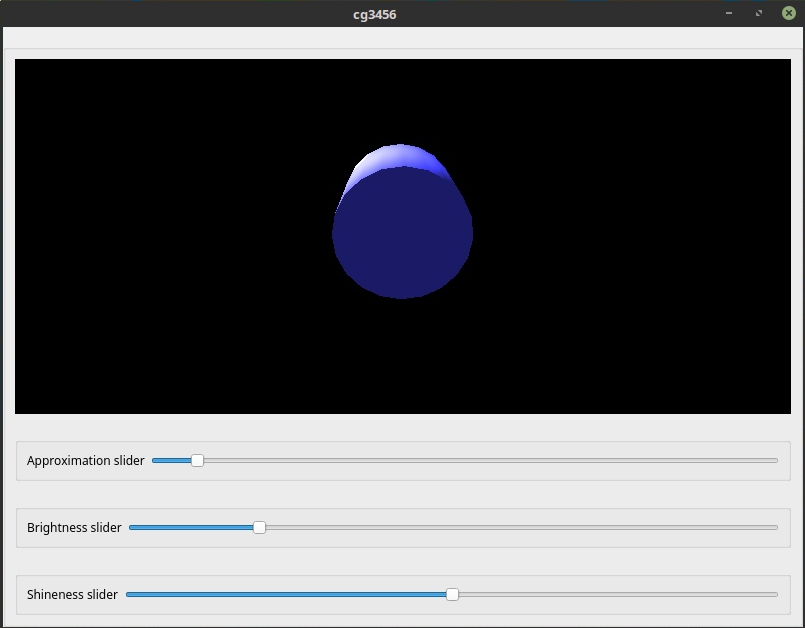
GLfloat attenuation;

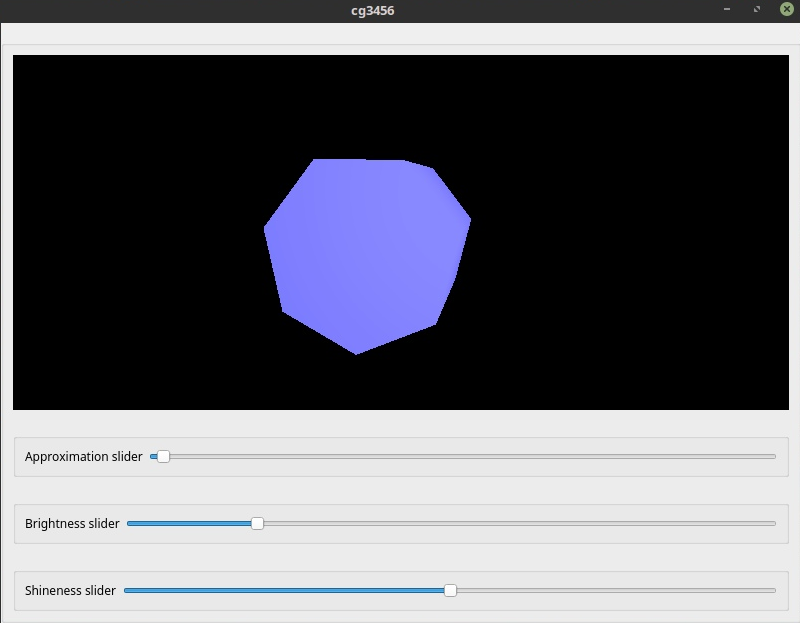
GLfloat ambientCoeff;

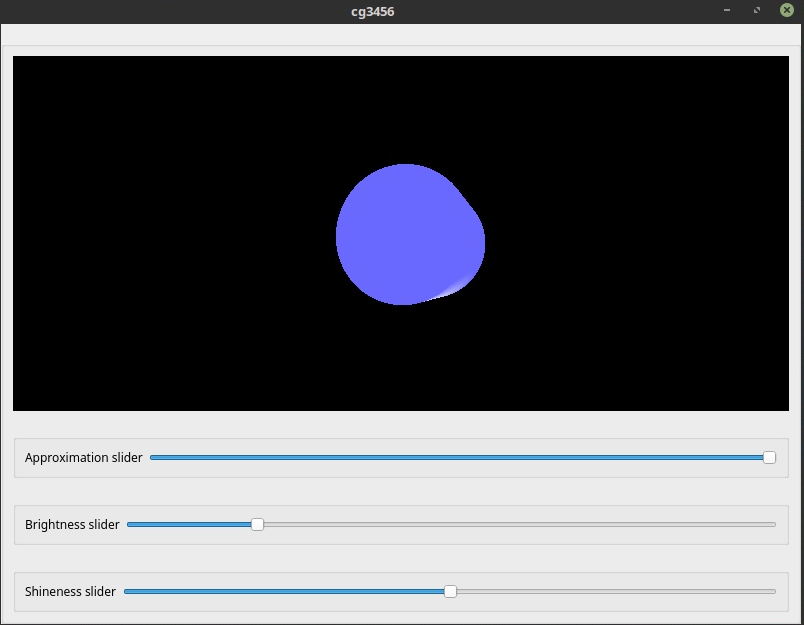
};

#endif // LIGHT\_HPP

# Скриншоты







# Выводы

Выполнив данные лабораторные работы, я познакомился с библиотекой OpenGL, которая позволяет создавать фотореалистичные изображения и трехмерные объекты. Это делается благодаря освещению и его корректному отражению от сторон объекта. В целом, можно сказать, что технология OpenGL незаменима при создании красивых фигур и реалистичных фотоэффектов.