МИНИСТЕРСТВА 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)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №7

Дисциплина: Компьютерная графика

Тема: «Освещение в OpenGL»

Выполнил: ст. группы ПВ-31

Донцов А.А.

Проверил: Осипов О. В.

Белгород, 2019

**Цель работы**

Изучение функций OpenGL для создания простейшего освещения с использованием среды разработки Qt Creator

**Задание**

1. Изучить функции OpenGL для создания освещения.

2. В соответствии с выбранной в предыдущей лабораторной работе предметной областью, добавить к разрабатываемой программе освещение.

**Выполнение работы**

**Main.cpp**

#include <QApplication>

#include "widget.h"

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

{

QGuiApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);

QApplication a(argc, argv);

Widget w;

w.showFullScreen();

w.show();

return a.exec();

}

**Widget.h**

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <QtGui>

#include <QGLWidget>

#include <gl/glu.h>

class Widget : public QGLWidget

{

public:

Widget(QWidget \*parent = 0);

float angle = 0.0;

float lx = 0.0f, lz = 1.0f, ly = 0.0f;

float x = 0.0f, y = 0.0f, z = 0.0f;

void *initializeGL*();

void *resizeGL*(int nWidth, int nHeight);

void *paintGL*();

GLuint textures[2];

void drawRoom();

void drawTree();

void LoadTexture();

QPointF previousPoint;

qreal dw, dh;

qreal step = 0.1f;

void *keyPressEvent*(QKeyEvent \*event);

void *mouseMoveEvent*(QMouseEvent \*event);

void *mousePressEvent*(QMouseEvent \*event);

qreal light\_y;

int type;

protected:

int id\_timer;

void timerEvent(QTimerEvent \*event);

};

#endif // WIDGET\_H

**Widget.cpp**

#include "widget.h"

Widget::Widget(QWidget \*parent)

: QGLWidget(parent)

{

resize(600, 600);

id\_timer=startTimer(25);

*//setMouseTracking(true);*

}

void Widget::*initializeGL*()

{

glClearColor(1, 1, 1, 1); *//* *заполняем* *экран* *белым* *цветом*

LoadTexture();

glEnable(GL\_DEPTH\_TEST); *//* *задаем* *глубину* *проверки* *пикселей*

glShadeModel(GL\_SMOOTH); *//* *убираем* *режим* *сглаживания* *цветов*

glEnable(GL\_CULL\_FACE); *//* *говорим,* *что* *будем* *строить* *только* *внешние* *поверхности*

glPolygonMode(GL\_FRONT\_AND\_BACK,GL\_FILL); *//* *фигуры* *будут* *закрашены* *с* *обеих* *сторон*

glEnable(GL\_LIGHTING);

glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, 1);

glEnable(GL\_NORMALIZE);

}

void Widget::*resizeGL*(int nWidth, int nHeight)

{

glViewport(0, 0, nWidth, nHeight); *//* *установка* *точки* *обзора*

glMatrixMode(GL\_PROJECTION); *//* *установка* *режима* *матрицы*

glLoadIdentity(); *//* *загрузка* *матрицы*

gluPerspective( 90.0f, (GLfloat)nWidth/(GLfloat)nHeight, 0.001f, 1000.0f );

}

void Widget::*paintGL*() *//* *рисование*

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); *//* *очистка* *экрана*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*if* (type % 3 == 0)

{

GLfloat qaAmbientLight[] = {0.0, 0.2, 0.2, 0.2};

GLfloat qaDiffuseLight[] = {1, 0.4, 0.4, 1.0};

GLfloat qaSpecularLight[] = {1, 1, 1, 1};

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, qaAmbientLight);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, qaDiffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, qaSpecularLight);

}

*else* *if* (type % 3 == 1)

{

GLfloat qaAmbientLight[] = {0.2, 0.0, 0.2, 0.2};

GLfloat qaDiffuseLight[] = {0.4, 1, 0.4, 1.0};

GLfloat qaSpecularLight[] = {1, 1, 1, 1};

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, qaAmbientLight);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, qaDiffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, qaSpecularLight);

}

*else*

{

GLfloat qaAmbientLight[] = {0.2, 0.2, 0.0, 0.2};

GLfloat qaDiffuseLight[] = {0.4, 0.4, 1, 1.0};

GLfloat qaSpecularLight[] = {1, 1, 1, 1};

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, qaAmbientLight);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, qaDiffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, qaSpecularLight);

}

*if* (light\_y >= 50)

{

light\_y = 0;

type++;

}

*else* light\_y+= 1;

GLfloat qaLightPosition[] = {0, light\_y, -2.0, 1.0};

glLightfv(GL\_LIGHT0, GL\_POSITION, qaLightPosition);

GLfloat qaAmbientLight1[] = {0.5, 0.5, 0.6, 0.0};

GLfloat qaDiffuseLight1[] = {0.5, 0.5, 0.4, 1.0};

GLfloat qaSpecularLight1[] = {0, 0, 0, 1};

glEnable(GL\_LIGHT1);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, qaAmbientLight1);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, qaDiffuseLight1);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, qaSpecularLight1);

GLfloat qaLightPosition1[] = {x, y, z +1 , 1.0};

glLightfv(GL\_LIGHT1, GL\_POSITION, qaLightPosition1);

GLfloat qaAmbientLight2[] = {0, 0, 0, 0.0};

GLfloat qaDiffuseLight2[] = {0.1, 10, 0.1, 1.0};

GLfloat qaSpecularLight2[] = {5, 0, 0, 1};

glEnable(GL\_LIGHT2);

glLightfv(GL\_LIGHT2, GL\_AMBIENT, qaAmbientLight2);

glLightfv(GL\_LIGHT2, GL\_DIFFUSE, qaDiffuseLight2);

glLightfv(GL\_LIGHT2, GL\_SPECULAR, qaSpecularLight2);

GLfloat qaLightPosition2[] = {0, 0.49, 2.6, 1.0};

GLfloat qaLightDirection[] = {0, -1, 0};

glLightfv(GL\_LIGHT2, GL\_POSITION, qaLightPosition2);

glLightfv(GL\_LIGHT2, GL\_SPOT\_DIRECTION, qaLightDirection);

glLightf(GL\_LIGHT2, GL\_SPOT\_CUTOFF, 60);

GLfloat qaAmbientLight3[] = {0, 0, 0, 0.0};

GLfloat qaDiffuseLight3[] = {0.1, 10, 0.1, 1.0};

GLfloat qaSpecularLight3[] = {5, 0, 0, 1};

glEnable(GL\_LIGHT3);

glLightfv(GL\_LIGHT3, GL\_AMBIENT, qaAmbientLight3);

glLightfv(GL\_LIGHT3, GL\_DIFFUSE, qaDiffuseLight3);

glLightfv(GL\_LIGHT3, GL\_SPECULAR, qaSpecularLight3);

GLfloat qaLightPosition3[] = {0, 0.49, 2.4, 1.0};

GLfloat qaLightDirection1[] = {0, -1, 0};

glLightfv(GL\_LIGHT3, GL\_POSITION, qaLightPosition3);

glLightfv(GL\_LIGHT3, GL\_SPOT\_DIRECTION, qaLightDirection1);

glLightf(GL\_LIGHT3, GL\_SPOT\_CUTOFF, 60);

GLfloat qaAmbientLight4[] = {0, 0, 0, 0.0};

GLfloat qaDiffuseLight4[] = {1, 1, 1, 1.0};

GLfloat qaSpecularLight4[] = {5, 0, 0, 1};

glEnable(GL\_LIGHT4);

glLightfv(GL\_LIGHT4, GL\_AMBIENT, qaAmbientLight4);

glLightfv(GL\_LIGHT4, GL\_DIFFUSE, qaDiffuseLight4);

glLightfv(GL\_LIGHT4, GL\_SPECULAR, qaSpecularLight4);

GLfloat qaLightPosition4[] = {-0.16, 0.4, 2.8, 1.0};

GLfloat qaLightDirection2[] = {0, -1, 0.5};

glLightfv(GL\_LIGHT4, GL\_POSITION, qaLightPosition4);

glLightfv(GL\_LIGHT4, GL\_SPOT\_DIRECTION, qaLightDirection2);

glLightf(GL\_LIGHT4, GL\_SPOT\_CUTOFF, 60);

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

glMatrixMode(GL\_MODELVIEW); *//* *задаем* *модельно-видовую* *матрицу*

glLoadIdentity(); *//* *загрузка* *единичную* *матрицу*

*//glRotatef(dw,* *0,* *1,* *0);*

*/\*float* *ambient[4]* *=* *{1,* *1,* *1,* *1};*

*glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT,* *ambient);\*/*

*if* (x >= 1)

x = 0.8;

*else* *if* (x <= -1)

x = -0.8;

*if* (z >= 2)

z = 1.8;

*else* *if* (z <= -1)

z = -0.8;

gluLookAt(x, y, z,

x+lx, y+ly, z+lz,

0.0f, 1.0f, 0.0f );

drawRoom();

drawTree();

glDisable(GL\_LIGHT0);

glDisable(GL\_LIGHT1);

glDisable(GL\_LIGHT2);

glDisable(GL\_LIGHT3);

}

void Widget::drawRoom()

{

*//glEnable(GL\_TEXTURE\_2D);*

glTranslatef(0.0f,0.0f,-1.0f);

glColor3f(0, 0, 0);

*//* *Плато*

glBegin(GL\_QUADS);

glVertex3f(-10, -0.51, -10);

glVertex3f(-10, -0.51, 10);

glVertex3f(10, -0.51, 10);

glVertex3f(10, -0.51, -10);

glEnd();

glColor3f(0, 0, 1); *//* *задаем* *цвет*

*//* *Стены* *без* *окна*

*//glBindTexture(GL\_TEXTURE\_2D,* *textures[0]);*

glBegin(GL\_QUADS);

glTexCoord2d(0, 0); glVertex3f(-1, 0.5, 0.0);

glTexCoord2d(0, 1); glVertex3f(-1, 0.5, 3);

glTexCoord2d(1, 1); glVertex3f(-1, -0.5, 3);

glTexCoord2d(1, 0); glVertex3f(-1, -0.5, 0.0);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(1, 0.5, 3);

glVertex3f(1, 0.5, 0);

glVertex3f(1, -0.5, 0);

glVertex3f(1, -0.5, 3);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(1, 0.5, 3);

glVertex3f(1, -0.5, 3);

glVertex3f(-1, -0.5, 3);

glVertex3f(-1, 0.5, 3);

glEnd();

*//* *Стена* *с* *окном*

glBegin(GL\_QUADS);

glTexCoord2d(1, 1); glVertex3f(1, 0.5, 0);

glTexCoord2d(0, 1); glVertex3f(0.5, 0.3, 0);

glTexCoord2d(0, 0); glVertex3f(0.5, -0.3, 0);

glTexCoord2d(1, 0); glVertex3f(1, -0.5, 0);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(-1, 0.5, 0);

glVertex3f(-1, -0.5, 0);

glVertex3f(-0.5, -0.3, 0);

glVertex3f(-0.5, 0.3, 0);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(-1, 0.5, 0);

glVertex3f(-0.5, 0.3, 0);

glVertex3f(0.5, 0.3, 0);

glVertex3f(1, 0.5, 0);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(-1, -0.5, 0);

glVertex3f(1, -0.5, 0);

glVertex3f(0.5, -0.3, 0);

glVertex3f(-0.5, -0.3, 0);

glEnd();

*//* *Пол*

glColor3f(1, 0, 0); *//* *задаем* *цвет*

glBegin(GL\_QUADS);

glVertex3f(-1, -0.5, 0);

glVertex3f(-1, -0.5, 3);

glVertex3f(1, -0.5, 3);

glVertex3f(1, -0.5, 0);

glEnd();

*//* *Потолок*

glColor3f(1, 0, 1); *//* *задаем* *цвет*

glBegin(GL\_QUADS);

glVertex3f(-1, 0.5, 0);

glVertex3f(1, 0.5, 0);

glVertex3f(1, 0.5, 3);

glVertex3f(-1, 0.5, 3);

glEnd();

*//* *Картина*

glColor3f(1, 1, 1); *//* *задаем* *цвет*

glBegin(GL\_QUADS);

glVertex3f(0, 0.4, 2.99);

glVertex3f(0, 0, 2.99);

glVertex3f(-0.32, 0, 2.99);

glVertex3f(-0.32, 0.4, 2.99);

glEnd();

}

void Widget::drawTree()

{

*//* *Верхий* *уровень*

glColor3f(0.44, 0.66, 0.5);

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*0.5, -0.5, 2.5 + 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*0.5, -0.5, 2.5 - 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*0.5, -0.5, 2.5 + 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45, -0.5, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*0.5, -0.5, 2.5 - 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

*//* *Средний* *уровень*

glColor3f(0.32, 0.54, 0.39);

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*sqrt(3)\*0.5, -0.5, 2.5 + 0.45\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*sqrt(3)\*0.5, -0.5, 2.5 + 0.45\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5, -0.5, 2.5 - 0.45);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45\*sqrt(3)\*0.5, -0.5, 2.5 + 0.45\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*sqrt(3)\*0.5, -0.5, 2.5 + 0.45\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5, -0.5, 2.5 - 0.45);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

*//* *Нижний* *уровень*

glColor3f(0.24, 0.4, 0.29);

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45, -0.5, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*0.5, -0.5, 2.5 + 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*0.5, -0.5, 2.5 - 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5+0.45, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*0.5, -0.5, 2.5 + 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex3f(0.5-0.45\*0.5, -0.5, 2.5 - 0.45\*sqrt(3)\*0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, 0.45, 2.5);

glEnd();

}

void Widget::LoadTexture()

{

glGenTextures(5, textures);

QImage walls;

walls.load(QDir::currentPath() + "\\walls.jpg");

walls = QGLWidget::convertToGLFormat(walls);

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, (GLsizei)walls.width(), (GLsizei)walls.height(), 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, walls.bits());

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

QImage el;

el.load(QDir::currentPath() + "\\el.jpg");

el = QGLWidget::convertToGLFormat(el);

glBindTexture(GL\_TEXTURE\_2D, textures[1]);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, (GLsizei)el.width(), (GLsizei)el.height(), 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, el.bits());

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

QImage pol;

pol.load(QDir::currentPath() + "\\pol.jpg");

pol = QGLWidget::convertToGLFormat(pol);

glBindTexture(GL\_TEXTURE\_2D, textures[2]);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, (GLsizei)pol.width(), (GLsizei)pol.height(), 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, pol.bits());

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

QImage pot;

pot.load(QDir::currentPath() + "\\potolok.jpg");

pot = QGLWidget::convertToGLFormat(pot);

glBindTexture(GL\_TEXTURE\_2D, textures[3]);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, (GLsizei)pot.width(), (GLsizei)pot.height(), 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, pot.bits());

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

QImage pic;

pic.load(QDir::currentPath() + "\\new\_year.jpg");

pic = QGLWidget::convertToGLFormat(pic);

glBindTexture(GL\_TEXTURE\_2D, textures[4]);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, (GLsizei)pic.width(), (GLsizei)pic.height(), 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, pic.bits());

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

}

void Widget::*keyPressEvent*(QKeyEvent \*event) {

*switch* (event->key()) {

*case* Qt::*Key\_W*:

x += lx \* step;

z += lz \* step;

*break*;

*case* Qt::*Key\_S*:

x -= lx \* step;

z -= lz \* step;

*break*;

}

//*updateGL*(); *//* *обновление* *изображения*

}

void Widget::*mousePressEvent*(QMouseEvent \*event)

{

previousPoint = event->windowPos();

}

void Widget::*mouseMoveEvent*(QMouseEvent \*event)

{

dh += (event->y() - previousPoint.y()) / 1000;

dw += (event->x() - previousPoint.x()) / 1000;

lx = -sin(dw);

ly = -tan(dh);

lz = +cos(dw);

*//Обновление* *данных* *о* *предыдущей* *координате*

previousPoint = event->windowPos();

//*updateGL*(); *//* *обновление* *изображения*

}

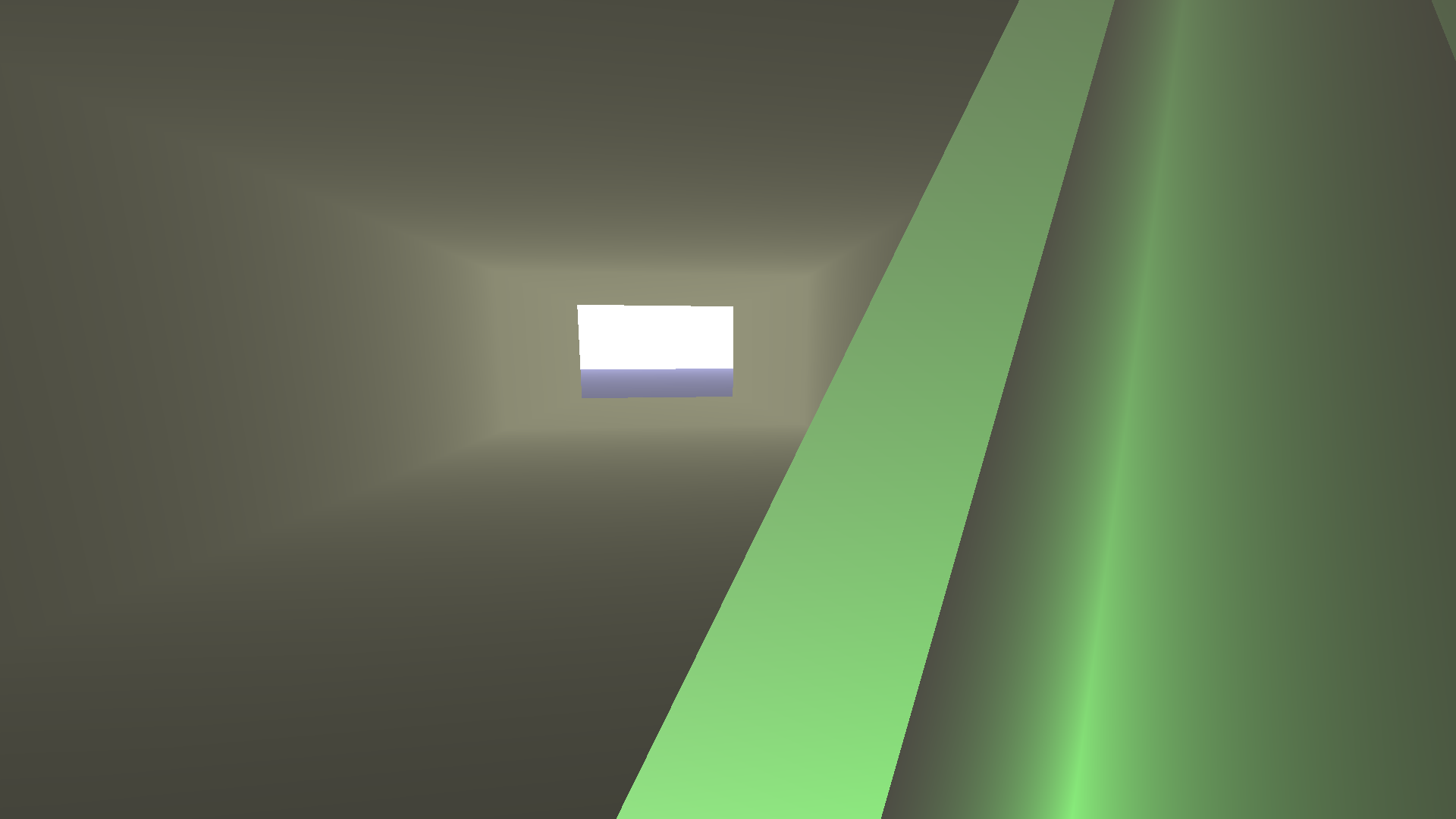
void Widget::timerEvent(QTimerEvent \*event)

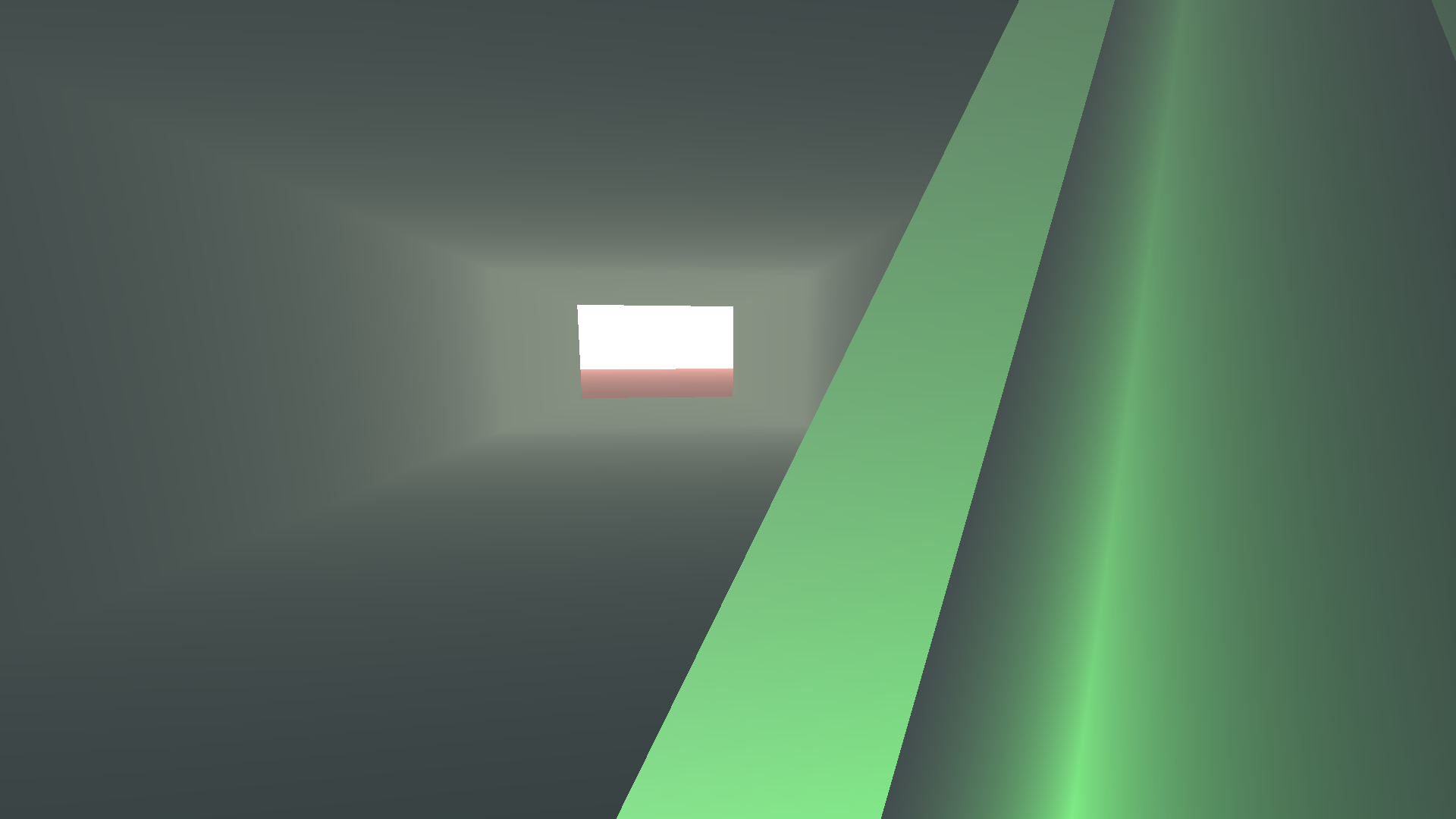
{

updateGL();

}

**Скриншот результата работы программы**:





**Вывод**: выполнение данной лабораторной работы помогло изучить функции OpenGL для создания простейшего освещения с использованием среды разработки Qt Creator