МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

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

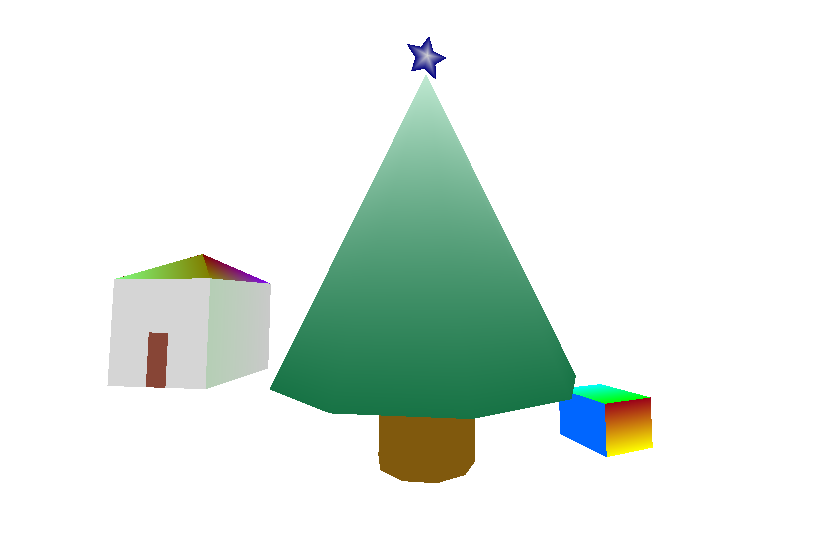
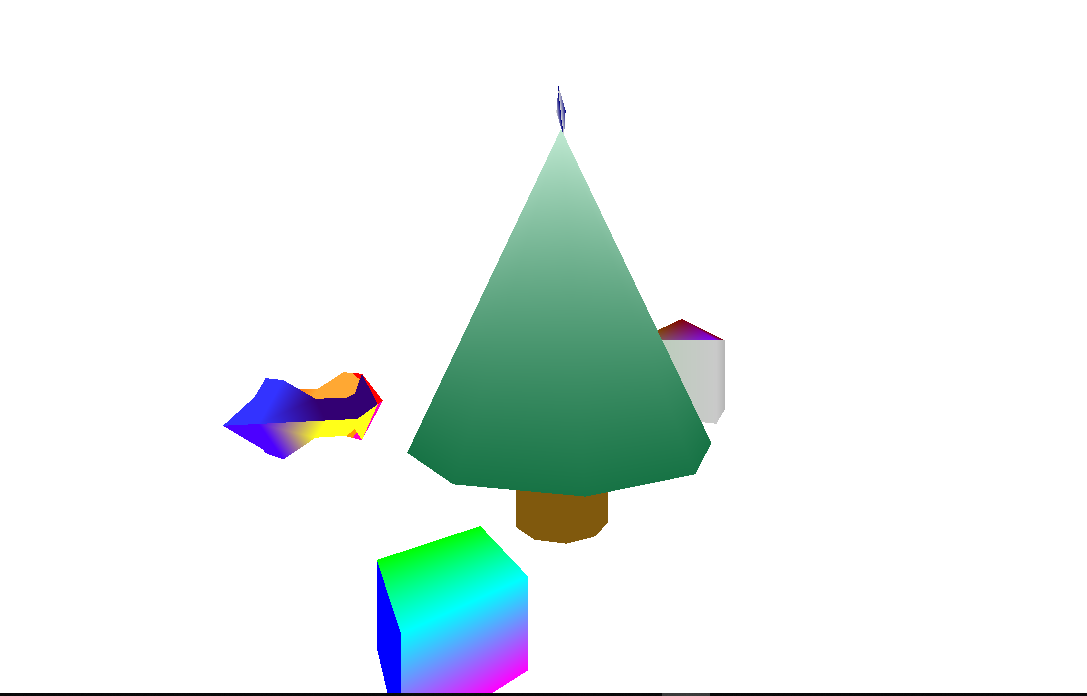
Дисциплина: Компьютерная графика

Лабораторная работа № 6

|  |  |
| --- | --- |
|  | Выполнила: ст. группы ПВ-31  Зановская А.И.  Проверил: Осипов О.В. |

Белгород

2018



Код программы

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

MainWindow::**MainWindow**(QWidget \*parent) :

QGLWidget(parent),

ui(new Ui::MainWindow)

{

timer = new QTimer();

connect(timer,SIGNAL (timeout()), this, SLOT(timerUpd()));

timer->start(10);

angle\_x=angle\_y=0;

nSca=ScaleHouse=1;

checkHouse=checkMirror=true;

xTra = yTra = zTra = 0.0;

angleTable=angleSpiderWeb=0;

angleVase=0;

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**timerUpd**()

{

angleTable++;

angleSpiderWeb+=4;

angleVase+=3;

if(checkHouse)

{

ScaleHouse+=0.006;

}

else

{

ScaleHouse-=0.006;

}

if(ScaleHouse>1.5) checkHouse=false;

else if (ScaleHouse<1.0) checkHouse=true;

repaint();

}

void MainWindow::***initializeGL***()

{

qglClearColor(Qt::white); //цвет фона

glEnable(GL\_DEPTH\_TEST);

}

void MainWindow::***resizeGL***(int width, int height)

{

glViewport(0,0,width,height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(90.0, (GLdouble)width/(GLdouble)height, 0.001, 1000.0);

}

void MainWindow::***paintGL***()

{

//Очистка буфера глубины и буфера цвета

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

//Переключение на матрицу вида

glMatrixMode(GL\_MODELVIEW);

//Загружаем единичную матрицу вида

glLoadIdentity();

glTranslatef(0,0,-3.0);

glRotatef(angle\_y, 1.0, 0.0,0.0);

glRotatef(angle\_x, 0.0, 1.0, 0.0);

glTranslatef(xTra, yTra, zTra);

glScalef(nSca, nSca, nSca);

drawBox();

drawRing();

drawElka();

drawHouse();

drawStar();

}

void MainWindow::***mousePressEvent***(QMouseEvent\* event) {

m\_mousePosition.setX(event->x());

m\_mousePosition.setY(event->y());

}

void MainWindow::***mouseMoveEvent***(QMouseEvent\* m\_event)

{

double dx = m\_event->x() - m\_mousePosition.x();

double dy = m\_event->y() - m\_mousePosition.y();

m\_mousePosition=m\_event->pos();

angle\_x += dx ;

angle\_y += dy ;

repaint();

}

void MainWindow::**scale\_plus**() // приблизить сцену

{

nSca = nSca\*1.1;

}

void MainWindow::**scale\_minus**() // удалиться от сцены

{

nSca = nSca/1.1;

}

void MainWindow::**translate\_down**() // транслировать сцену вниз

{

zTra -= 0.05;

}

void MainWindow::**translate\_up**() // транслировать сцену вверх

{

zTra += 0.05;

}

void MainWindow::**translate\_Vver**() // транслировать сцену вниз

{

yTra -= 0.05;

}

void MainWindow::**translate\_Vniz**() // транслировать сцену вверх

{

yTra += 0.05;

}

void MainWindow::***keyPressEvent***(QKeyEvent \*event)

{

switch (event->key())

{

case Qt::Key\_Plus:

scale\_plus(); // приблизить сцену

break;

case Qt::Key\_Equal:

scale\_plus(); // приблизить сцену

break;

case Qt::Key\_Minus:

scale\_minus(); // удалиться от сцены

break;

case Qt::Key\_D:

translate\_down(); // транслировать сцену вниз

break;

case Qt::Key\_A:

translate\_up(); // транслировать сцену вверх

break;

case Qt::Key\_W:

translate\_Vver(); // транслировать сцену вниз

break;

case Qt::Key\_S:

translate\_Vniz(); // транслировать сцену вверх

break;

}

*updateGL*(); // обновление изображения

}

void MainWindow::**drawHouse**(){

glPushMatrix();

glRotatef(90, -1.0, 0.0, 0.0);

glRotatef(0, -1.0, 0.0, 0.0);

glTranslatef(2.0,-1.0,-1.0);

//стены

glBegin(GL\_QUAD\_STRIP);

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(1, 1, 0);

glVertex3f(1, 1, 1);

glColor3f(0.71f, 0.71f, 0.71f);

glVertex3f(1, 0, 0);

glVertex3f(1, 0, 1);

glColor3f(0.79f, 0.79f, 0.79f);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1);

glColor3f(0.71f, 0.81f, 0.71f);

glVertex3f(0, 1, 0);

glVertex3f(0, 1, 1);

glEnd();

//стена с дверью

glBegin(GL\_QUADS);

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(0, 1, 0);

glVertex3f(0, 1, 1);

glVertex3f(0.4, 1, 1);

glVertex3f(0.4, 1, 0);

glVertex3f(0.4, 1, 1);

glVertex3f(0.4, 1, 0.5);

glVertex3f(0.6, 1, 0.5);

glVertex3f(0.6, 1, 1);

glVertex3f(1, 1, 0);

glVertex3f(1, 1, 1);

glVertex3f(0.6, 1, 1);

glVertex3f(0.6, 1, 0);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.25f, 0.0f, 0.5f);

glVertex3f(0, 1, 0);

glVertex3f(1, 1, 0);

glVertex3f(1, 0, 0);

glVertex3f(0, 0, 0);

glColor3f(0.53f, 0.27f, 0.21f);

glVertex3f(0.4, 1, 0);

glVertex3f(0.4, 1, 0.5);

glVertex3f(0.6, 1, 0.5);

glVertex3f(0.6, 1, 0);

glEnd();

glBegin(GL\_TRIANGLE\_STRIP);

glColor3f(0.5f, 0.0f, 0.0f);

glVertex3f(1, 1, 1);

glVertex3f(1, 0, 1);

glVertex3f(0.5, 0.5, 1.3);

glColor3f(0.5f, 0.0f, 5.0f);

glVertex3f(0, 0, 1);

glColor3f(0.5f, 0.5f, 0.0f);

glVertex3f(0, 1, 1);

glVertex3f(0.5, 0.5, 1.3);

glColor3f(0.5f, 1.0f, 0.5f);

glVertex3f(1, 1, 1);

glEnd();

glPopMatrix();

}

void MainWindow::**drawStar**(){

glPushMatrix();

glTranslatef(0.0,1.95,0.0);

glScalef(0.2, 0.2, 0.2);

glRotatef(36, 0.0, 1.0, 1.0);

int a = 36;

glBegin(GL\_TRIANGLES);

glColor3f(0.835f, 0.835f, 0.835f);

for (int i = 0; i < 10; i+=2)

{

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(cos(i\*a\*M\_PI/180), sin(i\*a\*M\_PI/180), 0); //выступ

glVertex3f(0.5\*cos((i+1)\*a\*M\_PI/180), 0.5\*sin((i+1)\*a\*M\_PI/180), 0);

glColor3f(0.0f, 0.0f, 0.5f);

glVertex3f(0, 0, 0.2);

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(0.5\*cos((i+1)\*a\*M\_PI/180), 0.5\*sin((i+1)\*a\*M\_PI/180), 0); glVertex3f(cos((i+2)\*a\*M\_PI/180), sin((i+2)\*a\*M\_PI/180), 0);

glColor3f(0.0f, 0.0f, 0.5f);

glVertex3f(0, 0, 0.2);

glColor3f(0.0f, 0.0f, 0.5f);

glVertex3f(cos(i\*a\*M\_PI/180), sin(i\*a\*M\_PI/180), 0); //выступ

glVertex3f(0.5\*cos((i+1)\*a\*M\_PI/180), 0.5\*sin((i+1)\*a\*M\_PI/180), 0);

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(0, 0, -0.2);

glColor3f(0.0f, 0.0f, 0.5f);

glVertex3f(0.5\*cos((i+1)\*a\*M\_PI/180), 0.5\*sin((i+1)\*a\*M\_PI/180), 0);

glVertex3f(cos((i+2)\*a\*M\_PI/180), sin((i+2)\*a\*M\_PI/180), 0);

glColor3f(0.835f, 0.835f, 0.835f);

glVertex3f(0, 0, -0.2);

}

glEnd();

glPopMatrix();

}

void MainWindow::**drawRing**()

{

glPushMatrix();

glRotatef(90, -1.0, 0.0, 0.0);

glTranslatef(0,0.0,-1.0);

glBegin(GL\_TRIANGLE\_STRIP);

glColor3f(1.0f, 0.0f, 0.0f);

glVertex3f(0.5, 4, 0.5);

glVertex3f(0.5, 3.75, 0.5);

glVertex3f(1, 4, 0);

glVertex3f(0.75, 3.75, 0);

glEnd();

glBegin(GL\_QUAD\_STRIP);

glColor3f(0.2f, 0.0f, 0.45f);

glVertex3f(0.5, 3.75, 0.5);

glVertex3f(0.75, 3.75, 0);

glVertex3f(0.25, 3.5, 0.25);

glVertex3f(0.25, 3.25, 0);

glVertex3f(0, 3.25, 0.25);

glVertex3f(0, 3, 0);//7

glVertex3f(-0.25, 3.5, 0.25);

glVertex3f(-0.25, 3.25, 0);//8

glColor3f(0.2f, 0.2f, 1.0f);

glVertex3f(-0.5, 3.75, 0.5);

glVertex3f(-0.75, 3.75, 0);

glVertex3f(-0.5, 4, 0.5);

glVertex3f(-1, 4, 0);

glVertex3f(-0.5, 4.25, 0.5);

glVertex3f(-0.75, 4.25, 0);

glColor3f(1.0f, 0.4f, 0.0f);

glVertex3f(-0.25, 4.5, 0.25);

glVertex3f(-0.25, 4.75, 0);//16

glVertex3f(0, 4.75, 0.25);

glVertex3f(0, 5, 0);

glColor3f(1.0f, 0.66f, 0.2f);

glVertex3f(0.25, 4.5, 0.25);

glVertex3f(0.25, 4.75, 0); //20

glVertex3f(0.5, 4.25, 0.5);

glVertex3f(0.75, 4.25, 0);

glVertex3f(0.5, 4, 0.5);

glVertex3f(1, 4, 0);

glEnd();

glBegin(GL\_TRIANGLE\_STRIP);

glColor3f(1.0f, 0.0f, 0.77f);

glVertex3f(0.5, 4, -0.5);

glVertex3f(0.5, 3.75, -0.5);

glVertex3f(1, 4, 0);

glVertex3f(0.75, 3.75, 0);

glEnd();

glBegin(GL\_QUAD\_STRIP);

glColor3f(1.0f, 1.0f, 0.1f);

glVertex3f(0.5, 3.75, -0.5);

glVertex3f(0.75, 3.75, 0);

glVertex3f(0.25, 3.5, -0.25);

glVertex3f(0.25, 3.25, 0);

glVertex3f(0, 3.25, -0.25);

glVertex3f(0, 3, 0);//7

glVertex3f(-0.25, 3.5, -0.25);

glVertex3f(-0.25, 3.25, 0);//8

glColor3f(0.3f, 0.0f, 1.0f);

glVertex3f(-0.5, 3.75, -0.5);

glVertex3f(-0.75, 3.75, 0);

glVertex3f(-0.5, 4, -0.5);

glVertex3f(-1, 4, 0);

glVertex3f(-0.5, 4.25, -0.5);

glVertex3f(-0.75, 4.25, 0);

glVertex3f(-0.25, 4.5, -0.25);

glVertex3f(-0.25, 4.75, 0);//16

glVertex3f(0, 4.75, -0.25);

glVertex3f(0, 5, 0);

glColor3f(1.0f, 0.6f, 0.0f);

glVertex3f(0.25, 4.5, -0.25);

glVertex3f(0.25, 4.75, 0); //20

glVertex3f(0.5, 4.25, -0.5);

glVertex3f(0.75, 4.25, 0);

glVertex3f(0.5, 4, -0.5);

glVertex3f(1, 4, 0);

glEnd();

glPopMatrix();

}

void MainWindow::**drawBox**(){

glPushMatrix();

glTranslatef(-1.5,-1,0.3);

glRotatef(90, 1.0, 0.0,0.0);

glRotatef(angleTable,0.0, 0.0,1.0);

glBegin(GL\_QUAD\_STRIP);

glColor3f(0.0f, 1.0f, 0.0f);

glVertex3f(0.25, 0.25, 0);

glVertex3f(-0.25, 0.25, 0);

glColor3f(0.0f, 1.0f, 1.0f);

glVertex3f(0.25, -0.25, 0);

glVertex3f(-0.25, -0.25, 0);

glColor3f(1.0f, 0.0f, 1.0f);

glVertex3f(0.25, -0.25, 0.4);

glVertex3f(-0.25, -0.25, 0.4);

glColor3f(1.0f, 1.0f, 0.0f);

glVertex3f(0.25, 0.25, 0.4);

glVertex3f(-0.25, 0.25, 0.4);

glColor3f(0.6f, 0.0f, 0.1f);

glVertex3f(0.25, 0.25, 0);

glVertex3f(-0.25, 0.25, 0);

glEnd();

glColor3f(0.0f, 0.0f, 1.0f);

glBegin(GL\_QUADS);

glVertex3f(0.25, 0.25, 0);

glVertex3f(0.25, -0.25, 0);

glVertex3f(0.25, -0.25, 0.4);

glVertex3f(0.25, 0.25, 0.4);

glColor3f(0.0f, 0.4f, 1.0f);

glVertex3f(-0.25, 0.25, 0);

glVertex3f(-0.25, -0.25, 0);

glVertex3f(-0.25, -0.25, 0.4);

glVertex3f(-0.25, 0.25, 0.4);

glEnd();

glPopMatrix();

}

// Рисует елочку

void MainWindow::**drawElka**()

{

glPushMatrix();

glRotatef(90, -1.0, 0.0, 0.0);

glRotatef(0, -1.0, 0.0, 0.0);

glTranslatef(0, 0, -1.0);

GLint n=10;

GLfloat angle;

GLfloat a = M\_PI / 180.0;

GLfloat delta = 360.0 / (GLfloat)n;

glColor3f(0.5f, 0.35f, 0.05f);

// рисуем ствол

glBegin(GL\_QUAD\_STRIP);

angle = 0.0;

for (int i = 0; i <= n; i++)

{

glVertex3f(cos(a \* angle)/2.5,sin(a \* angle)/2.5, 0.2);

glVertex3f(cos(a \* angle)/2.5,sin(a \* angle)/2.5, -0.5);

angle += delta;

}

glEnd();

glColor3f(0.09f, 0.45f, 0.27f);

glBegin(GL\_QUAD\_STRIP);

angle = 0.0;

for (int i = 0; i <= n; i++)

{

glColor3f(0.76f, 0.92f, 0.83f);

glVertex3f(0,0,2.8);

glColor3f(0.09f, 0.45f, 0.27f);

glVertex3f(1.2 \* cos(a \* angle), 1.2 \* sin(a \* angle), 0.2);

angle += delta;

}

glEnd();

glPopMatrix();

}