*Горшков Алексей Олегович*

*Группа 4212*

**Отчет о выполнении лабораторной работы №4 «Трехмерное моделирование полигональными сетками»**

1. **Задание:**

I*.*

1. *Создать приложение на основе библиотеки (GLUT) OpenGL Utility Toolkit, которое открывает окно для рисования геометрического объекта. При этом режим дисплея использует двойную буферизацию, режим RGBA и буфер глубины.*
2. *Установить перспективную проекцию с углом обзора 60о.*
3. *Расположить наблюдателя в позицию (0, 3, 3), направленного в точку (0, 0, 0) и вектором направления вверх (0, 1, 0).*
4. *Выполнить рисование тетраэдра, используя пример из листинга 1.*
5. *Создать локальный источник света, используя пример из листинга 2.*
6. *Осуществить вращение геометрического объекта (источника света) вокруг оси Y, при помощи обработчика отсутствия событий. При нажатии на клавишу ‘o’(‘l’) – вращается объект (источник света).*
7. *Осуществить визуализацию лицевых (нелицевых) граней. При нажатии на клавишу ‘f’(‘b’) – визуализируются лицевые (нелицевые) грани.*

II.

1. *Построить полигональную сетку геометрического объекта, который указан в варианте задания. Для этого необходимо заполнить таблицы списков: вершин, нормалей и граней.*
2. *Вместо рисования тетраэдра осуществить рисование геометрического объекта по заполненным таблицам.*
3. *Установить для лицевых граней объекта свойства материала, коэффициенты которого приведены в соответствующем варианте задания.*

(Вариант 6):

(материал - медь)

ambient = (0.19125, 0.0735, 0.0225, 1)

diffuse = (0.7038, 0.27048, 0.0828, 1)

specular = (0.256777, 0.137622, 0.086014, 1)

shininess = (12.8)

1. **Листинг программы:**

I*.*

#include <gl\glut.h>

#include <iostream>

//Переменные для вращения объекта/камеры на определенное число градусов.

GLfloat angle\_object{}, angle\_light{};

//Переменные для начала вращения объекта/камеры.

GLboolean object\_rotate{ false }, light\_rotate{ false };

void init(void)

{

glClearColor(0.5, 0.5, 0.71, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

//Функция для "отлова" событий клавиатуры (клавиатурных событий).

void KeyboardEvents(unsigned char key, int x, int y)

{

if (key == static\_cast<unsigned char>('o'))

{

object\_rotate = true;

light\_rotate = false;

}

else if (key == static\_cast<unsigned char>('l'))

{

light\_rotate = true;

object\_rotate = false;

}

else if (key == static\_cast<unsigned char>('f'))

{

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == static\_cast<unsigned char>('b'))

{

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

void myDisplay()

{

glPushMatrix();

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

glEnable(GL\_DEPTH\_TEST);//Включаем тестовую глубину для реализации изображения.

//Задание положения камеры.

gluLookAt(0, 3, 3, 0, 0, 0, 0, 1, 0);

glPushMatrix();

//Задание освещения.

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

GLfloat myLightPosition[] = { 1.0, 2.0, 2.0, 1.0 };

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

//Включаем источник света.

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPopMatrix();

glPushMatrix();

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

//Рисование тетраэдра.

//грань f0

glBegin(GL\_POLYGON);

glNormal3f(0.577, 0.577, 0.577);

glVertex3f(1, 0, 0);

glVertex3f(0, 1, 0);

glVertex3f(0, 0, 1);

glEnd();

//грань f1

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(0, 0, 0);

glVertex3f(0, 1, 0);

glVertex3f(1, 0, 0);

glEnd();

//грань f2

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1);

glVertex3f(0, 1, 0);

glEnd();

//грань f3

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(1, 0, 0);

glVertex3f(0, 0, 1);

glVertex3f(0, 0, 0);

glEnd();

glPopMatrix();

glDisable(GL\_LIGHTING);//Выключаем освещение для рисования линий и прочего в цвете.

//Рисование координатных осей.

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(3, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 3, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 3);

glEnd();

//Рисование источника освещения

//Расположение источника света для наглядности.

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

glPointSize(20.0);

glBegin(GL\_POINTS);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(1.0, 2.0, 2.0);

glEnd();

//Рисование направление к источнику света.

glBegin(GL\_LINES);

glColor3f(1, 1, 1); glVertex3f(0.0, 0.0, 0.0); glVertex3f(1.0, 2.0, 2.0);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void myReshape(int width, int height)

{

if (width / height < 1) glViewport(0, 0, width, width);

else glViewport(0, 0, height, height);

}

void myIdle()

{

if (light\_rotate)

{

angle\_light += 0.005;

if (angle\_light >= 360) angle\_light = 0;

}

else if (object\_rotate)

{

angle\_object += 0.01;

if (angle\_object >= 360) angle\_object = 0;

}

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("OpenGl\_Lab4\_Prog1");

glutDisplayFunc(myDisplay);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

glutKeyboardFunc(KeyboardEvents);

init();

glutMainLoop();

}

II*.*

#include <gl\glut.h>

#include <iostream>

//Переменные для вращения объекта/камеры на определенное число градусов.

GLfloat angle\_object{}, angle\_light{};

//Переменные для начала вращения объекта/камеры.

GLboolean object\_rotate{ false }, light\_rotate{ false };

void init(void)

{

glClearColor(0.04, 0.215, 0.11, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 50);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

//Функция для "отлова" событий клавиатуры (клавиатурных событий).

void KeyboardEvents(unsigned char key, int x, int y)

{

if (key == static\_cast<unsigned char>('o'))

{

object\_rotate = true;

light\_rotate = false;

}

else if (key == static\_cast<unsigned char>('l'))

{

light\_rotate = true;

object\_rotate = false;

}

else if (key == static\_cast<unsigned char>('f'))

{

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == static\_cast<unsigned char>('b'))

{

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

void myDisplay()

{

glPushMatrix();

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

glEnable(GL\_DEPTH\_TEST);//Включаем тестовую глубину для реализации изображения.

//Задаем положение камеры.

gluLookAt(10, 12, 20, 0, 0, 0, 0, 1, 0);

glPushMatrix();

//Задание освещения.

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

GLfloat myLightPosition[] = { 10.0, 2.0, 5.0, 1.0 };

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

//Включаем источник света.

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPopMatrix();

glPushMatrix();

//Задание материала объекта.

GLfloat myAmbient[] = { 0.19125, 0.0735, 0.0225, 1 };

GLfloat myDiffuse[] = { 0.7038, 0.27048, 0.0828, 1 };

GLfloat mySpecular[] = { 0.256777, 0.137622, 0.086014, 1 };

GLfloat myShininess[] = { 12.8 };

glMaterialfv(GL\_FRONT, GL\_AMBIENT, myAmbient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mySpecular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, myShininess);

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

//Рисование усеченной пирамиды.

//грань f0

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(0, 0, 0);

glVertex3f(8, 0, -4);

glVertex3f(6, 0, 2);

glEnd();

//грань f1

glBegin(GL\_POLYGON);

glNormal3f(-12, 9, 8);

glVertex3f(0, 0, 0);

glVertex3f(3, 4, 0);

glVertex3f(5, 4, -3);

glVertex3f(8, 0, -4);

glEnd();

//грань f2

glBegin(GL\_POLYGON);

glNormal3f(4, 3, -8);

glVertex3f(8, 0, -4);

glVertex3f(5, 4, -3);

glVertex3f(5, 4, 1);

glVertex3f(6, 0, 2);

glEnd();

//грань f3

glBegin(GL\_POLYGON);

glNormal3f(3, 1, 1);

glVertex3f(6, 0, 2);

glVertex3f(5, 4, 1);

glVertex3f(3, 4, 0);

glVertex3f(0, 0, 0);

glEnd();

//грань f4

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 4);

glVertex3f(3, 4, 0);

glVertex3f(5, 4, 1);

glVertex3f(5, 4, -3);

glEnd();

glEnable(GL\_NORMALIZE);

glPopMatrix();

glDisable(GL\_LIGHTING);//Выключаем освещение для рисования линий и прочего в цвете.

//Рисование координатных осей.

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(10, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 10, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 10);

glEnd();

//Рисование источника освещения

//Расположение источника света для наглядности.

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

glPointSize(20.0);

glBegin(GL\_POINTS);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(10.0, 2.0, 5.0);

glEnd();

//Рисование направление к источнику света.

glBegin(GL\_LINES);

glColor3f(1, 1, 1); glVertex3f(0.0, 0.0, 0.0); glVertex3f(10.0, 2.0, 5.0);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void myReshape(int width, int height)

{

if (width / height < 1) glViewport(0, 0, width, width);

else glViewport(0, 0, height, height);

}

void myIdle()

{

if (light\_rotate)

{

angle\_light += 0.05;

if (angle\_light >= 360) angle\_light = 0;

}

else if (object\_rotate)

{

angle\_object += 0.03;

if (angle\_object >= 360) angle\_object = 0;

}

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(0, 0);

glutCreateWindow("OpenGl\_Lab4\_Prog1");

glutDisplayFunc(myDisplay);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

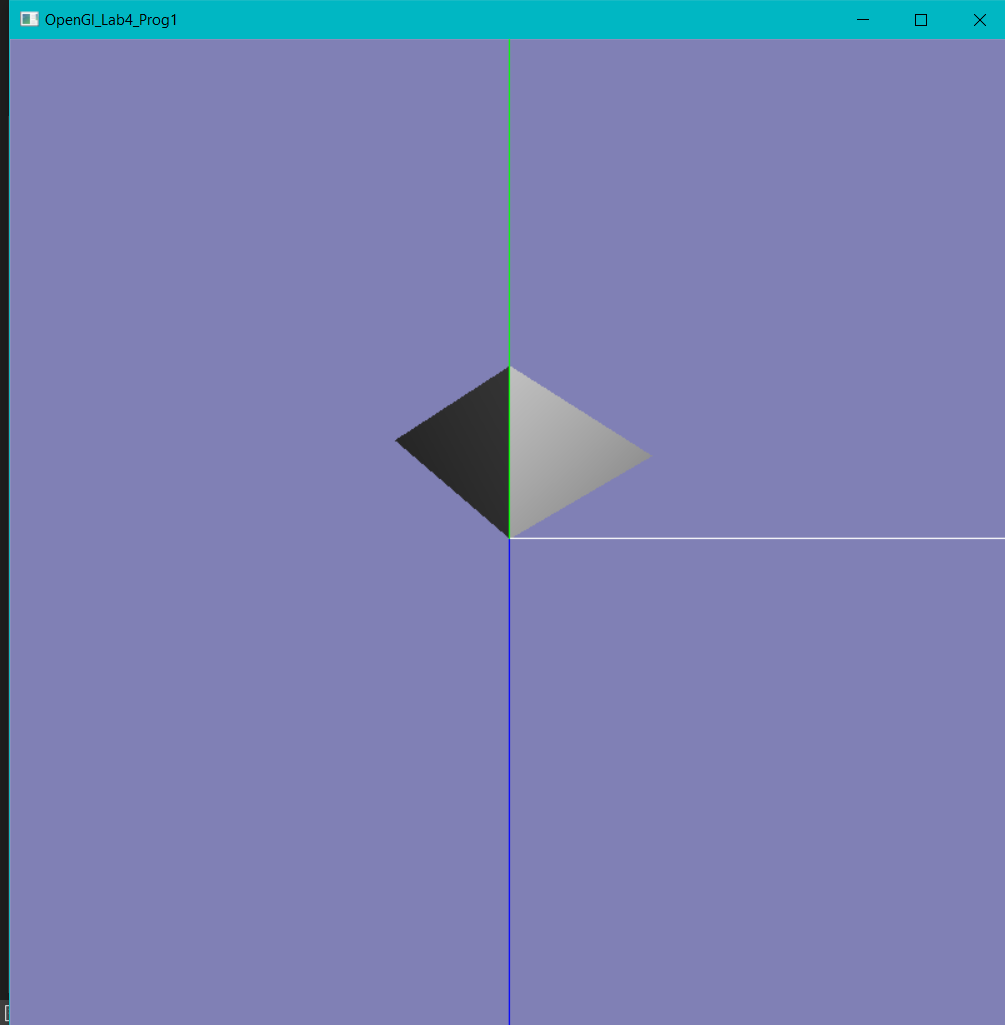
glutKeyboardFunc(KeyboardEvents);

init();

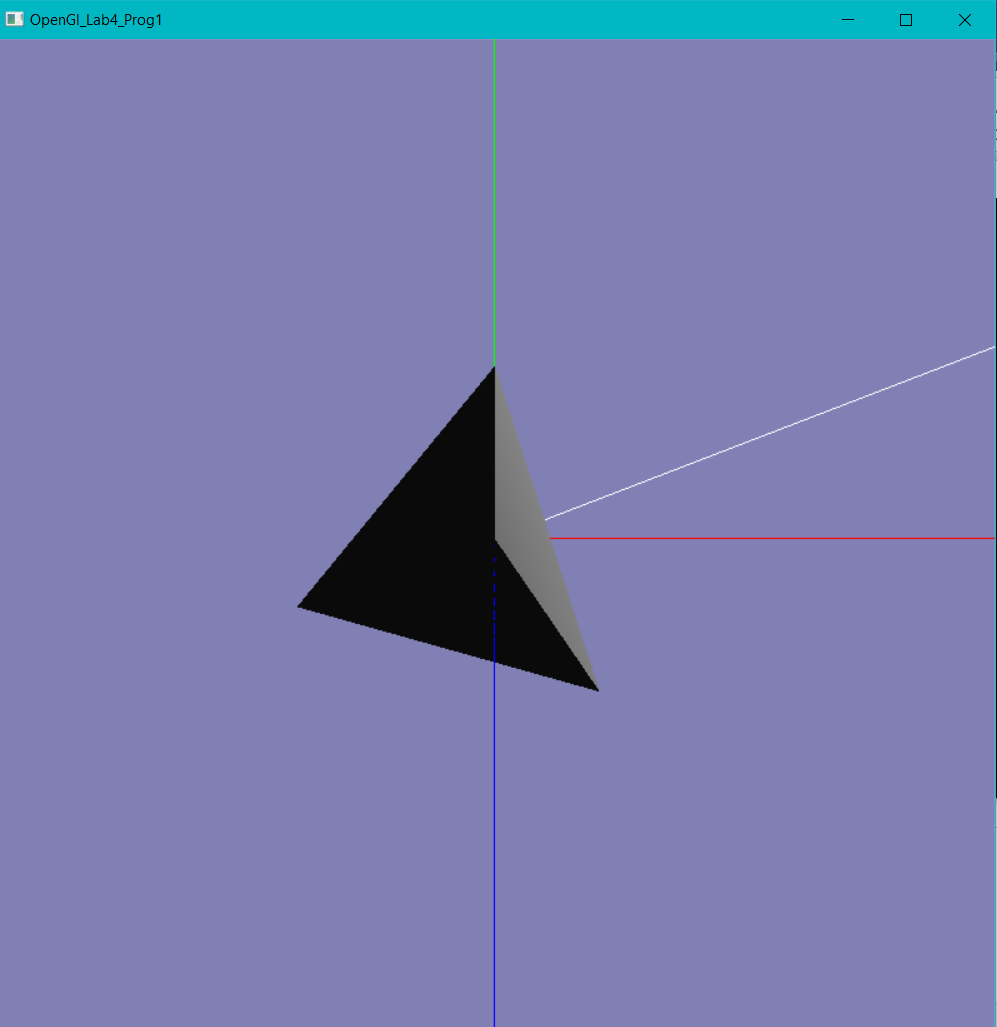
glutMainLoop();

}

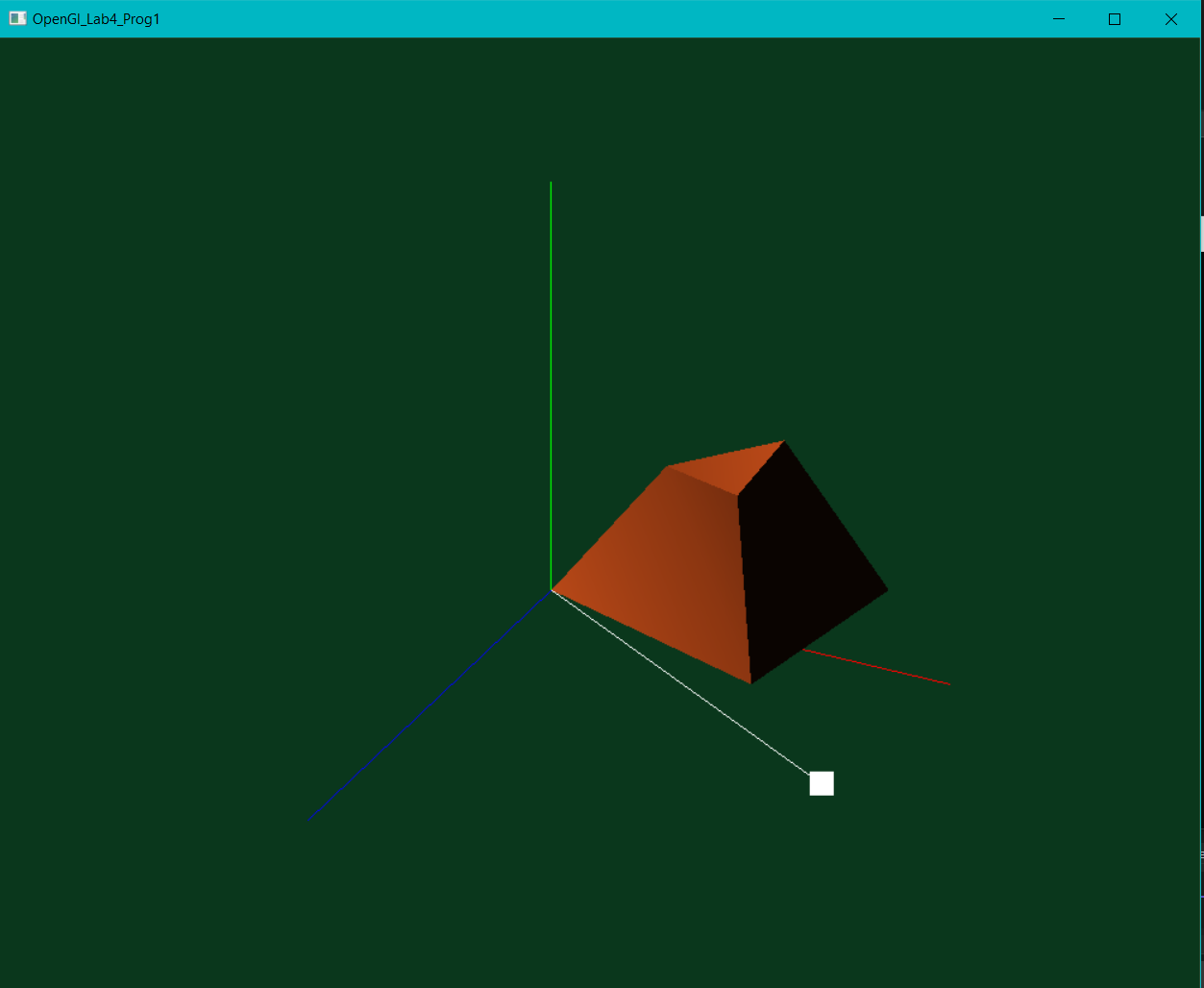
1. **Работа программы:**
2. Вращение объекта вокруг оси Y (клавиша ‘o’) и отображение лицевых граней (клавиша ‘f’).



Вращение источника света вокруг оси Y (клавиша ‘l’) и отображение нелицевых граней (клавиша ‘b’).



1. Вращение объекта вокруг оси Y (клавиша ‘o’) и отображение лицевых граней (клавиша ‘f’).



Вращение источника света вокруг оси Y (клавиша ‘l’) и отображение нелицевых граней (клавиша ‘b’).

