Белов В.А., группа 4210

Лабораторная работа No4 (Отчет)

Задание 1.

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

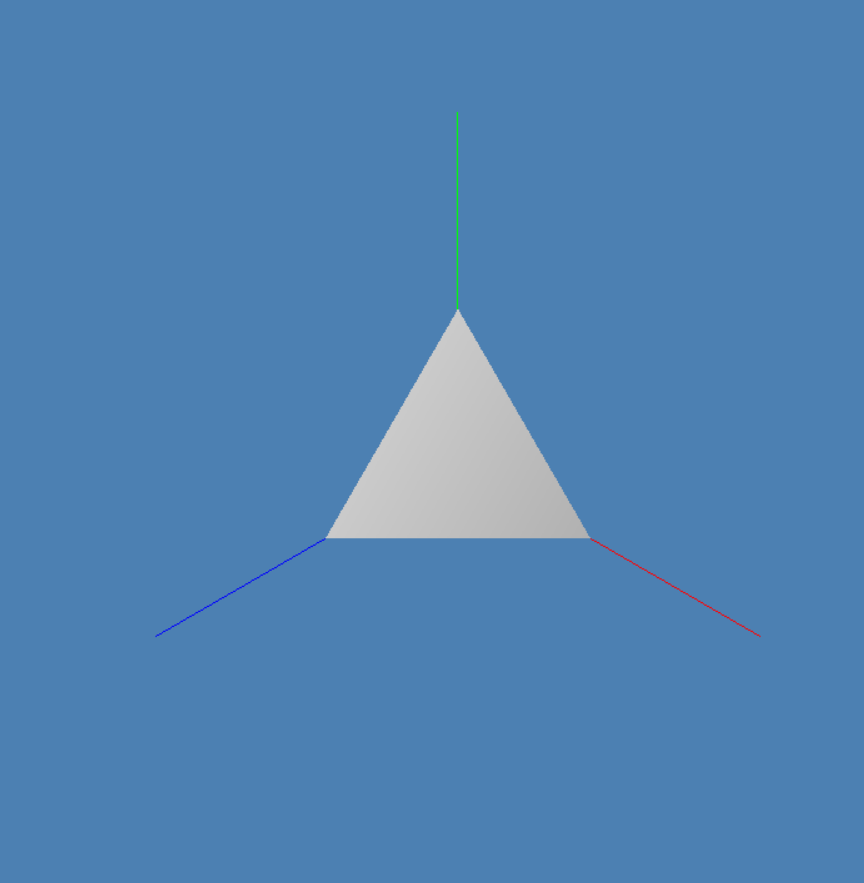
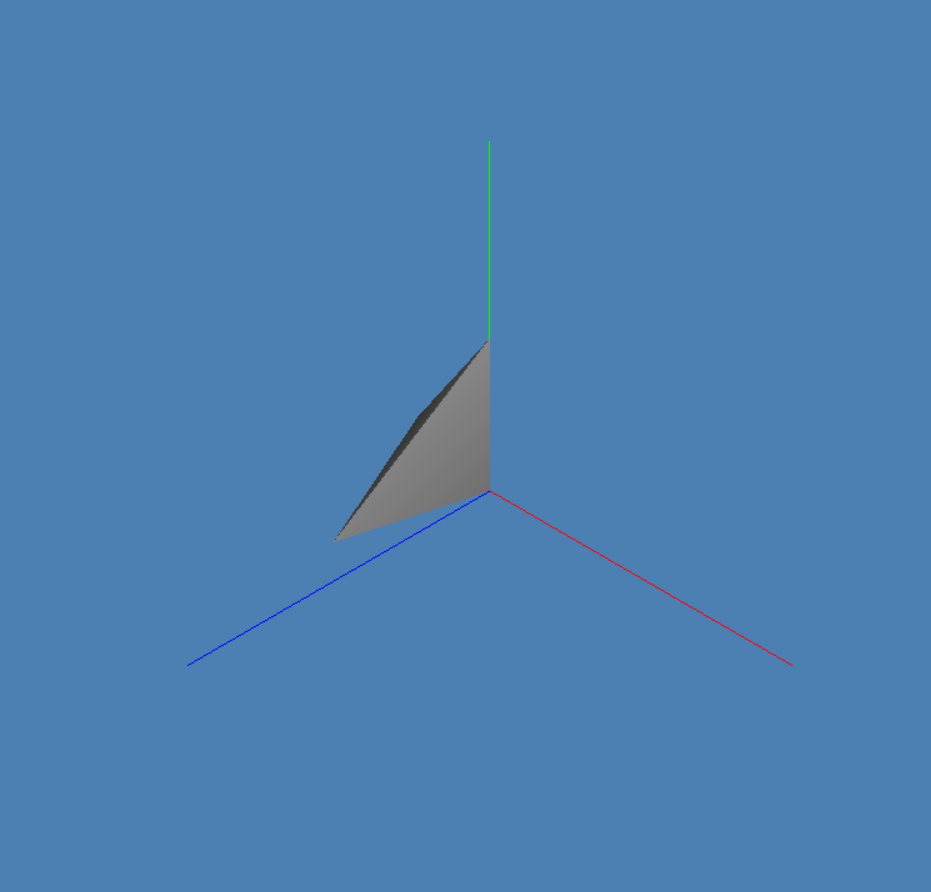
помощи обработчика отсутствия событий. При нажатии на клавишу ‘o’(‘l’) – вращается

объект (источник света).

7) Осуществить визуализацию лицевых (нелицевых) граней. При нажатии на клавишу

‘f’(‘b’) – визуализируются лицевые (нелицевые) грани.

Скриншот работы программы:

Листинг программы:

#include <gl/glut.h>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <windows.h>

GLfloat angle = 0, plusangle = 0;

GLfloat angle2 = 0, plusangle2 = 0;

int ch1 = 0, ch2 = 0;

void init(void)

{

glClearColor(0.3, 0.5, 0.7, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void myDisplay()

{

glPushMatrix(); //Сохраняем VM = 1

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); //вместе с

//очисткой буфера цвета предусмотреть очистку буфера глубины

glEnable(GL\_DEPTH\_TEST); //включение режима удаления невидимых

//(проверка глубины)

gluLookAt(3, 3, 3, 0, 0, 0, 0, 1, 0); // VM=Fwe // позиция наблюдателя

GLfloat myLightPosition[] = { 1.0, 2.0, 2.0, 1.0 }; // Источник света в CKw

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition); /\*Позиция

источника света будет преобразована в CKe\*/

glEnable(GL\_LIGHTING); // Включение расчета освещенности

glEnable(GL\_LIGHT0); // включаем этот конкретный источник

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle2, 0, 1, 0); // вращаем источник света

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

glPopMatrix(); // Восстанавливаем VM=Fwe

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle, 0, 1, 0); // VM=Fwe\*R

// грань f0

glBegin(GL\_POLYGON);

glColor3f(1,0,0);

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);

glColor3f(1, 0, 0);

glNormal3f(0, 0, -1); glVertex3f(0, 0, 0); glVertex3f(0, 1, 0); glVertex3f(1, 0, 0);

glEnd();

// грань f2

glBegin(GL\_POLYGON);

glColor3f(1, 0, 0);

glNormal3f(-1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(0, 0, 1); glVertex3f(0, 1, 0);

glEnd();

// грань f3

glBegin(GL\_POLYGON);

glColor3f(1, 0, 0);

glNormal3f(0, -1, 0); glVertex3f(1, 0, 0); glVertex3f(0, 0, 1); glVertex3f(0, 0, 0);

glEnd();

glPopMatrix(); // Восстанавливаем VM=Fwe

glDisable(GL\_LIGHTING); //Выключаем освещение

glBegin(GL\_LINES); //Рисуем координатные оси разного цвета в CKe

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(2, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 2, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 2);

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()

{

angle += plusangle; if (angle > 360.0) angle = 0;

angle2 += plusangle2; if (angle2 > 360.0) angle2 = 0;

glutPostRedisplay();

}

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

{

if (key == '0') {

if (ch1 % 2 == 0) plusangle = 0.1;

else plusangle = 0;

ch1++;

}

else if (key == 'l') {

if (ch2 % 2 == 0) plusangle2 = 0.1;

else plusangle2 = 0;

ch2++;

}

if (key == 'f') {

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == 'b') {

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

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

{

glutInit(&argc, argv);

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

//устанавливая режим дисплея в функции main(), предусмотрено использование буфера глубины

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("Лаба 4, Задание 1");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

init();

glutMainLoop();

}

Задание 2. (Вариант 3)

1) Построить полигональную сетку геометрического объекта, который указан в варианте

задания. Для этого необходимо заполнить таблицы списков: вершин, нормалей и

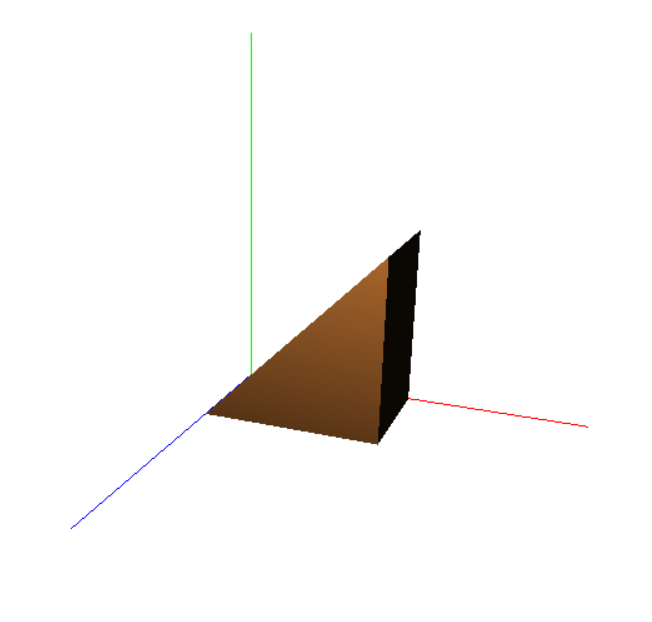
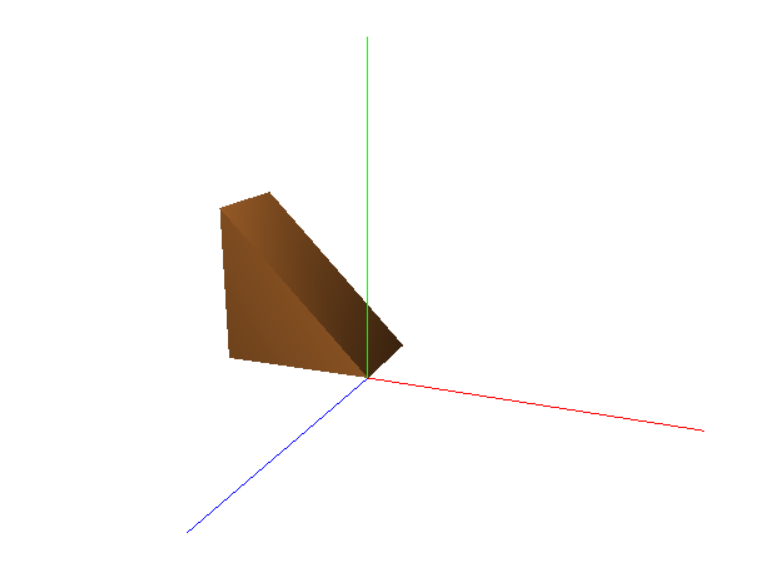
граней.

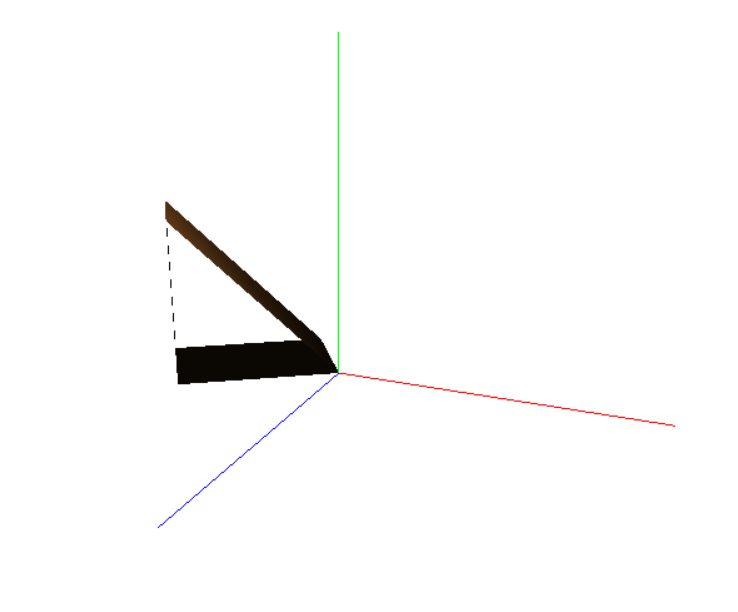
2) Вместо рисования тетраэдра осуществить рисование геометрического объекта по

заполненным таблицам.

3) Установить для лицевых граней объекта свойства материала, коэффициенты которого

приведены в соответствующем варианте задания.



Листинг программы:

#include <gl/glut.h>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <windows.h>

GLfloat angle = 0, plusangle = 0;

GLfloat angle2 = 0, plusangle2 = 0;

int ch1 = 0, ch2 = 0;

void init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void myDisplay()

{

glPushMatrix(); //Сохраняем VM = 1

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); //вместе с очисткой буфера цвета

//предусмотреть очистку буфера глубины

glEnable(GL\_DEPTH\_TEST); //включение режима удаления невидимых (проверка глубины)

gluLookAt(3, 3, 7, 0, 0, 0, 0, 1, 0); // VM=Fwe // позиция наблюдателя

GLfloat myLightPosition[] = { 1.0, 2.0, 2.0, 1.0 }; // Источник света в CKw

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition); /\*Позиция ис-точника света будет

преобразована в CKe\*/

glEnable(GL\_LIGHTING); // Включение расчета освещенности

glEnable(GL\_LIGHT0); // включаем этот конкретный источник

GLfloat myDiffuse[] = { 0.714, 0.4284, 0.18144, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

GLfloat myShininess[] = { 25.6 };

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

GLfloat myAmbient[] = { 0.2125, 0.1275, 0.054, 1 };

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

GLfloat mySpecular[] = { 0.393548, 0.271906, 0.166721, 1 };

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

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle2, 0, 1, 0); // вращаем источник света

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

glPopMatrix(); // Восстанавливаем VM=Fwe

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle, 0, 1, 0); // VM=Fwe\*R

// грань f0 (основание)

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(0, 0, 0); glVertex3f(1.5, 0, 0);

glVertex3f(1.5, 0, 1); glVertex3f(0, 0, 1);

glEnd();

// грань f1 (задняя стенка)

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(0, 0, 0); glVertex3f(1.5, 1.5, 0);

glVertex3f(1.5, 0, 0);

glEnd();

// грань f2 (передняя стенка)

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(0, 0, 1); glVertex3f(1.5, 1.5, 1);

glVertex3f(1.5, 0, 1);

glEnd();

// грань f4 (левая стенка)

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(0, 0, 0); glVertex3f(1.5, 1.5, 0);

glVertex3f(1.5, 1.5, 1); glVertex3f(0, 0, 1);

glEnd();

// грань f1 (правая стенка)

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glVertex3f(1.5, 0, 0); glVertex3f(1.5, 1.5, 0);

glVertex3f(1.5, 1.5, 1); glVertex3f(1.5, 0, 1);

glEnd();

glPopMatrix(); // Восстанавливаем VM=Fwe

glDisable(GL\_LIGHTING); //Выключаем освещение

glBegin(GL\_LINES); //Рисуем координатные оси разного цвета в CKe

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();

//Рисуем направление к источнику света в CKe

//glBegin(GL\_LINES); glColor3f(0,0,0); glVertex3f(1, 2, 2); glVertex3f(0.0, 0.0, 0.0); glEnd();

glPopMatrix();

glutSwapBuffers();

// usleep(50000);

}

void myReshape(int width, int height)

{

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

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

}

void myIdle()

{

angle += plusangle; if (angle > 360.0) angle = 0;

angle2 += plusangle2; if (angle2 > 360.0) angle2 = 0;

glutPostRedisplay();

}

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

{

if (key == '0') {

if (ch1 % 2 == 0) plusangle = 0.1;

else plusangle = 0;

ch1++;

}

else if (key == 'l') {

if (ch2 % 2 == 0) plusangle2 = 0.1;

else plusangle2 = 0;

ch2++;

}

if (key == 'f') {

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == 'b') {

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH); //устанавливая режим

//дисплея в функции main(), предусмотрено использование буфера глубины

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("Лаба 4, Задание 2 (Индивидуальное)");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

init();

glutMainLoop();

}