**Задание**

1. Осуществите рисование объекта, созданного в предыдущей лабораторной работе, при помощи дисплейного списка.
2. Наложите на объект три различные текстуры на три различные грани. (Для этого необходимо предварительно создать текстурные изображения в формате \*.bmp.)
3. Осуществите изменение параметров отображения текстуры объекта таким образом, чтобы при нажатии на левую кнопку мыши объект излучал свет согласно заданной текстуре, при нажатии на правую кнопку мыши объект закрашивался текстурой, и при этом моделировались коэффициенты отражения.

**Листинг**

#include <gl/glut.h>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 5\\GLAUX.H>

#pragma comment(lib, "C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 5\\GLAUX.lib")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

GLfloat angle = 0, plusangle = 0;

GLfloat angle2 = 0, plusangle2 = 0;

int ch1 = 0, ch2 = 0;

GLuint list = 0, list2 = 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();

AUX\_RGBImageRec\* image = auxDIBImageLoadA("C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 5\\texture\_1.bmp"); //для загрузки в оперативную память изображения

AUX\_RGBImageRec\* image2 = auxDIBImageLoadA("C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 5\\texture\_2.bmp");

AUX\_RGBImageRec\* image3 = auxDIBImageLoadA("C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 5\\texture\_3.bmp");

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

list = glGenLists(1); //генерирование одного или более незадействованных индексов.

glNewList(list, GL\_COMPILE); //Определение дисплейного списка

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image->sizeX, image->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image->data);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

// грань f0 (основание)

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glTexCoord2f(0, 0); glVertex3f(0, 0, 0);

glTexCoord2f(0, 1); glVertex3f(1, 0, 0);

glTexCoord2f(1, 0); glVertex3f(1, 0, 1);

glTexCoord2f(1, 1); glVertex3f(0, 0, 1);

glEnd();

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image3->sizeX, image3->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, image3->data);

// грань f1 (задняя стенка)

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glTexCoord2f(0, 0); glVertex3f(0, 0, 0);

glTexCoord2f(0, 1); glVertex3f(0.5, 1, 0);

glTexCoord2f(1, 0); glVertex3f(1, 1, 0);

glTexCoord2f(1, 1); glVertex3f(1, 0, 0);

glEnd();

// грань f2 (передняя стенка)

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glTexCoord2f(0, 0); glVertex3f(0, 0, 1);

glTexCoord2f(0, 1); glVertex3f(0.5, 1, 1);

glTexCoord2f(1, 0); glVertex3f(1, 1, 1);

glTexCoord2f(1, 1); glVertex3f(1, 0, 1);

glEnd();

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image2->sizeX, image2->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, image2->data);

// грань f3 (верхушка)

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0);

glTexCoord2f(0, 0); glVertex3f(0.5, 1, 0);

glTexCoord2f(0, 1); glVertex3f(1, 1, 0);

glTexCoord2f(1, 0); glVertex3f(1, 1, 1);

glTexCoord2f(1, 1); glVertex3f(0.5, 1, 1);

glEnd();

// грань f4 (левая стенка)

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glTexCoord2f(0, 0); glVertex3f(0, 0, 0);

glTexCoord2f(0, 1); glVertex3f(0.5, 1, 0);

glTexCoord2f(1, 0); glVertex3f(0.5, 1, 1);

glTexCoord2f(1, 1); glVertex3f(0, 0, 1);

glEnd();

// грань f5 (правая стенка)

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glTexCoord2f(0, 0); glVertex3f(1, 0, 0);

glTexCoord2f(0, 1); glVertex3f(1, 1, 0);

glTexCoord2f(1, 0); glVertex3f(1, 1, 1);

glTexCoord2f(1, 1); glVertex3f(1, 0, 1);

glEnd();

glEndList();

list2 = glGenLists(1); //генерирование одного или более незадействованных индексов.

glNewList(list2, GL\_COMPILE);

glBegin(GL\_LINES);

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();

glEndList();

}

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

glPushMatrix(); //Сохраняем VM=Fwe

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

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

glBegin(GL\_LINES); // Рисуем направление к источнику света в CKe

glColor3f(1, 1, 1); glVertex3f(myLightPosition[0], myLightPosition[1],

myLightPosition[2]); glVertex3f(0.0, 0.0, 0.0);

glEnd();

glPopMatrix(); // Восстанавливаем VM=Fwe

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

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

glPushMatrix(); //Сохраняем VM=Fwe

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

GLfloat myDiffuse[] = { 0.50754, 0.50754, 0.50754, 1 };

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

GLfloat myShininess[] = { 51.2 };

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

GLfloat myAmbient[] = { 0.19225, 0.19225, 0.19225, 1 };

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

GLfloat mySpecular[] = { 0.508273, 0.508273, 0.508273, 1 };

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

glCallList(list); //установка состояния, и команды рисования

glPopMatrix(); // Восстанавливаем VM=Fwe

glDisable(GL\_LIGHTING); //Выключаем освещение

glCallList(list2); //установка состояния, и команды рисования

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);

}

}

void mouseButton(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON) {

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL); //объект излучает свет согласно заданной текстуре

}

else if (button == GLUT\_RIGHT\_BUTTON) {

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE); //объект излучет свет согласно заданной текстуре

}

}

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 5");

glutDisplayFunc(myDisplay);

glutMouseFunc(mouseButton);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

init();

glutMainLoop();

}

**Выполнение программы**





