**Задание**

Создать любой логотип, используя полученные навыки при разработке лабораторных работ.

Для логотипа я выбрал свои инициалы (ВИ – Валиев Ильдар)

**Листинг**

#include <iostream>

#include <gl\glut.h>

#include <atlbase.h>

#include <atlconv.h>

#include <string>

#include <cmath>

#include <math.h>

#include <Windows.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;

int ch1 = 0;

GLuint list = 0, list2 = 0;

void init(void)

{

glClearColor(1, 1, 1, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

AUX\_RGBImageRec\* image = auxDIBImageLoadA("image1.bmp"); //для загрузки в оперативную память изображения

AUX\_RGBImageRec\* image2 = auxDIBImageLoadA("image2.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

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

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

GL\_UNSIGNED\_BYTE, image->data);

//f0(передняя стенка)

//В

glBegin(GL\_POLYGON);

glVertex3f(0, 0, 0.5); glVertex3f(0.4, 0, 0.5);

glVertex3f(0.4, 2.5, 0.5); glVertex3f(0, 2.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 2.5, 0.5); glVertex3f(1.05, 2.5, 0.5);

glVertex3f(1.05, 2, 0.5); glVertex3f(0.4, 2, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 1.5, 0.5); glVertex3f(1.1, 1.5, 0.5);

glVertex3f(1.1, 1, 0.5); glVertex3f(0.4, 1, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 0.5, 0.5); glVertex3f(1.1, 0.5, 0.5);

glVertex3f(1.1, 0, 0.5); glVertex3f(0.4, 0, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.5, 0.4, 0.5); glVertex3f(1.5, 1.1, 0.5);

glVertex3f(0.9, 1.1, 0.5); glVertex3f(0.9, 0.4, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.1, 0, 0.5); glVertex3f(1.5, 0.4, 0.5);

glVertex3f(1.1, 0.4, 0.5); glVertex3f(1.05, 0.4, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.45, 0.5); glVertex3f(1.5, 1.1, 0.5);

glVertex3f(1.1, 1.1, 0.5); glVertex3f(1.1, 1, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.4, 2.1, 0.5); glVertex3f(1.4, 1.4, 0.5);

glVertex3f(0.9, 1.4, 0.5); glVertex3f(0.9, 2.1, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 2.5, 0.5); glVertex3f(1.4, 2.1, 0.5);

glVertex3f(1.05, 2.1, 0.5); glVertex3f(1.05, 2, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.5, 0.5); glVertex3f(1.4, 1.4, 0.5);

glVertex3f(1.05, 1.1, 0.5); glVertex3f(1.05, 1, 0.5);

glEnd();

//И

glBegin(GL\_POLYGON);

glVertex3f(2, 0, 0.5); glVertex3f(2.4, 0, 0.5);

glVertex3f(2.4, 2.5, 0.5); glVertex3f(2, 2.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(3, 0, 0.5); glVertex3f(3.4, 0, 0.5);

glVertex3f(3.4, 2.5, 0.5); glVertex3f(3, 2.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(2.4, 0, 0.5); glVertex3f(2.4, 1, 0.5);

glVertex3f(3, 2.5, 0.5); glVertex3f(3, 1.5, 0.5);

glEnd();

//f1(задняя стенка)

//В

glBegin(GL\_POLYGON);

glVertex3f(0, 0, 0); glVertex3f(0.4, 0, 0);

glVertex3f(0.4, 2.5, 0); glVertex3f(0, 2.5, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 2.5, 0); glVertex3f(1.05, 2.5, 0);

glVertex3f(1.05, 2, 0); glVertex3f(0.4, 2, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 1.5, 0); glVertex3f(1.1, 1.5, 0);

glVertex3f(1.1, 1, 0); glVertex3f(0.4, 1, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.4, 0.5, 0); glVertex3f(1.1, 0.5, 0);

glVertex3f(1.1, 0, 0); glVertex3f(0.4, 0, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.5, 0.4, 0); glVertex3f(1.5, 1.1, 0);

glVertex3f(0.9, 1.1, 0); glVertex3f(0.9, 0.4, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.1, 0, 0); glVertex3f(1.5, 0.4, 0);

glVertex3f(1.1, 0.4, 0); glVertex3f(1.05, 0.4, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.45, 0); glVertex3f(1.5, 1.1, 0);

glVertex3f(1.1, 1.1, 0); glVertex3f(1.1, 1, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.4, 2.1, 0); glVertex3f(1.4, 1.4, 0);

glVertex3f(0.9, 1.4, 0); glVertex3f(0.9, 2.1, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 2.5, 0); glVertex3f(1.4, 2.1, 0);

glVertex3f(1.05, 2.1, 0); glVertex3f(1.05, 2, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.5, 0); glVertex3f(1.4, 1.4, 0);

glVertex3f(1.05, 1.1, 0); glVertex3f(1.05, 1, 0);

glEnd();

//И

glBegin(GL\_POLYGON);

glVertex3f(2, 0, 0); glVertex3f(2.4, 0, 0);

glVertex3f(2.4, 2.5, 0); glVertex3f(2, 2.5, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(3, 0, 0); glVertex3f(3.4, 0, 0);

glVertex3f(3.4, 2.5, 0); glVertex3f(3, 2.5, 0);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(2.4, 0, 0); glVertex3f(2.4, 1, 0);

glVertex3f(3, 2.5, 0); glVertex3f(3, 1.5, 0);

glEnd();

//f2(основание)

//В

glBegin(GL\_POLYGON);

glVertex3f(0, 0, 0); glVertex3f(1.1, 0, 0);

glVertex3f(1.1, 0, 0.5); glVertex3f(0, 0, 0.5);

glEnd();

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

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

glBegin(GL\_POLYGON);

glVertex3f(1.05, 2, 0); glVertex3f(0.4, 2, 0);

glVertex3f(0.4, 2, 0.5); glVertex3f(1.05, 2, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.1, 1, 0); glVertex3f(0.4, 1, 0);

glVertex3f(0.4, 1, 0.5); glVertex3f(1.1, 1, 0.5);

glEnd();

//И

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

GL\_UNSIGNED\_BYTE, image->data);

glBegin(GL\_POLYGON);

glVertex3f(2, 0, 0); glVertex3f(2.4, 0, 0);

glVertex3f(2.4, 0, 0.5); glVertex3f(2, 0, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(3, 0, 0); glVertex3f(3.4, 0, 0);

glVertex3f(3.4, 0, 0.5); glVertex3f(3, 0, 0.5);

glEnd();

//f3(верхняя стенка)

//В

glBegin(GL\_POLYGON);

glVertex3f(0, 2.5, 0); glVertex3f(1.1, 2.5, 0);

glVertex3f(1.1, 2.5, 0.5); glVertex3f(0, 2.5, 0.5);

glEnd();

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

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

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.5, 0); glVertex3f(0.4, 1.5, 0);

glVertex3f(0.4, 1.5, 0.5); glVertex3f(1.05, 1.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.1, 0.5, 0); glVertex3f(0.4, 0.5, 0);

glVertex3f(0.4, 0.5, 0.5); glVertex3f(1.1, 0.5, 0.5);

glEnd();

//И

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

GL\_UNSIGNED\_BYTE, image->data);

glBegin(GL\_POLYGON);

glVertex3f(2, 2.5, 0); glVertex3f(2.4, 2.5, 0);

glVertex3f(2.4, 2.5, 0.5); glVertex3f(2, 2.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(3, 2.5, 0); glVertex3f(3.4, 2.5, 0);

glVertex3f(3.4, 2.5, 0.5); glVertex3f(3, 2.5, 0.5);

glEnd();

//f4(левая стенка)

//В

glBegin(GL\_POLYGON);

glVertex3f(0, 0, 0); glVertex3f(0, 2.5, 0);

glVertex3f(0, 2.5, 0.5); glVertex3f(0, 0, 0.5);

glEnd();

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

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

glBegin(GL\_POLYGON);

glVertex3f(0.9, 1.1, 0); glVertex3f(0.9, 0.4, 0);

glVertex3f(0.9, 0.4, 0.5); glVertex3f(0.9, 1.1, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.9, 1.4, 0); glVertex3f(0.9, 2.1, 0);

glVertex3f(0.9, 2.1, 0.5); glVertex3f(0.9, 1.4, 0.5);

glEnd();

//И

glBegin(GL\_POLYGON);

glVertex3f(3, 0, 0); glVertex3f(3, 2.5, 0);

glVertex3f(3, 2.5, 0.5); glVertex3f(3, 0, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(2.4, 1, 0); glVertex3f(3, 2.5, 0);

glVertex3f(3, 2.5, 0.5); glVertex3f(2.4, 1, 0.5);

glEnd();

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

GL\_UNSIGNED\_BYTE, image->data);

glBegin(GL\_POLYGON);

glVertex3f(2, 0, 0); glVertex3f(2, 2.5, 0);

glVertex3f(2, 2.5, 0.5); glVertex3f(2, 0, 0.5);

glEnd();

//f5 (правая стенка)

//В

glBegin(GL\_POLYGON);

glVertex3f(1.5, 0.4, 0); glVertex3f(1.5, 1.1, 0);

glVertex3f(1.5, 1.1, 0.5); glVertex3f(1.5, 0.4, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.4, 2.1, 0); glVertex3f(1.4, 1.4, 0);

glVertex3f(1.4, 1.4, 0.5); glVertex3f(1.4, 2.1, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.1, 0, 0); glVertex3f(1.5, 0.4, 0);

glVertex3f(1.5, 0.4, 0.5);glVertex3f(1.1, 0, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 1.45, 0); glVertex3f(1.5, 1.1, 0);

glVertex3f(1.5, 1.1, 0.5); glVertex3f(1.05, 1.45, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.05, 2.5, 0); glVertex3f(1.4, 2.1, 0);

glVertex3f(1.4, 2.1, 0.5); glVertex3f(1.05, 2.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(1.4, 1.4, 0.5); glVertex3f(1.4, 1.4, 0);

glVertex3f(1.05, 1.1, 0); glVertex3f(1.05, 1.1, 0.5);

glEnd();

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

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

glBegin(GL\_POLYGON);

glVertex3f(0.4, 0.5, 0);glVertex3f(0.4, 2, 0);

glVertex3f(0.4, 2, 0.5);glVertex3f(0.4, 0.5, 0.5);

glEnd();

//И

glBegin(GL\_POLYGON);

glVertex3f(2.4, 0, 0); glVertex3f(2.4, 2.5, 0);

glVertex3f(2.4, 2.5, 0.5); glVertex3f(2.4, 0, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(2.4, 0, 0); glVertex3f(3, 1.5, 0);

glVertex3f(3, 1.5, 0.5); glVertex3f(2.4, 0, 0.5);

glEnd();

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

GL\_UNSIGNED\_BYTE, image->data);

glBegin(GL\_POLYGON);

glVertex3f(3.4, 0, 0); glVertex3f(3.4, 2.5, 0);

glVertex3f(3.4, 2.5, 0.5); glVertex3f(3.4, 0, 0.5);

glEnd();

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

//индексов.

glNewList(list2, GL\_COMPILE);

glEndList();

}

void myDisplay()

{

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

glPushMatrix(); //Сохраняем VM = 1

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

//цвета предусмотреть очистку буфера 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

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

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[] = { 0.75164, 0.60648, 0.22648, 1 };

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

GLfloat myShininess[] = { 51.2 };

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

GLfloat myAmbient[] = { 0.24725, 0.1995, 0.0745, 1 };

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

GLfloat mySpecular[] = { 0.628281, 0.555802, 0.366065, 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;

glutPostRedisplay();

}

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

{

if (key == '0') {

if (ch1 % 2 == 0) plusangle = 1;

else plusangle = 0;

ch1++;

}

}

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

{

setlocale(LC\_ALL, "");

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("Лого");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

init();

glutMainLoop();

}

**Выполнение программы**



