#include "glut.h"

#include "math.h"

#include <iostream>

#include <fstream>

using namespace std;

#pragma comment(lib, "glut32.lib")

#define GL\_PI 3.1415f

int LINE = GL\_POLYGON;

int p = 30;

static GLfloat xAngle = 0.0f;

static GLfloat yAngle = 0.0f;

static GLfloat zAngle = 0.0f;

/\*чтение текстур\*/

GLuint LoadTexture(char\* filename, GLint xsize, GLint ysize)

{

GLuint texture\_;

unsigned char\* data = nullptr;

ifstream file(filename, ios\_base::in | ios\_base::binary);

if (!file) return 0;

data = (unsigned char\*)malloc(xsize \* ysize \* 3);

file.seekg(54);

file.read((char\*)data, xsize \* ysize \* 3);

file.close();

for (int i = 0; i < xsize \* ysize; i++)

{

unsigned int index = i \* 3;

unsigned char B, R;

B = data[index];

R = data[index + 2];

data[index] = R;

data[index + 2] = B;

}

glGenTextures(1, &texture\_);

glBindTexture(GL\_TEXTURE\_2D, texture\_);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR\_MIPMAP\_LINEAR);

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, xsize, ysize, GL\_RGB, GL\_UNSIGNED\_BYTE, data);

free(data);

return texture\_;

}

GLuint\* textures = new GLuint[1];

bool flag = false;

void DrawFlatZ(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat z1, GLfloat z2)

{

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(x1, z1, y1);

glTexCoord2d(0.0, 1.0);

glVertex3f(x2, z2, y1);

glTexCoord2d(1.0, 1.0);

glVertex3f(x2, z2, y2);

glTexCoord2d(1.0, 0.0);

glVertex3f(x1, z1, y2);

glEnd();

}

void DrawFlatY(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat z1, GLfloat z2, GLfloat xx)

{

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(x1, y1, z1);

glTexCoord2d(0.0, 1.0);

glVertex3f(x2, y1, z1);

glTexCoord2d(1.0, 1.0);

glVertex3f(x2 + xx, y2, z2);

glTexCoord2d(1.0, 0.0);

glVertex3f(x1 + xx, y2, z2);

glEnd();

}

void draw()

{

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(-60, 10, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(0, 55, 0);

glTexCoord2d(1.0, 0.0);

glVertex3f(60, 10, 0);

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(0, -25, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(60, 10, 0);

glTexCoord2d(1.0, 0.0);

glVertex3f(40, -50, 0);

glEnd();

//передняя грань

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 50);

glTexCoord2d(0.0, 1.0);

glVertex3f(-60, 10, 50);

glTexCoord2d(1.0, 1.0);

glVertex3f(0, 55, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(60, 10, 50);

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 50);

glTexCoord2d(0.0, 1.0);

glVertex3f(0, -25, 50);

glTexCoord2d(1.0, 1.0);

glVertex3f(60, 10, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(40, -50, 50);

glEnd();

///

//левый низ

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(-60, 10, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(-60, 10, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(-40, -50, 50);

glEnd();

//левый верх

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-60, 10, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(0, 55, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(0, 55, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(-60, 10, 50);

glEnd();

//правый верх

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(0, 55, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(60, 10, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(60, 10, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(0, 55, 50);

glEnd();

//правй низ

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(60, 10, 0);

glTexCoord2d(0.0, 1.0);

glVertex3f(40, -50, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(40, -50, 50);

glTexCoord2d(1.0, 0.0);

glVertex3f(60, 10, 50);

glEnd();

//низ

glBegin(LINE);

glTexCoord2d(0.0, 0.0);

glVertex3f(-40, -50, 50);

glTexCoord2d(0.0, 1.0);

glVertex3f(-40, -50, 0);

glTexCoord2d(1.0, 1.0);

glVertex3f(40, -50, 0);

glTexCoord2d(1.0, 0.0);

glVertex3f(40, -50, 50);

glEnd();

}

void drawLine()

{

glBegin(GL\_LINE\_LOOP);

glColor3f(0, 0, 0);

glLineWidth(5);

glVertex3f(-40.1, -50.1, 0.1);

glVertex3f(-60.1, 10.1, 0.1);

glVertex3f(0.1, 55.1, 0.1);

glVertex3f(60.1, 10.1, 0.1);

glVertex3f(40.1, -50.1, 0.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glColor3f(0, 0, 0);

glLineWidth(5);

glVertex3f(-40.1, -50.1, 50.1);

glVertex3f(-60.1, 10.1, 50.1);

glVertex3f(0.1, 55.1, 50.1);

glVertex3f(60.1, 10.1, 50.1);

glVertex3f(40.1, -50.1, 50.1);

glEnd();

/\*glBegin(GL\_LINES);

glVertex3f(-40.1, -50.1, 0);

glVertex3f(-40.1, -50.1, 50);

glVertex3f(-60.1, 10.1, 0);

glVertex3f(-60.1, 10.1, 50);

glVertex3f(0.1, 55.1, 0);

glVertex3f(0.1, 55.1, 50);

glVertex3f(60.1, 10.1, 0);

glVertex3f(60.1, 10.1, 50);

glVertex3f(40.1, -50.1, 0);

glVertex3f(40.1, -50.1, 50);\*/

//DrawFlatZ(0, -15, 0, 15, -125, -145);

}

void load\_text()

{

if (flag) return;

textures[0] = LoadTexture((char\*)"texture.bmp", 450, 450);

cout << textures[0];

flag = true;

}

void RenderScene(void)

{

load\_text();

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

glLoadIdentity();

glEnable(GL\_DEPTH\_TEST);

glPushMatrix();

glRotatef(xAngle, 1, 0, 0);

glRotatef(yAngle, 0, 1, 0);

glRotatef(zAngle, 0, 0, 1);

glColor3f(1, 1, 1);

glLineWidth(9);

glEnable(GL\_TEXTURE\_2D);

LINE = GL\_POLYGON;

draw();//полигоны

glDisable(GL\_TEXTURE\_2D);

LINE = GL\_LINE\_LOOP;

glColor3f(0.0, 0.0, 0);

drawLine();//ребра

glPopMatrix();

glutSwapBuffers();

}

void SetupRC()

{

glClearColor(0.5f, 0.1f, 0.6f, 1);

}

void SpecialKeys(int key, int x, int y)

{

if (key == GLUT\_KEY\_UP)

xAngle -= 5.0f;

if (key == GLUT\_KEY\_DOWN)

xAngle += 5.0f;

if (key == GLUT\_KEY\_LEFT)

yAngle -= 5.0f;

if (key == GLUT\_KEY\_RIGHT)

yAngle += 5.0f;

if (key == GLUT\_KEY\_F1)

zAngle -= 5.0f;

if (key == GLUT\_KEY\_F2)

zAngle += 5.0f;

if (key > 355.0f)

xAngle = 0.0f;

if (key < 0.0f)

xAngle = 355.0f;

if (key > 355.0f)

yAngle = 0.0f;

if (key < 0.0f)

yAngle = 355.0f;

if (key > 355.0f)

zAngle = 0.0f;

if (key < 0.0f)

zAngle = 355.0f;

glutPostRedisplay();

}

void ChangeSize(int w, int h)

{

GLfloat nRange = 120.0f;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(-nRange, nRange, -nRange \* h / w, nRange \* h / w, -nRange, nRange);

else

glOrtho(-nRange \* w / h, nRange \* w / h, -nRange, nRange, -nRange, nRange);

// Reset Model view matrix stack

glMatrixMode(GL\_MODELVIEW);

}

int XMOUSE = 0, YMOUSE = 0, ZMOUSE = 0;

int X\_OLD = 0, Y\_OLD = 0, Z\_OLD = 0;

void Mouse\_(int ax, int ay)

{

XMOUSE = ax;

YMOUSE = ay;

if (XMOUSE > X\_OLD)

{

yAngle -= 2.0f;

}

else if (XMOUSE < X\_OLD)

{

yAngle += 2.0f;

}

if (YMOUSE > Y\_OLD)

{

xAngle += 2.0f;

}

else if (YMOUSE < Y\_OLD)

{

xAngle -= 2.0f;

}

X\_OLD = XMOUSE;

Y\_OLD = YMOUSE;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitWindowPosition(0, 0);

glutInitWindowSize(1366, 768);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutCreateWindow("Lab3");

glutReshapeFunc(ChangeSize);

glutSpecialFunc(SpecialKeys);

glutDisplayFunc(RenderScene);

glutPassiveMotionFunc(Mouse\_);

SetupRC();

glutMainLoop();

return 0;

}