

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт информационных систем и технологий

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Программирование специализированных вычислительных устройств

Отчет по лабораторной работе

«Моделирование 3D объектов с использованием Visual C++ и OpenGL»

вариант № 7

Выполнил студент гр. ИДБ-22-06 Мустафаева П.М.

Проверил преподаватель Лаверычев М.А.

Москва 2023г.

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Изучить теоретические основы OpenGL и построить 3D-объект в соответствии со своим вариантом.



Рис. 1.

**ПРИМЕНЯЕМЫЕ ТЕХНОЛОГИИ**

Visual Studio 2022, библиотека windows.h., библиотека iostream.

**РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАНИЯ**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<windows.h>

#include<D:/Мустафаева/Project2/glut.h>

#include <iostream>

const float light0\_position[4] = { 1, 0, 1, 0 };//положение источника освещения

double rotate\_y = 0;//вращение сцены

double rotate\_x = 0;

double rotate = 0;

void specialKeys(int key, int x, int y) {

if (key == GLUT\_KEY\_RIGHT) {

rotate\_y += 5;

}

else if (key == GLUT\_KEY\_LEFT) {

rotate\_y -= 5;

}

else if (key == GLUT\_KEY\_UP)

rotate\_x -= 5;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x += 5;

glutPostRedisplay();

}

GLuint groundTex;

GLuint LoadTexture(const char\* filename)//загрузка текстуры из файла

{//Функция считывания текстуры из файла

GLuint texture;

int width, height;

unsigned char\* data;

FILE\* file;

file = fopen(filename, "rb");

if (file == NULL) return 0; //Не считывать несуществующий файл

width = 96; //Размер изображения в пикселях

height = 96;

data = (unsigned char\*)malloc(width \* height \* 3);

fread(data, width \* height \* 3, 1, file);

fclose(file);

for (int i = 0; i < width \* height; ++i)

{

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\_NEAREST);

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, width, height, GL\_RGB, GL\_UNSIGNED\_BYTE, data);

free(data);

return texture;

}

void Ground() { // Отображение поверхности стола

groundTex = LoadTexture("Okume.bmp");

glEnable(GL\_TEXTURE\_2D);

glBindTexture(GL\_TEXTURE\_2D, groundTex);

glColor3f(0.6f, 0.5f, 0.3f);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f);

glVertex3f(0.5f, -0.5f, -0.5f);

glTexCoord2f(1.0f, 0.0f);

glVertex3f(-0.5f, -0.5f, -0.5f);

glTexCoord2f(1.0f, 1.0f);

glVertex3f(-0.5f, -0.5f, 0.5f);

glTexCoord2f(0.0f, 1.0f);

glVertex3f(0.5, -0.5f, 0.5f);

glEnd();

glDisable(GL\_POLYGON\_OFFSET\_FILL);

glBindTexture(GL\_TEXTURE\_2D, 0);

glDisable(GL\_TEXTURE\_2D);

}

void Display()

{

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

float x = 0.0;

float y = 0.0;

float z = 0.0;

glLoadIdentity();

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

Ground();

glPushMatrix(); // центр

glColor3f(0.0f, 0.0f, 0.0f);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x, y , z );

glScalef(0.05f, 0.05f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

// Корпус //

glPushMatrix();

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x, y, z);

GLUquadric\* obj = gluNewQuadric();//для цилиндра

glColor4f(0.0f, 1.0f, 1.0f, 1.0f);

gluDisk(obj, 0, 0.3, 100, 100);

gluCylinder(obj, 0.3, 0.3, 0.1, 100, 100);//отрисовка цилиндра

glPopMatrix();

glPushMatrix(); // Минутная стрелка

glColor3ub(0, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x+0.09, y, z);

glScalef(0.25f, 0.02f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); // Часы

glColor3ub(0, 0, 0);

glRotatef(180, 0.0, 0.0, 0.0);

glTranslatef(x , y + 0.09, z);

glScalef(0.03f, 0.22f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); // 12

glColor3ub(0, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x, y-0.25 , z);

glScalef(0.02f, 0.07f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //1

glColor3ub(180, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x+0.15, y-0.21, z );

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //2

glColor3ub(180, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x +0.23 , y +0.12, z );

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //3

glColor3ub(0, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x - 0.25, y , z );

glScalef(0.07f, 0.02f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //4

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x - 0.23, y + 0.12, z );

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //5

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x + 0.15, y + 0.21, z );

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //6

glColor3ub(0, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x, y + 0.25, z);

glScalef(0.02f, 0.07f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //7

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x - 0.15, y + 0.21, z );

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //8

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x + 0.23, y - 0.12, z);

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //9

glColor3ub(0, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x + 0.25, y, z);

glScalef(0.07f, 0.02f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //10

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x - 0.23, y - 0.12, z);

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glPushMatrix(); //11

glColor3ub(255, 0, 0);

glRotatef(180, 1.0, 0.0, 0.0);

glTranslatef(x - 0.15, y - 0.21, z + +0.005);

glScalef(0.03f, 0.03f, 0.05f);

glutSolidCube(0.80f);

glPopMatrix();

glFlush();

glutSwapBuffers();

}

void Initialize()

{

glClearColor(1.0, 1.0, 1.0, 0.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1, 1, -1, 1, 0, 0);

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(800, 800);

glutInitWindowPosition(100, 200);

glutCreateWindow("Часы");

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightModelf(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_position);

float ambient[4] = { 0.5, 0.5, 0.7, 1 };

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient);

glutDisplayFunc(Display);

glutSpecialFunc(specialKeys);

Initialize();

glutMainLoop();

return 0;

}

**РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

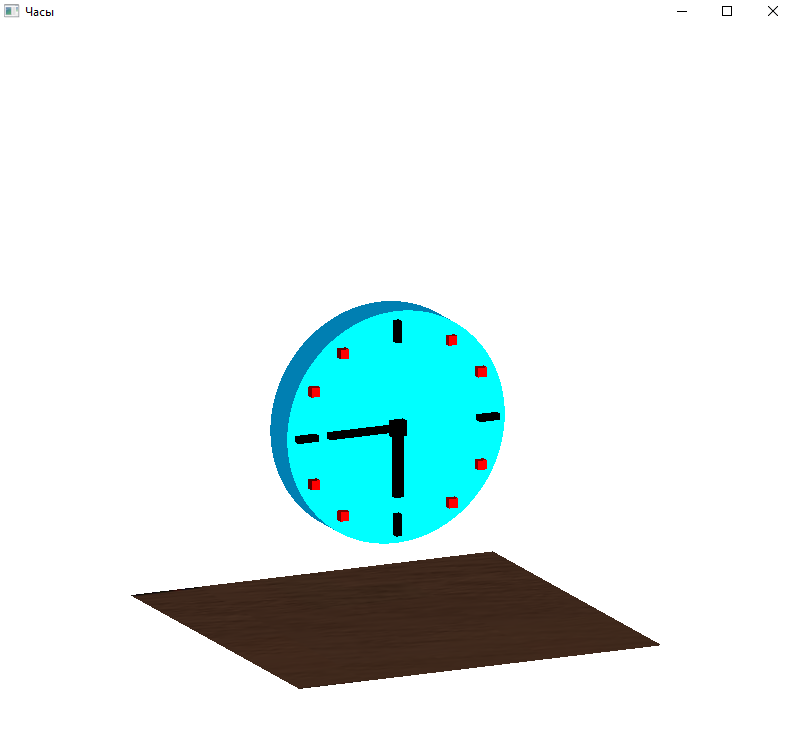


Рис. 2. Результат выполнения кода