Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**«Графическое приложение»**

**Отчет по лабораторной работе №4**

**По дисциплине**

**«Компьютерная графика»**

Студент гр. 442-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Кадочников

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Проверил: доцент каф. АСУ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Шелестов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Томск 2024

Оглавление

[1.Цель работы 3](#_Toc160437285)

[2 Вариант работы и задание 4](#_Toc160437286)

[4 Листинг программы 6](#_Toc160437287)

[5 Выводы по выполненной работе. 17](#_Toc160437288)

1.Цель работы

Освоить принципы работы с использованием языка программирования Visual C++ и библиотеки OpenGL при создании простых 2D и 3D объектов.

2 Вариант работы и задание

Вариант 1

Рисунок 2.1 Основное задание

3 Описание выполнения основного задания

Создав новый проект в Visual studio и подключив библиотеки нужные для работы с OpenGL, начал писать код. Создав класс окна и создав само окно нужно было объяснить программе, что в этом окне должен работать OpenGL поэтому я написал функцию «EnableOpenGL», которая активирует OpenGL в созданном окне. После чего написал простой обработчик сообщений окна. Написал функцию, которая рисует моё 2D задание. Далее для динамического обновления картинки в окне я вызываю функцию отрисовки задания внутри обработчика сообщений. А в конце программы я написал функцию, которая отключает OpenGL и удаляет созданное окно. Таким образом первая часть задания была выполнена. Вторую часть задания я решил делать в отдельном проекте так как это удобнее чем делать всё в одном. Создав новый проект и снова подключив библиотеки, я опять инициализировал OpenGL и создал окно. После чего написал функцию отображения каркасного тора. Добавил функцию вращения для лучшего рассмотрения фигуры и добавил возможность переключение отображения фигуры между линиями каркаса тора и самой фигуры.

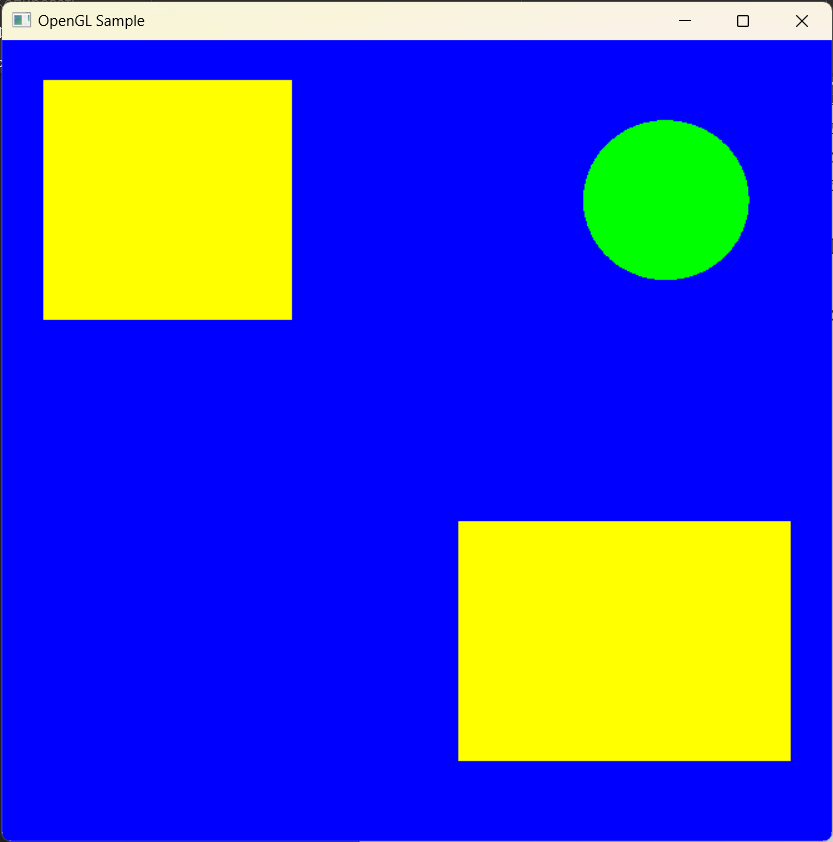
Ниже представлены результаты работы программ(Рисунок 3.1 – 3.3).

Рисунок 3.1 Результат работы первой программы

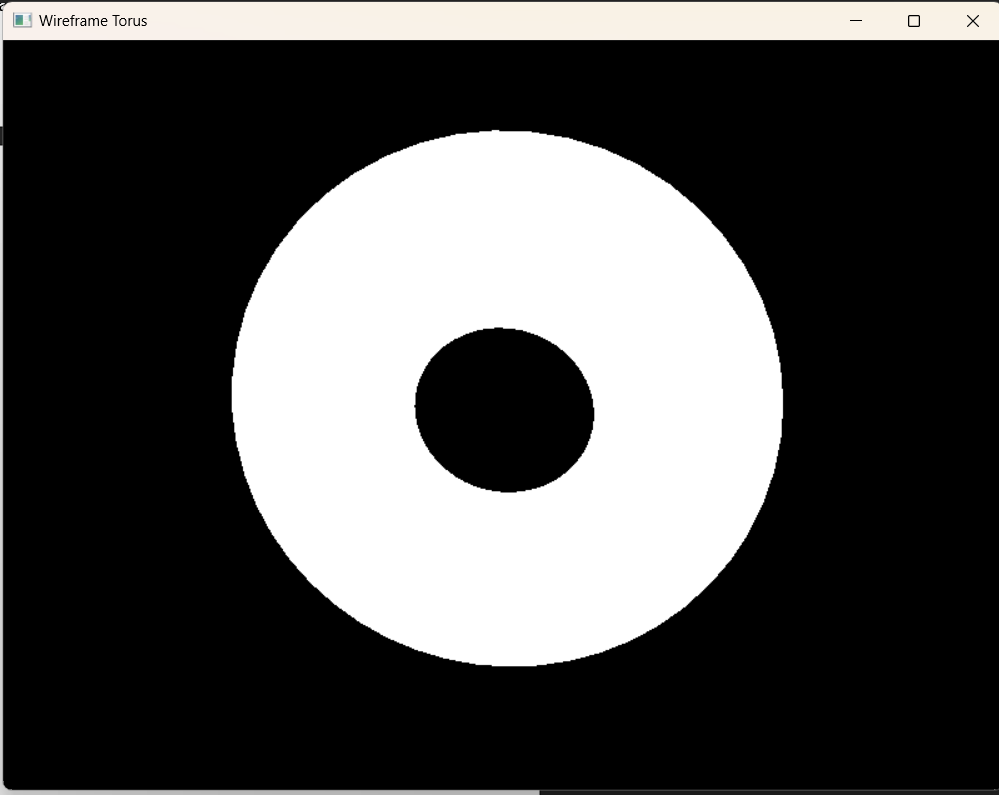
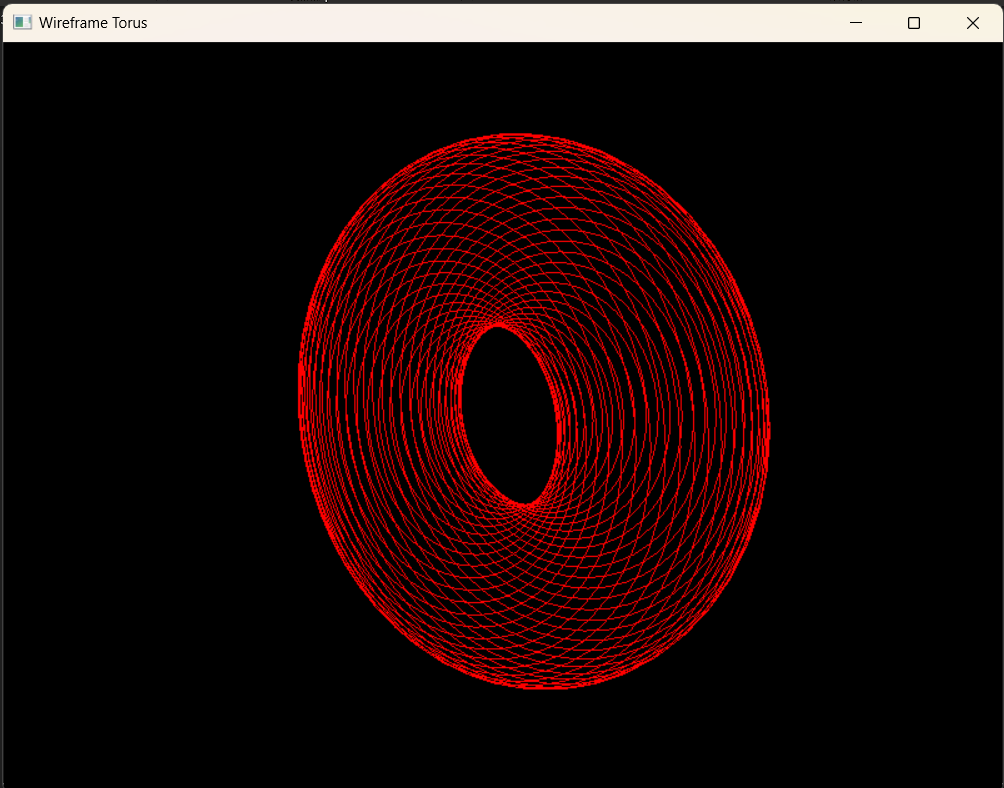


Рисунок 3.3 Результат работы второй программы, но отображаются только линии каркасного тора

Рисунок 3.2 Результат работы второй программы

4 Листинг программы

**Программа 1(2D):**

#include <windows.h>

#include <gl/gl.h>

#include <math.h>

LRESULT CALLBACK WindowProc(HWND, UINT, WPARAM, LPARAM);

void EnableOpenGL(HWND hwnd, HDC\*, HGLRC\*);

void DisableOpenGL(HWND, HDC, HGLRC);

void drow1task(HDC& hDC) {

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

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_QUADS);

glColor3f(0.0f, 0.0f, 1.0f);

glVertex2f(-2.0f, -1.0f);

glVertex2f(1.0f, -1.0f);

glVertex2f(1.0f, 1.0f);

glVertex2f(-2.0f, 1.0f);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 0.0f);

glVertex2f(-0.9f, 0.9f);

glVertex2f(-0.3f, 0.9f);

glVertex2f(-0.3f, 0.3f);

glVertex2f(-0.9f, 0.3f);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 0.0f);

glVertex2f(0.9f, -0.8f);

glVertex2f(0.1f, -0.8f);

glVertex2f(0.1f, -0.2f);

glVertex2f(0.9f, -0.2f);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0.0f, 1.0f, 0.0f);

for(float i = 0; i < 360; i += 1) {

glVertex2f((float)0.6 + (float)0.2 \* (float)cos(i), (float)0.6 + (float)0.2 \* (float)sin(i));

}

glEnd();

SwapBuffers(hDC);

}

int WINAPI WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ int nCmdShow) {

WNDCLASSEX wcex{};

HWND hwnd;

HDC hDC;

HGLRC hRC;

float theta = 0.0f;

/\* register window class \*/

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_OWNDC;

wcex.lpfnWndProc = WindowProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)GetStockObject(BLACK\_BRUSH);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"GLSample";

wcex.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);;

if(!RegisterClassEx(&wcex))

return 0;

hwnd = CreateWindowEx(0, L"GLSample", L"OpenGL Sample", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 680, 680, NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

EnableOpenGL(hwnd, &hDC, &hRC);

MSG msg;

BOOL bQuit = FALSE;

while(!bQuit) {

if(PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE)) {

if(msg.message == WM\_QUIT) {

bQuit = TRUE;

}

else {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

else {

drow1task(hDC);

Sleep(1);

}

}

DisableOpenGL(hwnd, hDC, hRC);

DestroyWindow(hwnd);

return 0;

}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

switch(uMsg) {

case WM\_CLOSE:

PostQuitMessage(0);

break;

case WM\_DESTROY:

return 0;

case WM\_KEYDOWN:

{

switch(wParam) {

case VK\_ESCAPE:

PostQuitMessage(0);

break;

}

}

break;

default:

return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

return 0;

}

void EnableOpenGL(HWND hwnd, HDC\* hDC, HGLRC\* hRC) {

PIXELFORMATDESCRIPTOR pfd;

int iFormat;

\*hDC = GetDC(hwnd);

ZeroMemory(&pfd, sizeof(pfd));

pfd.nSize = sizeof(pfd);

pfd.nVersion = 1;

pfd.dwFlags = PFD\_DRAW\_TO\_WINDOW |

PFD\_SUPPORT\_OPENGL | PFD\_DOUBLEBUFFER;

pfd.iPixelType = PFD\_TYPE\_RGBA;

pfd.cColorBits = 24;

pfd.cDepthBits = 16;

pfd.iLayerType = PFD\_MAIN\_PLANE;

iFormat = ChoosePixelFormat(\*hDC, &pfd);

SetPixelFormat(\*hDC, iFormat, &pfd);

\*hRC = wglCreateContext(\*hDC);

wglMakeCurrent(\*hDC, \*hRC);

}

void DisableOpenGL(HWND hwnd, HDC hDC, HGLRC hRC) {

wglMakeCurrent(NULL, NULL);

wglDeleteContext(hRC);

ReleaseDC(hwnd, hDC);

}

**Программа 2(3D):**

#include <GL/glut.h>

#include <math.h>

const int slices = 50;

const int stacks = 50;

const float innerRadius = 0.5;

const float outerRadius = 1.0;

bool showWireframe = true;

bool showInside = false;

float angle = 0.0f;

void drawWireframeTorus() {

for(int i = 0; i < slices; ++i) {

float theta1 = 2.0f \* 3.1415926f \* float(i) / float(slices);

float theta2 = 2.0f \* 3.1415926f \* float(i + 1) / float(slices);

glBegin(GL\_QUAD\_STRIP);

for(int j = 0; j <= stacks; ++j) {

float phi = 2.0f \* 3.1415926f \* float(j) / float(stacks);

float x = (outerRadius + innerRadius \* cos(phi)) \* cos(theta2);

float y = (outerRadius + innerRadius \* cos(phi)) \* sin(theta2);

float z = innerRadius \* sin(phi);

glVertex3f(x, y, z);

x = (outerRadius + innerRadius \* cos(phi)) \* cos(theta1);

y = (outerRadius + innerRadius \* cos(phi)) \* sin(theta1);

z = innerRadius \* sin(phi);

glVertex3f(x, y, z);

}

glEnd();

}

}

void collorPick(float x1, float x2, float y1, float y2, float z1, float z2) {

glBegin(GL\_LINES);

glColor3f(1.0f, 0.0f, 0.0f); // Красный

glVertex3f(x1, y1, z1);

glVertex3f(x2, y2, z2);

if(showInside) {

glColor3f(0.0f, 1.0f, 0.0f); // Зеленый

glVertex3f(x1, y1, z1);

glVertex3f(x1, y1, -z1);

//glColor3f(0.0f, 0.0f, 1.0f); // Синий

//glVertex3f(x2, y2, z2);

//glVertex3f(x2, y2, -z2);

}

glEnd();

}

void drawSolidTorus() {

for(int i = 0; i < slices; ++i) {

float theta1 = 2.0f \* 3.1415926f \* float(i) / float(slices);

float theta2 = 2.0f \* 3.1415926f \* float(i + 1) / float(slices);

for(int j = 0; j <= stacks; ++j) {

float phi = 2.0f \* 3.1415926f \* float(j) / float(stacks);

float x1 = (outerRadius + innerRadius \* cos(phi)) \* cos(theta1);

float y1 = (outerRadius + innerRadius \* cos(phi)) \* sin(theta1);

float z1 = innerRadius \* sin(phi);

float x2 = (outerRadius + innerRadius \* cos(phi)) \* cos(theta2);

float y2 = (outerRadius + innerRadius \* cos(phi)) \* sin(theta2);

float z2 = innerRadius \* sin(phi);

collorPick(x1, x2, y1, y2, z1, z2);

}

}

}

void drawTorus() {

if(showWireframe)

drawWireframeTorus();

else

drawSolidTorus();

}

void display() {

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

glLoadIdentity();

glTranslatef(0.0f, 0.0f, -5.0f);

glRotatef(30.0f, 1.0f, 0.0f, 0.0f);

glRotatef(30.0f, 0.0f, 1.0f, 0.0f);

glColor3f(1.0f, 1.0f, 1.0f);

glRotatef(angle, 1.0f, 1.0f, 0.0f);

glRotatef(angle, 0.0f, 1.0f, 0.0f);

drawTorus();

glutSwapBuffers();

}

void reshape(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0f, (float)w / (float)h, 0.1f, 100.0f);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void update(int value) {

angle += 0.5f;

if(angle > 360) {

angle -= 360;

}

glutPostRedisplay();

glutTimerFunc(16, update, 0);

}

void keyboard(unsigned char key, int x, int y) {

if(key == ' ') {

showWireframe = !showWireframe;

glutPostRedisplay();

}

if(key == 'v') {

showInside = !showInside;

glutPostRedisplay();

}

}

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

glutInit(&argc, argv);

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

glutInitWindowSize(800, 600);

glutCreateWindow("Wireframe Torus");

glEnable(GL\_DEPTH\_TEST);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutTimerFunc(16, update, 0);

glutMainLoop();

return 0;

}

5 Выводы по выполненной работе.

В ходе выполнения лабораторной работы я освоил основные принципы работы с языком программирования Visual C++ и библиотеки OpenGL при создании простых 2D и 3D объектов.