| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| --- | --- |
| БГТУ.СМК-Ф-4.2-К5-01 |

| Факультет |  | О |  | Естественнонаучный |
| --- | --- | --- | --- | --- |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Компьютерная геометрия и графика | | |

Отчёт по практической работе № 1

| Подключение библиотек; контекст устройства, |
| --- |
| контекст воспроизведения, общий вид программы |
|  |

| Выполнил студент группы | | | |  | И508Б |
| --- | --- | --- | --- | --- | --- |
| Кабиров К.Р. | | | | | |
| Фамилия И.О. | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | |
|  | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2022г. |

САНКТ-ПЕТЕРБУРГ

2022 г

Цель работы – создание программы-заготовки для работы с библиотекой ОpenGL.

В результате выполнения шагов из методических материалов на экран будет выведено три точки на белом экране. Результат работы программы представлен на рисунке 1.

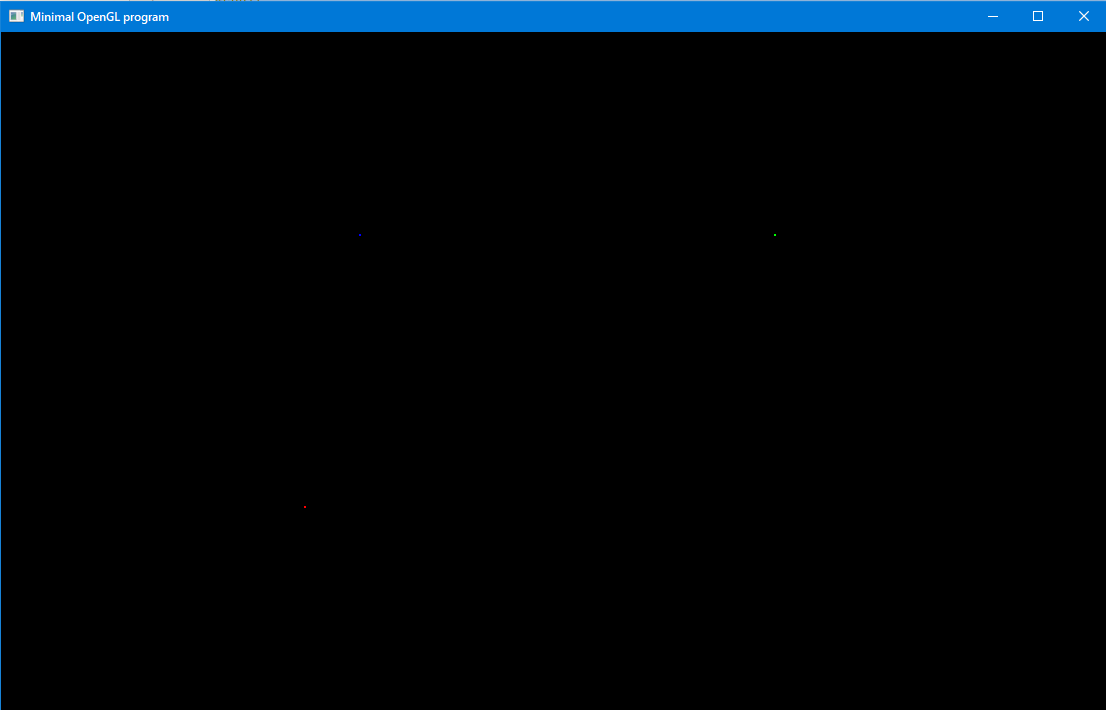


Рисунок 1 – Результат работы программы

Приложение А

Код программы:

#ifdef \_\_APPLE\_CC\_\_

#include <GLUT/glut.h>

#else

#include "GL/glut.h"

#endif

static HGLRC hRC; // Постоянный контекст рендеринга

static HDC hDC; // Приватный контекст устройства GDI

BOOL keys[256]; // Массив для процедуры обработки клавиатуры

GLvoid InitGL(GLsizei Width, GLsizei Height) //Вызвать после создания окна GL

{

glClearColor(0.0f, 0.0f, 0.0f, 0.0f); // Очистка экрана в черный цвет

}

GLvoid ReSizeGLScene(GLsizei Width, GLsizei Height) {

if (Height == 0)// Предотвращение деления на ноль, если окно слишком мало

Height = 1;

glViewport(0, 0, Width, Height); // Сброс текущей области вывода и перспективных преобразований

}

GLvoid DrawGLScene(GLvoid) {

glClear(GL\_COLOR\_BUFFER\_BIT); // очистка экрана

glPointSize(2); //размер точки

glBegin(GL\_POINTS);

glColor3d(1, 0, 0);

glVertex3d(-0.45, -0.4, 0); // первая точка

glColor3d(0, 1, 0);

glVertex3d(0.4, 0.4, 0); // вторая точка

glColor3d(0, 0, 1);

glVertex3d(-0.35, 0.4, 0); // третья точка

glEnd();

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

RECT Screen; // используется позднее для размеров окна

GLuint PixelFormat;

static PIXELFORMATDESCRIPTOR pfd = {

sizeof(PIXELFORMATDESCRIPTOR), // Размер этой структуры

1, // Номер версии

PFD\_DRAW\_TO\_WINDOW | // Формат для Окна

PFD\_SUPPORT\_OPENGL | // Формат для OpenGL

PFD\_DOUBLEBUFFER, // Формат для двойного буфера

PFD\_TYPE\_RGBA, // Требуется RGBA формат

16, // Выбор 16 бит глубины цвета

0, 0, 0, 0, 0, 0, // Игнорирование цветовых битов

0, // нет буфера прозрачности

0, // Сдвиговый бит игнорируется

0, // Нет буфера аккумуляции

0, 0, 0, 0, // Биты аккумуляции игнорируются

16, // 16 битный Z-буфер (буфер глубины)

0, // Нет буфера трафарета

0, // Нет вспомогательных буферов

PFD\_MAIN\_PLANE, // Главный слой рисования

0, // Резерв

0, 0, 0 // Маски слоя игнорируются

};

switch (message) // Тип сообщения

{

case WM\_CREATE:

hDC = GetDC(hWnd); // Получить контекст устройства для окна

PixelFormat = ChoosePixelFormat(hDC, &pfd); // Найти ближайшее совпадение для формата пикселей

if (!PixelFormat) {

MessageBox(nullptr, "Не найден подходящий формат пикселя.",

"Ошибка", MB\_OK | MB\_ICONERROR);

PostQuitMessage(0); // Это сообщение говорит, что программа должна завершиться

break; // Предотвращение повтора кода

}

if (!SetPixelFormat(hDC, PixelFormat, &pfd)) {

MessageBox(0, "Формат пикселя не установлен.",

"Ошибка", MB\_OK | MB\_ICONERROR);

PostQuitMessage(0);

break;

}

hRC = wglCreateContext(hDC);

if (!hRC) {

MessageBox(0, "Контекст воспроизведения не создан.",

"Ошибка", MB\_OK | MB\_ICONERROR);

PostQuitMessage(0);

break;

}

if (!wglMakeCurrent(hDC, hRC)) {

MessageBox(0, "Невозможно активизировать GLRC.",

"Ошибка", MB\_OK | MB\_ICONERROR);

PostQuitMessage(0);

break;

}

GetClientRect(hWnd, &Screen);

InitGL(Screen.right, Screen.bottom);

break;

case WM\_DESTROY:

case WM\_CLOSE:

ChangeDisplaySettings(NULL, 0);

wglMakeCurrent(hDC, NULL);

wglDeleteContext(hRC);

ReleaseDC(hWnd, hDC);

PostQuitMessage(0);

break;

case WM\_KEYDOWN:

keys[wParam] = TRUE;

break;

case WM\_KEYUP:

keys[wParam] = FALSE;

break;

case WM\_SIZE:

ReSizeGLScene(LOWORD(lParam), HIWORD(lParam));

break;

default:

return (DefWindowProc(hWnd, message,

wParam, lParam));

}

return (0);

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

MSG msg; // Структура сообщения Windows

WNDCLASS wc; // Структура класса Windows для установки типа окна

HWND hWnd; // Сохранение дескриптора окна

wc.style = CS\_HREDRAW | CS\_VREDRAW | CS\_OWNDC;

wc.lpfnWndProc = (WNDPROC) WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = NULL;

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

wc.hbrBackground = NULL;

wc.lpszMenuName = NULL;

wc.lpszClassName = "OpenGL WinClass";

if (!RegisterClass(&wc)) {

MessageBox(0, "Ошибка регистрации класса окна.", "Ошибка", MB\_OK | MB\_ICONERROR);

return FALSE;

}

// Заголовок вверху окна

hWnd = CreateWindow("OpenGL WinClass", "Minimal OpenGL program",

WS\_POPUP | WS\_CLIPCHILDREN | WS\_CLIPSIBLINGS | WS\_OVERLAPPEDWINDOW, 0, 0, 1920, 1080, NULL, NULL,

hInstance, NULL);

if (!hWnd) {

MessageBox(0, "Ошибка создания окна.",

"Ошибка", MB\_OK | MB\_ICONERROR);

return FALSE;

}

DEVMODE dmScreenSettings; // Режим работы

memset(&dmScreenSettings, 0, sizeof(DEVMODE)); // Очистка для хранения установок

dmScreenSettings.dmSize = sizeof(DEVMODE); // Размер структуры Devmode

dmScreenSettings.dmPelsWidth = 1920; // Ширина экрана

dmScreenSettings.dmPelsHeight = 1080; // Высота экрана

dmScreenSettings.dmFields = DM\_PELSWIDTH | DM\_PELSHEIGHT; // Режим Пикселя

//ChangeDisplaySettings(&dmScreenSettings, CDS\_FULLSCREEN); // Переключение в полный экран

ShowWindow(hWnd, SW\_SHOW);

UpdateWindow(hWnd);

SetFocus(hWnd);

while (true) {

// Обработка всех сообщений

while (PeekMessage(&msg, NULL, 0, 0, PM\_NOREMOVE)) {

if (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

} else {

return TRUE;

}

}

DrawGLScene(); // Нарисовать сцену

SwapBuffers(hDC); // Переключить буфер экрана

if (keys[VK\_ESCAPE]) SendMessage(hWnd, WM\_CLOSE, 0, 0); // Если ESC - выйти

}

}

Содержимое файла «CMakeLists.txt»:

cmake\_minimum\_required(VERSION 3.23)

project(lab11)

set(OPENGL\_INCLUDE\_DIR D:/JetBrains/CLion 2022.2.4/bin/mingw/include)

set(OPENGL\_LIBRARIES D:/JetBrains/CLion 2022.2.4/bin/mingw/lib)

find\_package(OpenGL REQUIRED)

find\_package(GLUT REQUIRED)

add\_executable(lab11 main.cpp)

target\_link\_libraries(lab11 ${OPENGL\_LIBRARIES})

target\_link\_libraries(lab11 ${GLUT\_LIBRARIES})

target\_link\_libraries(lab11 -lopengl32 -g -lfreeglut)