Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра Информатики

Отчёт по лабораторной работе №1

По дисциплине “Инженерная и компьютерная графика”

Выполнил:

Студент группы ПРО-231 К.И. Арменшин

Проверил:

доцент кафедры ВМиК Котельников В.А.

Уфа – 2023

**Цель:**

Знакомство с OpenGL.

**Ход работы:**

**Урок 1.**

Создано окно для отрисовки объектов (Рисунок 1).

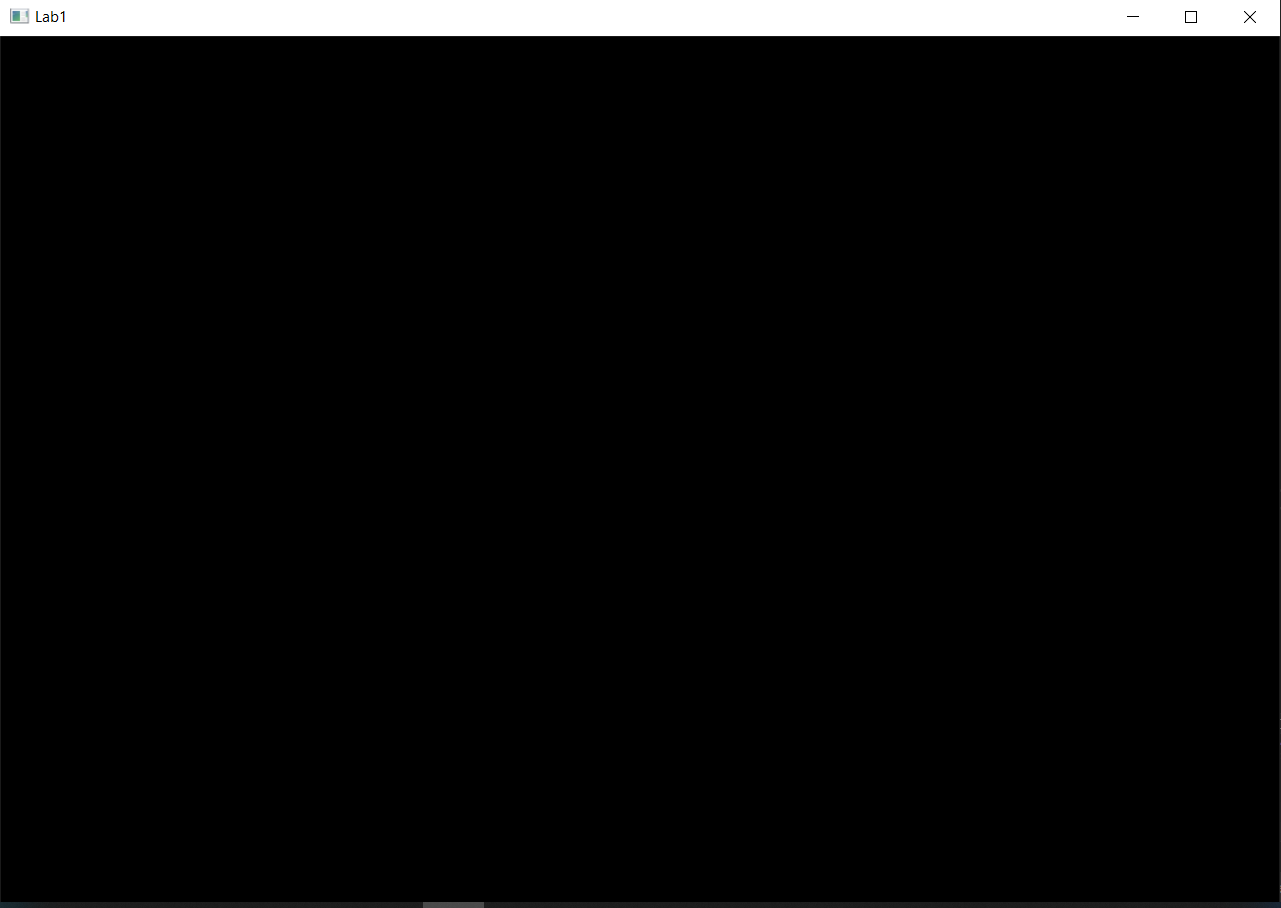


Рисунок . Создание окна

**Урок 2.**

Нарисована точка в окне, созданном в уроке 1 (Рисунок 2).

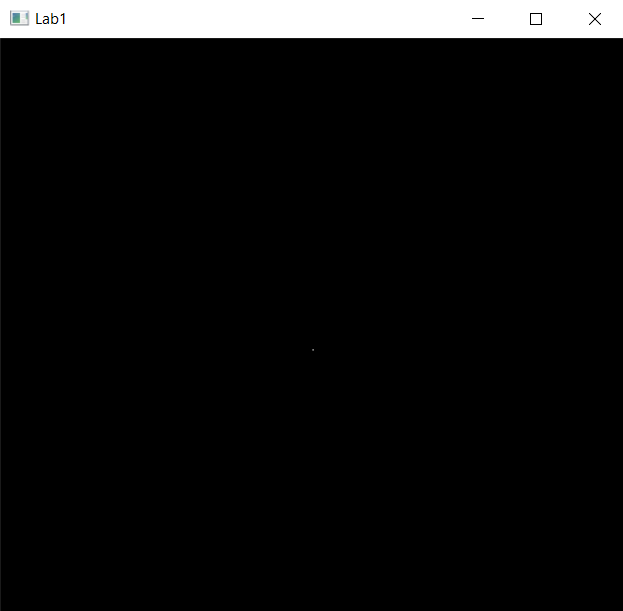


Рисунок . Отрисовка точки

**Урок 3.**

Нарисован треугольник по трём точкам (Рисунок 3).

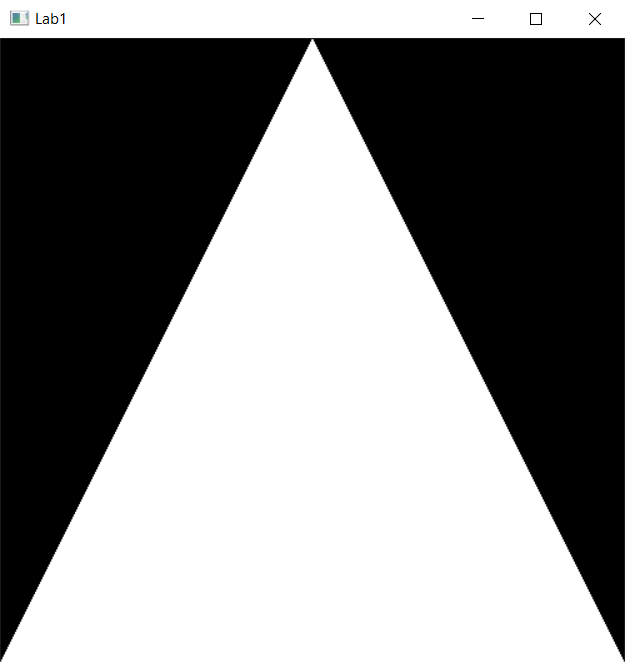


Рисунок . Отрисовка треугольника

**Вывод**

В результате выполнения данной лабораторной работы были выполнены уроки 1-3, в ходе которых была разработана программа, создающая окно и рисующая в нём треугольник, а также были установлены библиотеки GLUT и GLEW.

Полный код находится в Github:

https://github.com/karimarm/Labs\_ECG.git

**Приложение 1**

L1\_ECG.cpp

#include <stdio.h>

#include <GL/glew.h>

#include <GL/freeglut.h>

#include "math\_3d.h"

// Глобальная переменная для хранения указателя на буфер вершин

GLuint VBO;

static void RenderSceneCB()

{

// Очистка буфера

glClear(GL\_COLOR\_BUFFER\_BIT);

glEnableVertexAttribArray(0); // Массив атрибутов вершин

glBindBuffer(GL\_ARRAY\_BUFFER, VBO); // Привязка буфера

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, 0); // Восприятие данных внутри буфера

glDrawArrays(GL\_TRIANGLES, 0, 3); // Отрисовка

glDisableVertexAttribArray(0); // Отключить атрибуты вершин

// Меняет буферы местами

glutSwapBuffers();

}

static void InitializeGlutCallbacks()

{

// Функция обратного вызова отрисовки одного кадра

glutDisplayFunc(RenderSceneCB);

}

static void CreateVertexBuffer()

{

// Массив вершин треугольника

Vector3f Vertices[3];

Vertices[0] = Vector3f(-1.0f, -1.0f, 0.0f);

Vertices[1] = Vector3f(1.0f, -1.0f, 0.0f);

Vertices[2] = Vector3f(0.0f, 1.0f, 0.0f);

glGenBuffers(1, &VBO);// Функция для генерации объектов

glBindBuffer(GL\_ARRAY\_BUFFER, VBO); // Привязка буфера

glBufferData(GL\_ARRAY\_BUFFER, sizeof(Vertices), Vertices, GL\_STATIC\_DRAW); // Наполнение буфера данными

}

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

{

// Инициализация GLUT

glutInit(&argc, argv);

// Настройка опций GLUT

// GLUT\_DOUBLE - двойная буферизация

// GLUT\_RGBA - буфер цвета

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

// Задание параметров окна и его создание с заголовком "Lab 1"

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Lab 1");

// Отрисовка кадра

InitializeGlutCallbacks();

// Инициализация GLEW и проверка на ошибки

GLenum res = glewInit();

if (res != GLEW\_OK) {

fprintf(stderr, "Error: '%s'\n", glewGetErrorString(res));

return 1;

}

// Цвет очистки буфера кадра

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

CreateVertexBuffer();

// Передача контроля GLUT'у

glutMainLoop();

return 0;

}