Министерство науки и высшего образования Российской Федерации
Мытищинский филиал
Федерального государственного бюджетного образовательного
учреждение высшего образования
«Московский государственный технический университет имени Н.Э.
Баумана
(национальный исследовательский университет)»
(МФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **КОСМИЧЕСКИЙ**
КАФЕДРА **К-3**

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №3.1
по ДИСЦИПЛИНЕ
«КОМПЬЮТЕРНАЯ ГРАФИКА»

Студент **КЗ-43Б** **Н. А. Чуйко**
(Группа) (И.О.Фамилия)

Преподаватель **В.В. Афанасьева**
(И.О.Фамилия)

2023 г.

ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ № 3.1

1. Реализовать базовую часть «построение 2D графика функции».
    - Разобраться в коде.

Листинг кода

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tao.FreeGlut;
using Tao.OpenGl;

namespace LW_3_Graph_
{
    public partial class Form1 : Form
    {

        // размеры окна
        double ScreenW, ScreenH;

        // отношения сторон окна визуализации
        // для корректного перевода координат мыши в координаты,
        // принятые в программе

        private float devX;
        private float devY;

        // массив, который будет хранить значения x,y точек графика
        private float[,] GrapValuesArray;
        // количество элементов в массиве
        private int elements_count = 0;

        // флаг, означающий, что массив с значениями координат графика пока еще не заполнен
        private bool not_calculate = true;

        // номер ячейки массива, из которой будут взяты координаты для красной точки
        // для визуализации текущего кадра
        private int pointPosition = 0;

        // вспомогательные переменные для построения линий от курсора мыши к координатным осям
        float lineX, lineY;

        // текущение координаты курсора мыши
        float Mcoord_X = 0, Mcoord_Y = 0;

        private void AnT_MouseMove(object sender, MouseEventArgs e)
        {
            // сохраняем координаты мыши
            Mcoord_X = e.X;
            Mcoord_Y = e.Y;

            // вычисляем параметры для будущей дорисовки линий от указателя мыши к координатным осям.
            lineX = devX * e.X;
            lineY = (float)(ScreenH - devY * e.Y);


        }

        public Form1()
        {
            InitializeComponent();
            AnT.InitializeContexts();
        }

        private void PointInGrap_Tick(object sender, EventArgs e)
        {
            // если мы дошли до последнего элемента массива
            if (pointPosition == elements_count - 1)
                pointPosition = 0; // переходим к начальному элементу

            // функция визуализации
            Draw();

            // переход к следующему элементу массива
            pointPosition++;

        }

        // функция визуализации текста
        private void PrintText2D(float x, float y, string text)
        {

            // устанавливаем позицию вывода растровых символов
            // в переданных координатах x и y.
            Gl.glRasterPos2f(x, y);

            // в цикле foreach перебираем значения из массива text,
            // который содержит значение строки для визуализации
            foreach (char char_for_draw in text)
            {
                // символ C визуализируем с помощью функции glutBitmapCharacter, используя шрифт GLUT_BITMAP_9_BY_15.
                Glut.glutBitmapCharacter(Glut.GLUT_BITMAP_9_BY_15, char_for_draw);
            }

        }
```

```
// функция, производящая вычисления координат графика
// и заносящая их в массив GrapValuesArray
private void functionCalculation()
{

    // определение локальных переменных X и Y
    float x = 0, y = 0;

    // инициализация массива, который будет хранить значение 300 точек,
    // из которых будет состоять график

    GrapValuesArray = new float[300, 2];

    // счетчик элементов массива
    elements_count = 0;

    // вычисления всех значений y для x, принадлежащего промежутку от -15 до 15 с шагом в 0.01f
    for (x = -15; x < 15; x += 0.1f)
    {
        // вычисление y для текущего x
        // по формуле y = (float)Math.Sin(x)*3 + 1;
        // эта строка задает формулу, описывающую график функции для нашего уравнения y = f(x).
        y = (float)Math.Sin(x) * 3 + 1;

        // запись координаты x
        GrapValuesArray[elements_count, 0] = x;
        // запись координаты y
        GrapValuesArray[elements_count, 1] = y;
        // подсчет элементов
        elements_count++;

    }

    // изменяем флаг, сигнализировавший о том, что координаты графика не вычислены
    not_calculate = false;

}


// визуализация графика
private void DrawDiagram()
{

    // проверка флага, сигнализирующего о том, что координаты графика вычислены
    if (not_calculate)
    {
        // если нет, то вызываем функцию вычисления координат графика
        functionCalculation();
    }

    // стартуем отрисовку в режиме визуализации точек
    // объединяемых в линии (GL_LINE_STRIP)
    Gl.glBegin(Gl.GL_LINE_STRIP);

    // рисуем начальную точку
    Gl.glVertex2d(GrapValuesArray[0, 0], GrapValuesArray[0, 1]);

    // проходим по массиву с координатами вычисленных точек
    for (int ax = 1; ax < elements_count; ax += 2)
    {
        // передаем в OpenGL информацию о вершине, участвующей в построении линий
        Gl.glVertex2d(GrapValuesArray[ax, 0], GrapValuesArray[ax, 1]);
    }

    // завершаем режим рисования
    Gl.glEnd();
    // устанавливаем размер точек, равный 5 пикселям
    Gl.glPointSize(5);
    // устанавливаем текущим цветом - красный цвет
    Gl.glColor3f(255, 0, 0);
    // активируем режим вывода точек (GL_POINTS)
    Gl.glBegin(Gl.GL_POINTS);
    // выводим красную точку, используя ту ячейку массива, до которой мы дошли (вычисляется в функции обработчике событий таймера)
    Gl.glVertex2d(GrapValuesArray[pointPosition, 0], GrapValuesArray[pointPosition, 1]);
    // завершаем режим рисования
    Gl.glEnd();
    // устанавливаем размер точек равный единице
    Gl.glPointSize(1);

}


// функция, управляющая визуализацией сцены
private void Draw()
{

    // очистка буфера цвета и буфера глубины
    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);

    // очищение текущей матрицы
    Gl.glLoadIdentity();

    // установка черного цвета
    Gl.glColor3f(0, 0, 0);

    // помещаем состояние матрицы в стек матриц
    Gl.glPushMatrix();

    // выполняем перемещение в пространстве по осям X и Y
    Gl.glTranslated(15, 15, 0);

    // активируем режим рисования (Указанные далее точки будут выводиться как точки GL_POINTS)
    Gl.glBegin(Gl.GL_POINTS);

    // с помощью прохода вдумя циклами, создаем сетку из точек
```

```
    for (int ax = -15; ax < 15; ax++)
    {
        for (int bx = -15; bx < 15; bx++)
        {
            // вывод точки
            Gl.glVertex2d(ax, bx);
        }
    }

    // завершение режима рисования примитивов
    Gl.glEnd();

    // активируем режим рисования, каждые 2 последовательно вызванные команды glVertex
    // объединяются в линии
    Gl.glBegin(Gl.GL_LINES);

    // далее мы рисуем координатные оси и стрелки на их концах
    Gl.glVertex2d(0, -15);
    Gl.glVertex2d(0, 15);

    Gl.glVertex2d(-15, 0);
    Gl.glVertex2d(15, 0);

    // вертикальная стрелка
    Gl.glVertex2d(0, 15);
    Gl.glVertex2d(0.1, 14.5);
    Gl.glVertex2d(0, 15);
    Gl.glVertex2d(-0.1, 14.5);

    // горизонтальная трелка
    Gl.glVertex2d(15, 0);
    Gl.glVertex2d(14.5, 0.1);
    Gl.glVertex2d(15, 0);
    Gl.glVertex2d(14.5, -0.1);

    // завершаем режим рисования
    Gl.glEnd();

    // выводим подписи осей "x" и "y"
    PrintText2D(15.5f, 0, "x");
    PrintText2D(0.5f, 14.5f, "y");

    // вызываем функцию рисования графика
    DrawDiagram();

    // возвращаем матрицу из стека
    Gl.glPopMatrix();

    // выводим текст со значением координат возле курсора
    PrintText2D(devX * Mcoord_X + 0.2f, (float)ScreenH - devY * Mcoord_Y + 0.4f, "[ x: " + (devX * Mcoord_X - 15).ToString() + " ; y: " + ((float)ScreenH - devY * Mcoord_Y - 15).ToString() + "]");

    // устанавливаем красный цвет
    Gl.glColor3f(255, 0, 0);

    // включаем режим рисования линий, для того чтобы нарисовать
    // линии от курсора мыши к координатным осям
    Gl.glBegin(Gl.GL_LINES);

    Gl.glVertex2d(lineX, 15);
    Gl.glVertex2d(lineX, lineY);
    Gl.glVertex2d(15, lineY);
    Gl.glVertex2d(lineX, lineY);

    Gl.glEnd();

    // дожидаемся завершения визуализации кадра
    Gl.glFlush();

    // сигнал для обновление элемента реализующего визуализацию.
    AnT.Invalidate();

}



private void Form1_Load(object sender, EventArgs e)
{


    // инициализация библиотеки glut
    Glut.glutInit();
    // инициализация режима экрана
    Glut.glutInitDisplayMode(Glut.GLUT_RGB | Glut.GLUT_DOUBLE);

    // установка цвета очистки экрана (RGBA)
    Gl.glClearColor(255, 255, 255, 1);

    // установка порта вывода
    Gl.glViewport(0, 0, AnT.Width, AnT.Height);

    // активация проекционной матрицы
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    // очистка матрицы
    Gl.glLoadIdentity();

    // определение параметров настройки проекции в зависимости от размеров сторон элемента AnT.
    if ((float)AnT.Width <= (float)AnT.Height)
    {
        ScreenW = 30.0;
        ScreenH = 30.0 * (float)AnT.Height / (float)AnT.Width;
        Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);
    }
    else
    {
        ScreenW = 30.0 * (float)AnT.Width / (float)AnT.Height;
```

```
                ScreenH = 30.0;
                Glu.gluOrtho2D(0.0, 30.0 * (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);
            }

            // сохранение коэффициентов, которые нам необходимы для перевода координат указателя в оконной системе в координаты,
            // принятые в нашей OpenGL сцене
            devX = (float)ScreenW / (float)AnT.Width;
            devY = (float)ScreenH / (float)AnT.Height;

            // установка объектно-видовой матрицы
            Gl.glMatrixMode(Gl.GL_MODELVIEW);

            // старт счетчика, отвечающего за вызов функции визуализации сцены
            PointInGrap.Start();

        }
    }
}
```

Скриншот