|  |  |
| --- | --- |
|  | **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***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

# *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* по ДИСЦИПЛИНЕ

**«компьютерная графика»**

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_\_\_К3-51Б\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Бортников М.Д.

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**В.В. Афанасьева**\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Задание к лабораторной работе № 3**

Реализовать базовую часть «построение 2D графика функции».

Разобраться в коде.

Для базового варианта разобраться с тем, какие значения принимают величины Xmin, Xmax, Ymin, Ymax, и где в коде программы они задействованы.

Реализовать ввод переменных Xmin, Xmax, Ymin, Ymax в окне формы и отображать график функции согласно выбранным параметрам.

Реализовать отображение графика функции согласно индивидуальному заданию (предлагается добавить на форму «combobox» с двумя полями: стандартная функция и индивидуальное задание – для примера можно взять функцию у=|х|-2).

Реализовать выбор цвета отрисовки: для линии графика функции, для осей координат и их подписей, для отображения позиций курсора мыши (предлагается добавить на форму 2 «combobox»: 1- для какого объекта меняем цвет, 2- выбор цветов - например: красный, синий, зеленый, черный).

**Код программы**

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;

// для работы с библиотекой OpenGL

using Tao.OpenGl;

// для работы с библиотекой FreeGLUT

using Tao.FreeGlut;

// для работы с элементом управления SimpleOpenGLControl

using Tao.Platform.Windows;

namespace Labb1

{

public partial class Form1 : Form

{

float x\_min = -15, x\_max = 15, y\_min = -15, y\_max = 15;

float pointR, pointG, pointB;

float axisR, axisG, axisB;

float graphR, graphG, graphB;

// 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 textBox1\_TextChanged(object sender, EventArgs e)

{

y\_max = Convert.ToInt32(textBox1.Text); ScreenUpdate();

}

private void ScreenUpdate()

{

functionCalculation();

// 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 = 1.01 \* (x\_max - x\_min);

ScreenH = 1.01 \* (y\_max - y\_min);

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

}

else

{

ScreenW = 1.01 \* (x\_max - x\_min);

ScreenH = 1.01 \* (y\_max - y\_min);

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

}

// 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);

// изменяем флаг, сигнализировавший о том, что координаты графика не вычислены

not\_calculate = true;

// старт счетчика, отвечающего за вызов функции визуализации сцены

PointInGrap.Start();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

x\_min = Convert.ToInt32(textBox3.Text); ScreenUpdate();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

y\_min = Convert.ToInt32(textBox2.Text); ScreenUpdate();

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

x\_max = Convert.ToInt32(textBox4.Text); ScreenUpdate();

}

public Form1()

{

InitializeComponent();

comboBox1.SelectedIndex = 0;

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++;

}

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 = 1.01 \* (x\_max - x\_min);

ScreenH = 1.01 \* (y\_max - y\_min);

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

}

else

{

ScreenW = 1.01 \* (x\_max - x\_min);

ScreenH = 1.01 \* (y\_max - y\_min);

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

}

// 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();

}

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);

}

// 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);

}

}

private void AnT\_Load(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

functionCalculation();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

switch (comboBox2.SelectedIndex)

{

case 0:

pointR = colorDialog1.Color.R;

pointG = colorDialog1.Color.G;

pointB = colorDialog1.Color.B;

break;

case 1:

axisR = colorDialog1.Color.R;

axisG = colorDialog1.Color.G;

axisB = colorDialog1.Color.B;

break;

case 2:

graphR = colorDialog1.Color.R;

graphG = colorDialog1.Color.G;

graphB = colorDialog1.Color.B;

break;

default:

break;

}

}

// 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[(int)(x\_max - x\_min) \* 10 + 1, 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 = x\_min; x < x\_max; 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).

switch(comboBox1.SelectedIndex)

{

case 0:

y = (float)Math.Sin(x) \* 3 + 1;

break;

case 1:

y = (float)Math.Abs(x) - 2;

break;

case 2:

y = (float)(1 / x) + 3;

break;

case -1:

y = (float)Math.Sin(x) \* 3 + 1;

break;

}

//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(); pointPosition = 0;

}

// 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(pointR, pointG, pointB);

// 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(-x\_min, -y\_min, 0);

float w, h; w = -x\_min; h = -y\_min;

// 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 = (int)x\_min; ax < x\_max; ax++)

{

for (int bx = (int)y\_min; bx < y\_max; 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);

Gl.glColor3f((float)(axisR / 255.0), (float)(axisG / 255.0), (float)(axisB / 255.0));

// далее мы рисуем координатные оси и стрелки на их концах

Gl.glVertex2d(0, y\_min);

Gl.glVertex2d(0, y\_max);

Gl.glVertex2d(x\_min, 0);

Gl.glVertex2d(x\_max, 0);

// вертикальная стрелка

Gl.glVertex2d(0, y\_max);

Gl.glVertex2d(0.1, y\_max - 0.5);

Gl.glVertex2d(0, y\_max);

Gl.glVertex2d(-0.1, y\_max);

// горизонтальная стрелка

Gl.glVertex2d(x\_max, 0);

Gl.glVertex2d(x\_max - 0.5, 0.1);

Gl.glVertex2d(x\_max, 0);

Gl.glVertex2d(x\_max - 0.5, -0.1);

// завершаем режим рисования

Gl.glEnd();

// выводим подписи осей "x" и "y"

PrintText2D((float)(x\_max + 0.5), 0, "x");

PrintText2D(0.5f, (float)(y\_max - 0.5), "y");

Gl.glColor3f((float)(graphR / 255.0), (float)(graphG / 255.0), (float)(graphB / 255.0));

// 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 - w).ToString() + " ; y: " + ((float)ScreenH - devY \* Mcoord\_Y - h).ToString() + "]");

// устанавливаем красный цвет

Gl.glColor3f((float)(pointR / 255.0), (float)(pointG / 255.0), (float)(pointB / 255.0));

// включаем режим рисования линий, для того чтобы нарисовать

// линии от курсора мыши к координатным осям

Gl.glBegin(Gl.GL\_LINES);

Gl.glVertex2d(lineX, h);

Gl.glVertex2d(lineX, lineY);

Gl.glVertex2d(w, lineY);

Gl.glVertex2d(lineX, lineY);

Gl.glEnd();

// дожидаемся завершения визуализации кадра

Gl.glFlush();

// сигнал для обновление элемента реализующего визуализацию.

AnT.Invalidate();

}

}

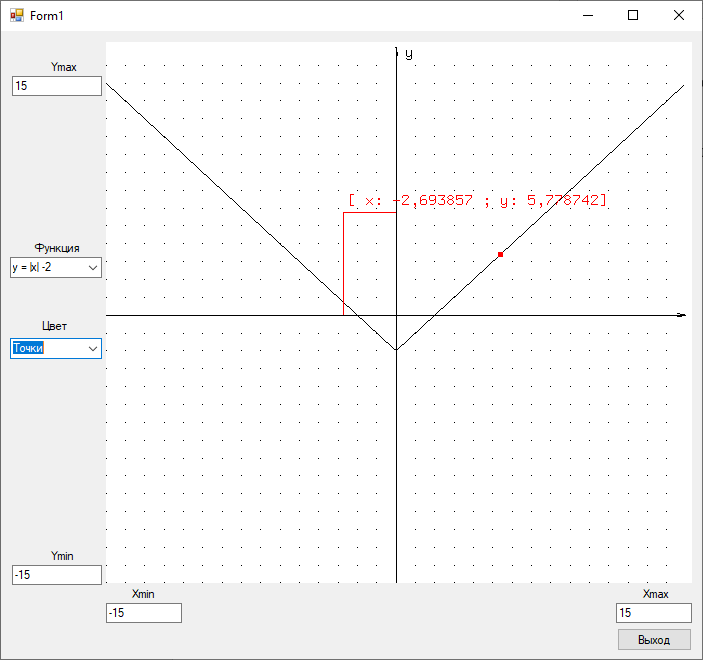
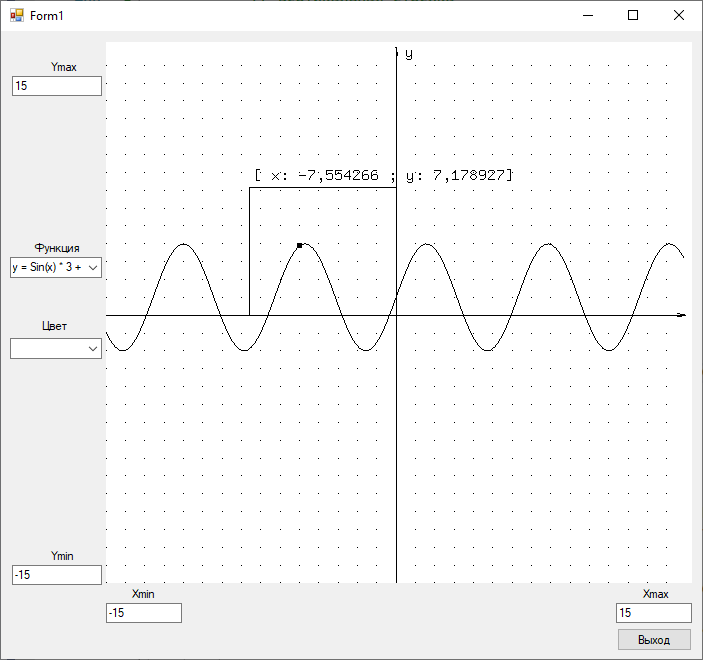
}

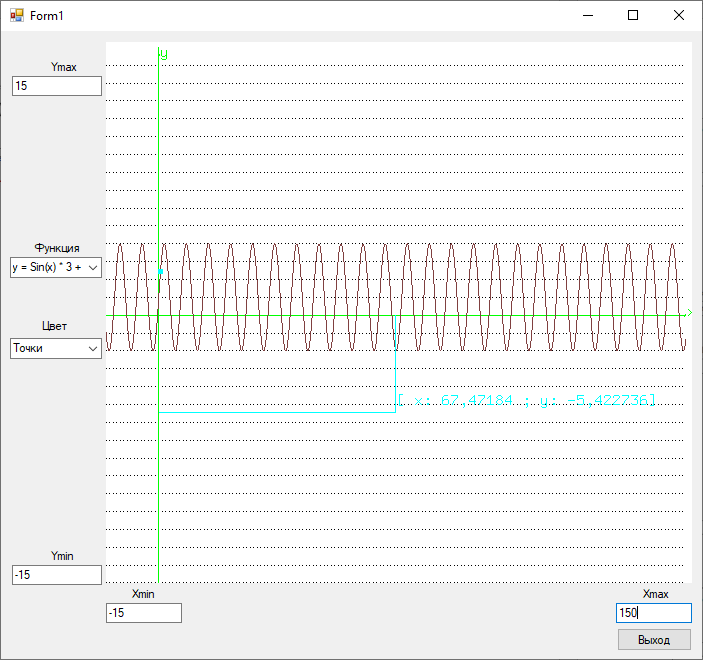
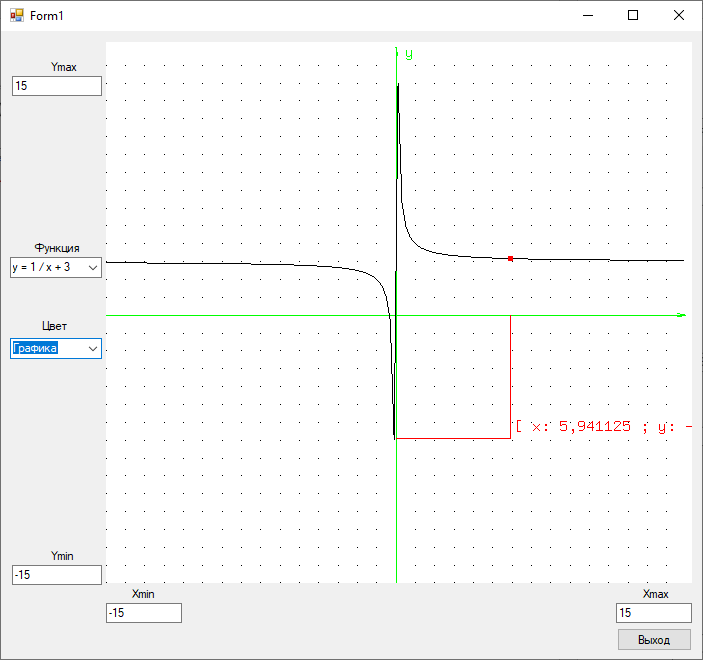
}

}

}

**Результаты работы приложения**

****

****