УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Общенаучный факультет

Кафедра ВВТиС

Отчет по лабораторной работе № 2

Тема «Создание графических приложений в системе Windows с использованием OpenTK и OpenGL».

Группа ПМИ-248

Студент Фаизова А.Н.

(дата) (подпись) (Фамилия И.О.)

Проверил Мухтаров А.Р.

(дата) (подпись) (Фамилия И.О.)

Уфа 2018

**Цель:** изучив теоретический материал по визуализации данных при помощи OpenTK и OpenGL, разработать приложение на С# для визуализации функций.

**Индивидуальное задание:**

Разработать приложение на С# для визуализации функции y = 3x2-x3 (кубическая парабола).

**Описание программы:**

В программе анимировано демонстрируется, как меняется значение функции на графике. Реализуется это следующим образом: по графику двигается красная точка, принимающая значения у для заданного х в нашем графике (по всей видимой области).

Для создания приложения используется элемент GLControl. Для корректной анимации движения точки добавили в нашу форму объект «таймер».

Исходный код основан на 7 функциях:

GL.Viewport();

/\* Определяем значение порта вывода, устанавливая значения в данной функции \*/

GL.MatrixMode(MatrixMode.Projection);

/\* Функция предназначена для того, чтобы задавать матричный режим: будет определена матрица, над которой мы в дальнейшем будем производить операции. В нашем случае это projection – матрица проекций \*/

GL.LoadIdentity();

/\* Очищаем матрицу – меняем на единичную\*/

GL.Ortho();

/\* Устанавливаем тип текущей проекции – ортогональная проекция \*/

GL.MatrixMode(MatrixMode.Modelview);

/\* Устанавливаем в качестве текущей матрицы - объектновидовую матрицу и очищаем ее \*/

void DrawDiagram();

/\* Функция, выполняющая визуализацию графика \*/

void Draw();

/\* Функция , в которой визуализируется координатная сетка под графиком, координатные, а так же вызываемся функция рисования графика, выводятся линии, соединяющие указатель мыши и оси координат. \*/

Исходный 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;

using OpenTK;

using OpenTK.Graphics;

using OpenTK.Graphics.OpenGL;

namespace LLL

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

double ScreenW, ScreenH;

private float devX;

private float devY;

private float[,] GrapValuesArray;

private int elements\_count = 0;

private bool not\_calculate = true;

private int pointPosition = 0;

float lineX, lineY;

float Mcoord\_X = 0, Mcoord\_Y = 0;

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

{

GL.Viewport(0, 0, AnT.Width, AnT.Height);

GL.MatrixMode(MatrixMode.Projection);

GL.LoadIdentity();

GL.Ortho(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0, -1, 1);

GL.MatrixMode(MatrixMode.Modelview);

}

// определение параметров настройки проекции в зависимости от размеров сторон элемента AnT

private void Form1\_Load(object sender, EventArgs e)

{

if ((float)AnT.Width <= (float)AnT.Height)

{

ScreenW = 30.0;

ScreenH = 30.0 \* (float)AnT.Height / (float)AnT.Width;

GL.Ortho(0.0, ScreenW, 0.0, ScreenH, -1, 1);

}

else

{

ScreenH = 30.0;

ScreenW = 30.0 \* (float)AnT.Width / (float)AnT.Height;

GL.Ortho(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0, -1, 1);

}

// сохраниение коэф-тов, необходимых для перевода указателя в оконной системе в координаты в нашей OpenGL сцене

devX = (float)ScreenW / (float)AnT.Width;

devY = (float)ScreenH / (float)AnT.Height;

// установка объектно-видовой матрицы

GL.MatrixMode(MatrixMode.Modelview);

PointInGrap.Start();

}

private void PointInGrap\_Tick(object sender, EventArgs e)

{

if (pointPosition == elements\_count - 1)

pointPosition = 0;

Draw();

pointPosition++;

}

private void AnT\_MouseMove(object sender, MouseEventArgs e)

{

Mcoord\_X = e.X;

Mcoord\_Y = e.Y;

lineX = devX \* e.X;

lineY = (float)(ScreenH - devY \* e.Y);

}

private void functionCalculation()

{

float x = 0, y = 0;

GrapValuesArray = new float[300, 2];

elements\_count = 0;

for (x = -5; x < 5; x += 0.1f)

{

y = 3 \* x \* x - x \* x \* x;

GrapValuesArray[elements\_count, 0] = x;

GrapValuesArray[elements\_count, 1] = y;

elements\_count++;

}

not\_calculate = false;

}

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.Begin(PrimitiveType.LineStrip);

// рисуем начальную точку

GL.Vertex2(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.Vertex2(GrapValuesArray[ax, 0], GrapValuesArray[ax, 1]);

// завершаем режим рисования

GL.End();

// устанавливаем размер точек, равный 5 пикселям

GL.PointSize(5);

// устанавливаем текущим цветом - красный

GL.Color3(Color.Red);

// активируем режим вывода точек

GL.Begin(PrimitiveType.Points);

//выводим красную точку, используя ту ячейку массива, до к-й

//мы дошли( вычисляется в функции обработчике событий таймера)

GL.Vertex2(GrapValuesArray[pointPosition, 0], GrapValuesArray[pointPosition, 1]);

GL.End();

}

private void Draw()

{

GL.ClearColor(Color.Bisque);

// очистка буфера цвета и очистка буфера глубины

GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

GL.LoadIdentity();

// установка черного цвета

GL.Color3(0, 0, 0);

// помещаем состояние матрицы в стек матриц

GL.PushMatrix();

// выполняем перемещение в пространстве по осям X и Y

GL.Translate(15, 15, 0);

// активируем режим рисования

GL.Begin(PrimitiveType.Points);

// создаем сетку из точек

for (float ax = -15; ax < 15; ax++)

{

for(float bx = -15; bx <15; bx++)

{

// вывод точки

GL.Vertex2(ax, bx);

}

}

// завершение режима рисования примитивов

GL.End();

// активируем режим рисования, каждые 2 последовательно вызванные команды glVertex объединяются в линии

GL.Begin(PrimitiveType.Lines);

// рисуем координатные оси и стрелки на их концах

GL.Vertex2(0, -15);

GL.Vertex2(0, 15);

GL.Vertex2(-15, 0);

GL.Vertex2(15, 0);

GL.Vertex2(0, 15);

GL.Vertex2(0.1, 14.5);

GL.Vertex2(0, 15);

GL.Vertex2(-0.1, 14.5);

GL.Vertex2(15, 0);

GL.Vertex2(14.5, 0.1);

GL.Vertex2(15, 0);

GL.Vertex2(14.5, -0.1);

GL.End();

// вызываем функцию рисования графика

DrawDiagram();

GL.PointSize(1);

// возвращаем матрицу из стека

GL.PopMatrix();

GL.Color3(Color.Red);

// включаем режим рисования линий, для того чтобы нарисовать линии от курсора мыши к координатным осям

GL.Begin(PrimitiveType.Lines);

GL.Vertex2(lineX, 15);

GL.Vertex2(lineX, lineY);

GL.Vertex2(15, lineY);

GL.Vertex2(lineX, lineY);

GL.End();

// переключаем буфер вывода

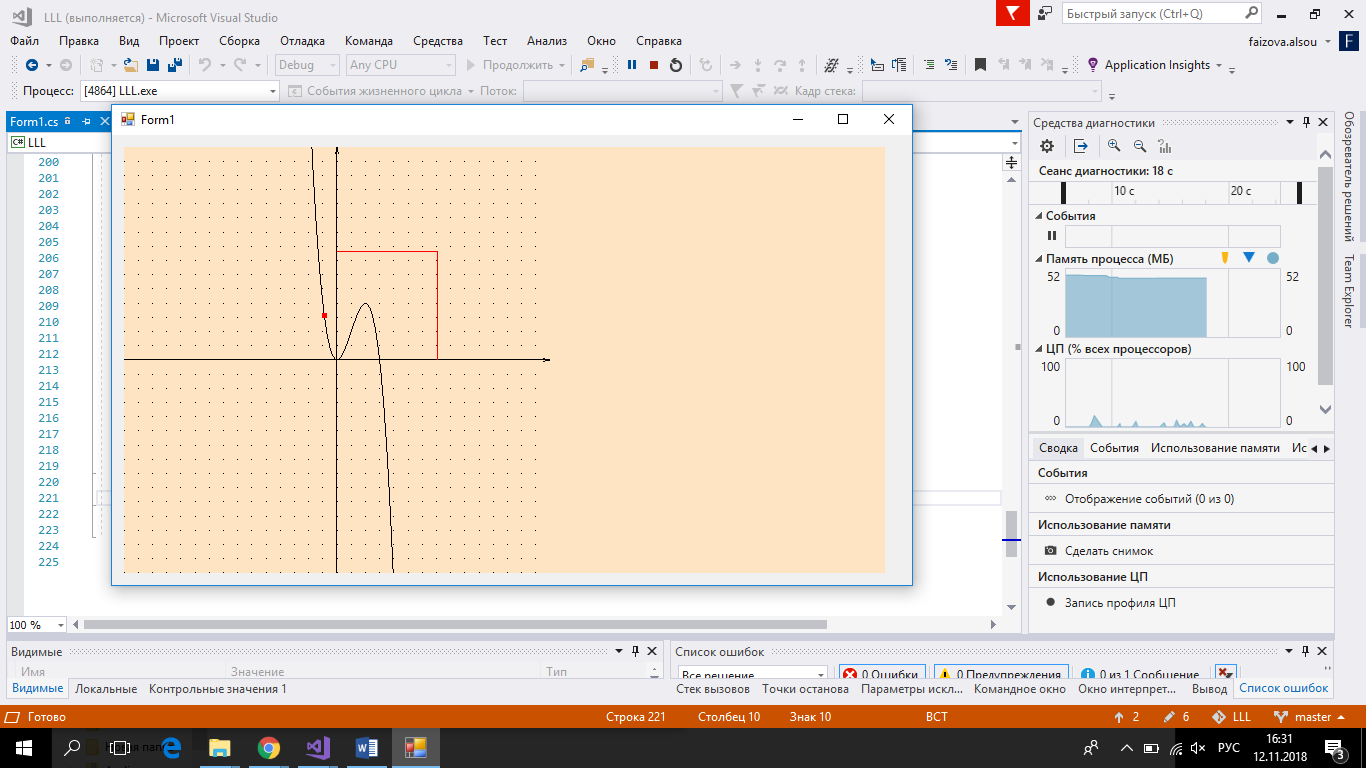
AnT.SwapBuffers();

}

}

}

Пример выполнения программы:



**Вывод:**

На данной лабораторной работе мы создали проект, подключили к нему библиотеку OpenGL. Программа анимировано демонстрирует, как меняются значения функции на графике; выводятся линии, соединяющие указатель мыши и оси координат.