Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем

и программной инженерии

**Лабораторная работа № 5**

**по дисциплине**

**«Программирование компьютерной графики»**

Тема работы: Инициализация OpenGL

Выполнил:

ст. гр. ПРИ-120

М. А. Бочков

Принял:

преп. каф. ИСПИ

Жигалов И.Е.

Владимир, 2023

**Цель работы:**

Изучение принципов инициализация OpenGL и визуализация объектов при разработке приложений в C#.

**Ход работы:**

**Вариант №6**

**Задание**

1. Ознакомиться по методическим указаниям и литературе с теоретическим материалом.

2. Выполнить действия, приведенные в разделах 5.2, 5.3, 5.4. При разработке программы имя проекта, создаваемого в MS Visual Studio, должно содержать фамилию студента и группу (например, Ivanov\_Ivan\_ISG\_105\_lab\_1).

3. При выполнении задания для отрисовки линий использовать цвет, указанный в варианте задания.

4. При рисовании устанавливать толщину линий (функция glLineWidth) из варианта задания.

5. При выполнении задания раздела 5.2 визуализировать с помощью линий свою фамилию и инициалы.

6. При выполнении задания раздела 5.3 использовать функцию, указанную в варианте.

7. При выполнении задания раздела 5.4 добавить в окно и в его заголовок свою фамилию, имя и группу.

**Задание 1**

Листинг C#:

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 Bochkov\_Mikhail\_PRI\_120\_lab\_5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

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

{

// инициализация Glut

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

// очистка окна

Gl.glClearColor(255, 255, 255, 1);

// установка порта вывода в соотвествии с размерами элемента anT

Gl.glViewport(0, 0, AnT.Width, AnT.Height);

// настройка проекции

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

// теперь необходимо корректно настроить 2D ортогональную проекцию

// в зависимости от того, какая сторона больше

// мы немного варьируем то, как будут сконфигурированы настройки проекции

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

{

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Height / (float)AnT.Width, 0.0, 30.0);

}

else

{

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

}

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

// настройка параметров OpenGL для визуализации

Gl.glEnable(Gl.GL\_DEPTH\_TEST);

Gl.glEnable(Gl.GL\_COLOR\_MATERIAL);

}

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

{

Gl.glLineWidth(1.1F);

// очищаем буфер цвета и глубины

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT | Gl.GL\_DEPTH\_BUFFER\_BIT);

// очищаем текущую матрицу

Gl.glLoadIdentity();

// устанавливаем текущий цвет - красный

Gl.glColor3f(139, 0, 255);

// активируем режим рисования линий на основе

// последовательного соединения всех вершин в отрезки

Gl.glBegin(Gl.GL\_LINE\_LOOP);

// первая вершина будет находиться в начале координат

Gl.glVertex2d(4, 30);

Gl.glVertex2d(1, 30);

Gl.glVertex2d(1, 25);

Gl.glVertex2d(4, 25);

Gl.glVertex2d(4, 28);

Gl.glVertex2d(1, 28);

Gl.glVertex2d(1, 29);

Gl.glVertex2d(4, 29);

Gl.glVertex2d(4, 30);

// завершаем режим рисования

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(8, 30);

Gl.glVertex2d(6, 30);

Gl.glVertex2d(6, 25);

Gl.glVertex2d(8, 25);

Gl.glVertex2d(8, 30);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(12, 30);

Gl.glVertex2d(12, 25);

Gl.glVertex2d(12, 27.5);

Gl.glVertex2d(10, 27.5);

Gl.glVertex2d(10, 30);

Gl.glVertex2d(10.6, 30);

Gl.glVertex2d(10.6, 28);

Gl.glVertex2d(11.6, 28);

Gl.glVertex2d(11.6, 30);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(14, 30);

Gl.glVertex2d(14, 25);

Gl.glVertex2d(14, 27.5);

Gl.glVertex2d(16, 30);

Gl.glVertex2d(14, 27.5);

Gl.glVertex2d(16, 25);

Gl.glVertex2d(14, 27.5);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(20, 30);

Gl.glVertex2d(18, 30);

Gl.glVertex2d(18, 25);

Gl.glVertex2d(20, 25);

Gl.glVertex2d(20, 30);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(22, 30);

Gl.glVertex2d(24, 30);

Gl.glVertex2d(24, 28);

Gl.glVertex2d(22, 28);

Gl.glVertex2d(22, 25);

Gl.glVertex2d(24, 25);

Gl.glVertex2d(24, 27);

Gl.glVertex2d(22, 27);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(28, 25);

Gl.glVertex2d(30, 30);

Gl.glVertex2d(32, 25);

Gl.glVertex2d(34, 30);

Gl.glVertex2d(36, 25);

Gl.glVertex2d(35.5, 25);

Gl.glVertex2d(34, 29);

Gl.glVertex2d(32.5, 25);

Gl.glVertex2d(31.5, 25);

Gl.glVertex2d(30, 29);

Gl.glVertex2d(28.5, 25);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_LOOP);

Gl.glVertex2d(38, 25);

Gl.glVertex2d(40, 30);

Gl.glVertex2d(42, 25);

Gl.glVertex2d(41, 27.5);

Gl.glVertex2d(39, 27.5);

Gl.glEnd();

// дожидаемся конца визуализации кадра

Gl.glFlush();

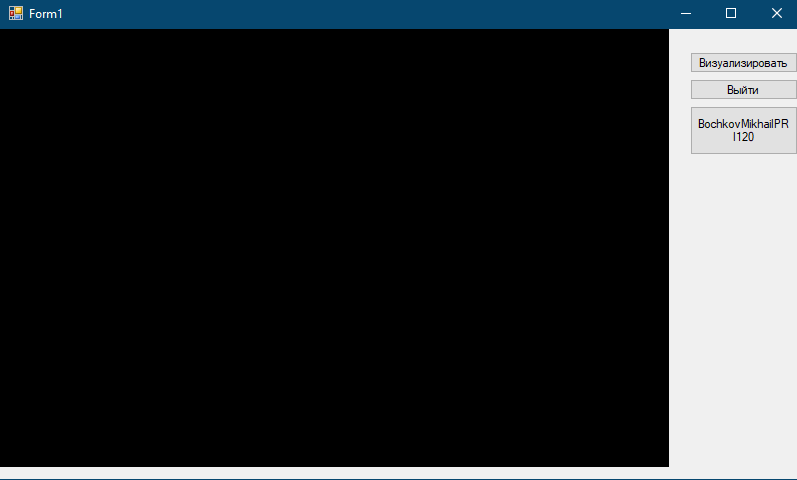
// посылаем сигнал перерисовки элемента AnT.

AnT.Invalidate();

}

}

}



Скриншот 1- Стартовое окно



Скриншот 2- окно после визуализации

**Задание 1**

Листинг C#:

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 Bochkov\_Mikhail\_PRI\_120\_lab\_52

{

public partial class Form1 : Form

{

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 double[,] 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);

}

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)

{

// визуализируем символ с помощью функции glutBitmapCharacter, используя шрифт GLUT\_BITMAP\_9\_BY\_15

Glut.glutBitmapCharacter(Glut.GLUT\_BITMAP\_9\_BY\_15, char\_for\_draw);

}

}

private void functionCalculation()

{

// определение локальных переменных X и Y

double x = 0, y = 0;

// инициализация массива, который будет хранить значение 300 точек

// из которых будет состоять график

GrapValuesArray = new double[300, 2];

// счетчик элементов массива

elements\_count = 0;

// вычисления всех значений y для x, пренадлежащего промежутку от -15 до 15, с шагом в 0.01f

for (x = -3; x < 3; 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 = (double)3 \* (x + 3) / (Math.Pow(x,2)+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 вычисленны

not\_calculate = false;

}

private void DrawDiagram()

{

Gl.glLineWidth(1.1F);

Gl.glColor3f(139, 0, 255);

// 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 текущий цвет - красный

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 функии-обработчике событий таймера)

Gl.glVertex2d(GrapValuesArray[pointPosition, 0], GrapValuesArray[pointPosition, 1]);

// завершаем режим рисования

Gl.glEnd();

// устанавливаем размер точек, равный еденице

Gl.glPointSize(1);

}

private void Draw()

{

// очистка буфера цвета и буфера глубины

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT | Gl.GL\_DEPTH\_BUFFER\_BIT);

// очистка текущей матрицы

Gl.glLoadIdentity();

// установка черного цвета

Gl.glColor3f(0, 0, 0);

// помещаем состояние матрицы в стек матриц

Gl.glPushMatrix();

// выполняем перемещение в прострастве по осям X и Y

Gl.glTranslated(15, 15, 0);

// активируем рижим рисования (указанные далее точки будут выводиться как точки GL\_POINTS)

Gl.glBegin(Gl.GL\_POINTS);

// с помощью прохода двумя 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 две точки, последовательно вызванные коммандой glVertex,

// объединяются в линию

Gl.glBegin(Gl.GL\_LINES);

// рисуем координатные оси и стрелки на них

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 бибилиотеки 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 коэфицентов, 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();

}

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();

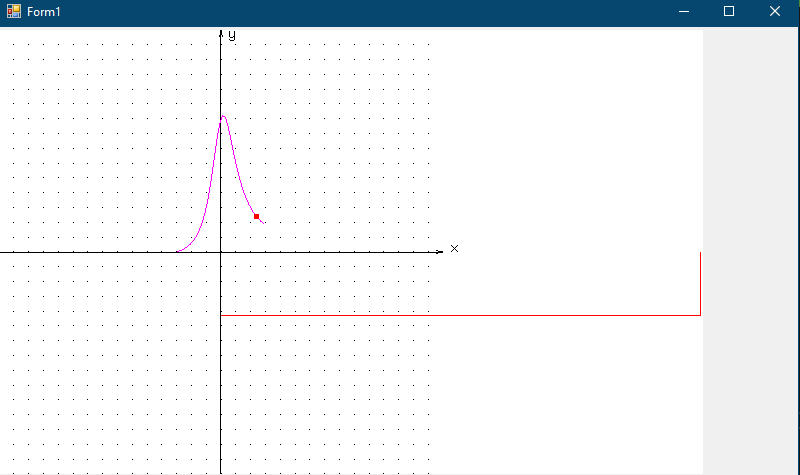
// переход к следующему элементу массива

pointPosition++;

}

}

}



Скриншот 3 - Результат выполнения задания 2

**Задание 3**

Листинг C#:

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 Bochkov\_Mikhail\_PRI\_120\_lab\_53

{

public partial class Form1 : Form

{

double a = 1, b = 0, c = 0;

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

// генерация коэфицента

a = (double)trackBar1.Value / 1000.0;

// вывод значения коэфицента, управляемого данным ползунком.

// (под TrackBa'ом)

label4.Text = a.ToString();

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

// генерация коэфицента

b = (double)trackBar2.Value / 1000.0;

// вывод значения коэфицента, управляемого данным ползунком.

// (под TrackBa'ом)

label5.Text = b.ToString();

}

private void trackBar3\_Scroll(object sender, EventArgs e)

{

// генерация коэфицента

c = (double)trackBar3.Value / 1000.0;

// вывод значения коэфицента, управляемого данным ползунком.

// (под TrackBa'ом)

label6.Text = c.ToString();

}

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

{

// страт таймера, отвечающего за вызов фукнции,

// визуализирующей кадр

RenderTimer.Start();

}

private void button2\_Click(object sender, EventArgs e)

{

// выход из приложения

Application.Exit();

}

private void timer1\_Tick(object sender, EventArgs e)

{

// функция визуализации

Draw();

}

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

{

// инициализация библиотеки GLUT

Glut.glutInit();

// инициализация режима окна

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

// устанавливаем цвет очистки окна

Gl.glClearColor(255, 255, 255, 1);

// устанавливаем порт вывода, основываясь на размерах элемента управления AnT

Gl.glViewport(0, 0, AnT.Width, AnT.Height);

// устанавливаем проекционную матрицу

Gl.glMatrixMode(Gl.GL\_PROJECTION);

// очищаем ее

Gl.glLoadIdentity();

// теперь необходимо корректно настроить 2D ортогональную проекцию

// в зависимости от того, какая сторона больше

// мы немного варьируем конфигурацией настройки проекции

if (AnT.Width <= AnT.Height)

Glu.gluOrtho2D(0.0, 30.0, 0.0, 30.0 \* (float)AnT.Height / (float)AnT.Width);

else

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

// переходим к объектно-видовой матрице

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

}

private void Draw()

{

// очищаем буфер цвета

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT);

// активируем рисование в режиме GL\_TRIANGLES, при котором три заданные

// с помощью функции glVertex2d или glVertex3d вершины

// будут объединяться в полигон (треугольник)

Gl.glBegin(Gl.GL\_TRIANGLES);

// устанавливаем параметр цвета, основанный на параметрах a b c

Gl.glColor3d(0.0 + a, 0.0 + b, 0.0 + c);

// рисуем вершину в координатах 5,5

Gl.glVertex2d(5.0, 5.0);

// устанавливаем параметр цвета , основанный на параметрах с a b

Gl.glColor3d(0.0 + c, 0.0 + a, 0.0 + b);

// рисуем вершину в координатах 25,5

Gl.glVertex2d(25.0, 5.0);

// устанавливаем параметр цвета , основанный на параметрах b c a

Gl.glColor3d(0.0 + b, 0.0 + c, 0.0 + a);

// рисуем вершину в координатах 25,5

Gl.glVertex2d(5.0, 25.0);

// завершаем режим рисования примитивов

Gl.glEnd();

// дожидаемся завершения визуализации кадра

Gl.glFlush();

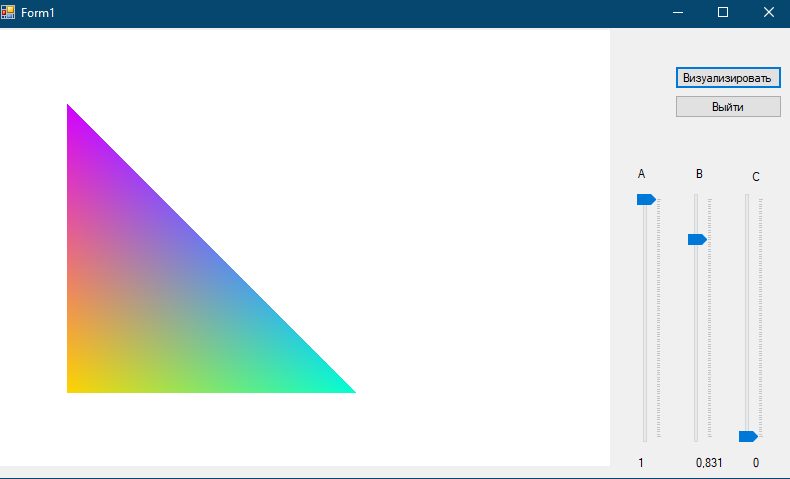
// обновляем изображение в элементе AnT

AnT.Invalidate();

}

}

}



Скриншот 4 - Результат выполнения задания №3

**Вывод**

В процессе выполнения работы были изучены принципы инициализации OpenGL и визуализация объектов при разработке приложений в С#;