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 №2.3
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 № 2.3

Рисование «вывески сизображением (животного, насекомого, рыбок и т.д.)» Реализовать отрисовку «вывески», с учетом того, что каждая «вывеска» должна рисоваться в «условном» прямоугольнике, для которого заданы:

- W – ширина,
- H – высота,
- X0, Y0- точка привязки левого нижнего угла.

Листинг кода

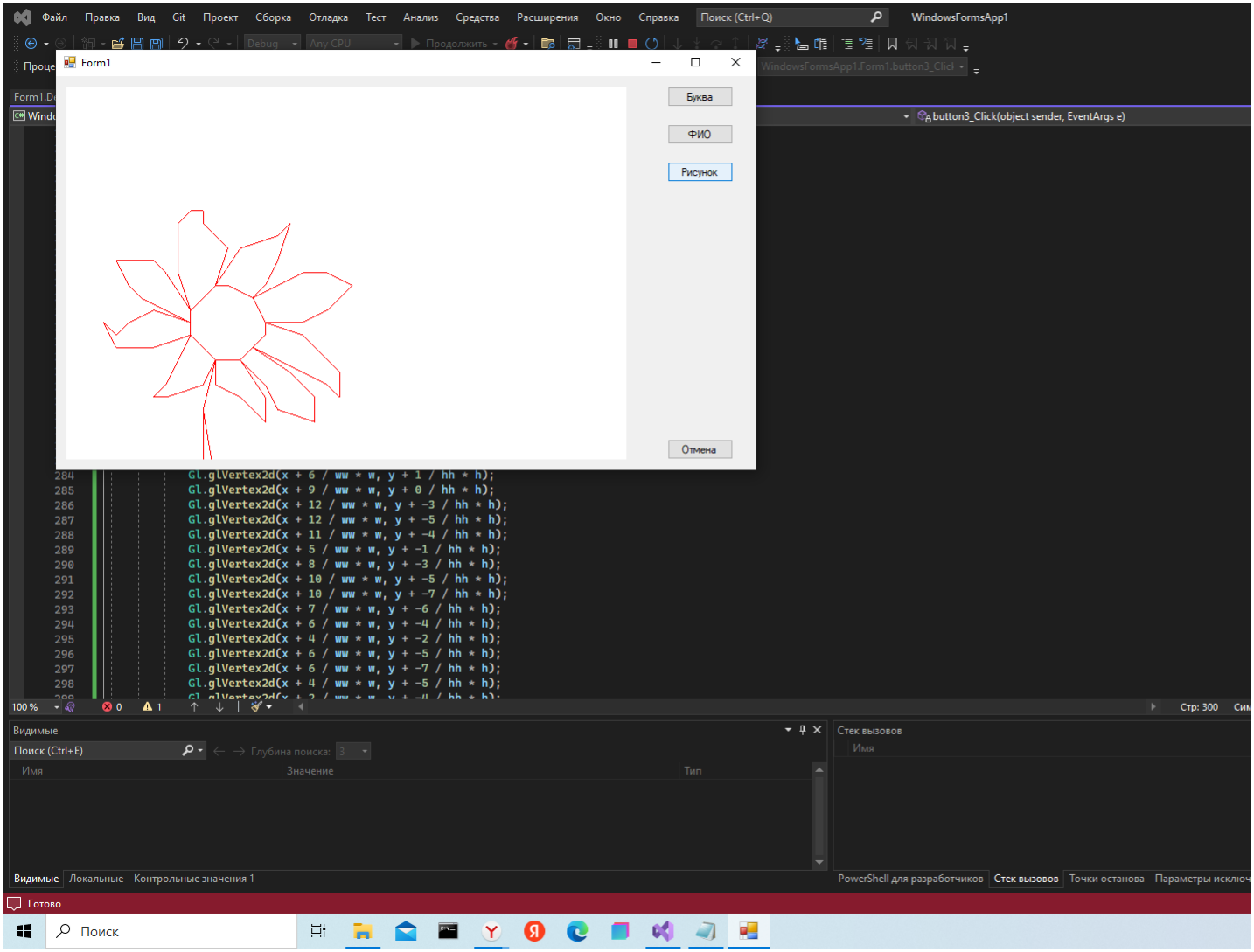
```
 using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tao.FreeGlut;
using Tao.OpenGl;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            AnT.InitializeContexts();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // очищаем буфер цвета
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);

            // очищаем текущую матрицу
            Gl.glLoadIdentity();
            // устанавливаем текущий цвет - красный
            Gl.glColor3f(255, 0, 0);


            // 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(8, 7);
            Gl.glVertex2d(15, 27);
            Gl.glVertex2d(17, 27);
            Gl.glVertex2d(23, 7);
            Gl.glVertex2d(21, 7);
            Gl.glVertex2d(19, 14);
            Gl.glVertex2d(12.5, 14);
            Gl.glVertex2d(10, 7);

            // завершаем режим рисования
            Gl.glEnd();

            // вторая линия

            Gl.glBegin(Gl.GL_LINE_LOOP);

            Gl.glVertex2d(18.5, 16);
            Gl.glVertex2d(16, 25);
            Gl.glVertex2d(13.2, 16);

            // завершаем режим рисования
            Gl.glEnd();



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

            // посылаем сигнал перерисовки элемента AnT.
            AnT.Invalidate();
        }


        private void simpleOpenGlControl1_Load(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // инициализация Glut
            Glut.glutInit();
            Glut.glutInitDisplayMode(Glut.GLUT_RGB | Glut.GLUT_SINGLE);

            // очистка окна
            Gl.glClearColor(255, 255, 255, 1);

            // установка порта вывода в соответствии с размерами элемента anT
            Gl.glViewport(0, 0, AnT.Width, AnT.Height);

            // настройка проекции
```

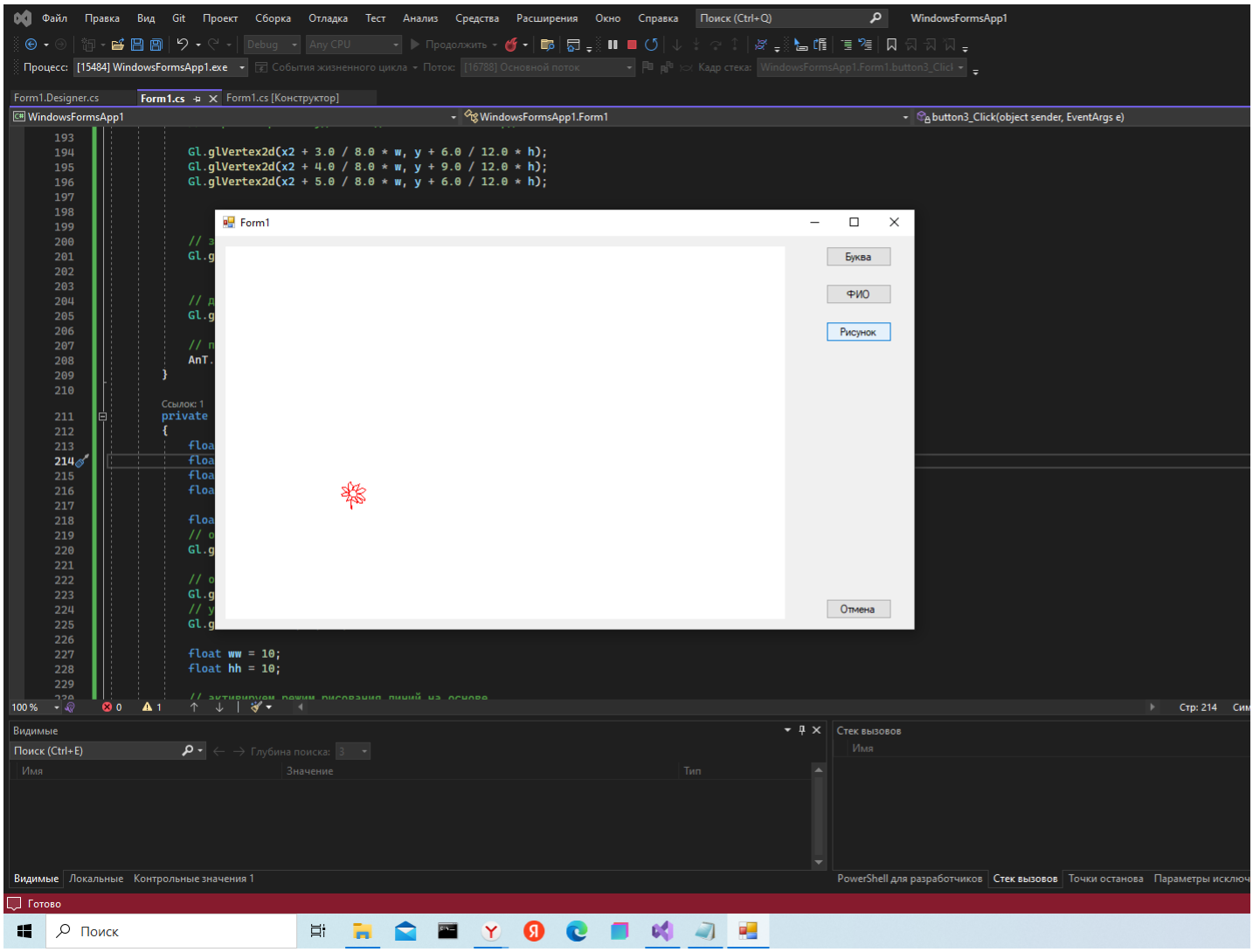
```
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();

    // теперь необходимо корректно настроить 2D ортогональную проекцию
    // в зависимости от того, какая сторона больше
    // мы немного варьируем то, как будет сконфигурированный 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();
}

private void button2_Click(object sender, EventArgs e)
{
    float w = 6;
    float h = 4;
    float x = 1;
    float y = 1;

    float d = 1;
    // очищаем буфер цвета
    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);

    // очищаем текущую матрицу
    Gl.glLoadIdentity();
    // устанавливаем текущий цвет - красный
    Gl.glColor3f(255, 0, 0);


    // 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(x + 0.0 / 8.0 * w, y + 4.0 / 12.0 * h);
    Gl.glVertex2d(x + 0.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x + 2.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x + 2.0 / 8.0 * w, y + 6.0 / 12.0 * h);
    Gl.glVertex2d(x + 6.0 / 8.0 * w, y + 6.0 / 12.0 * h);
    Gl.glVertex2d(x + 6.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x + 8.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x + 8.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x + 6.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x + 6.0 / 8.0 * w, y + 4.0 / 12.0 * h);

    // завершаем режим рисования
    Gl.glEnd();

    // вторая линия

    float x1 = x + d + w;


    Gl.glBegin(Gl.GL_LINE_LOOP);
    // первая вершина будет находиться в начале координат

    Gl.glVertex2d(x1 + 0.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 0.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 2.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 2.0 / 8.0 * w, y + 7.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 6.0 / 8.0 * w, y + 7.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 6.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 8.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 8.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 6.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 6.0 / 8.0 * w, y + 6.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 2.0 / 8.0 * w, y + 6.0 / 12.0 * h);
    Gl.glVertex2d(x1 + 2.0 / 8.0 * w, y + 0.0 / 12.0 * h);

    // завершаем режим рисования
    Gl.glEnd();

    float x2 = x + 2 * d + 2 * w;


    Gl.glBegin(Gl.GL_LINE_LOOP);
    // первая вершина будет находиться в начале координат

    Gl.glVertex2d(x2 + 0.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 4.0 / 8.0 * w, y + 12.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 8.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 6.0 / 8.0 * w, y + 0.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 5.0 / 8.0 * w, y + 5.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 3.0 / 8.0 * w, y + 5.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 2.0 / 8.0 * w, y + 0.0 / 12.0 * h);


    // завершаем режим рисования
    Gl.glEnd();

    Gl.glBegin(Gl.GL_LINE_LOOP);
    // первая вершина будет находиться в начале координат

    Gl.glVertex2d(x2 + 3.0 / 8.0 * w, y + 6.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 4.0 / 8.0 * w, y + 9.0 / 12.0 * h);
    Gl.glVertex2d(x2 + 5.0 / 8.0 * w, y + 6.0 / 12.0 * h);
```

```
        // завершаем режим рисования
        Gl.glEnd();


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

        // посылаем сигнал перерисовки элемента AnT.
        AnT.Invalidate();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        float w = 1;
        float h = 1;
        float x = 10;
        float y = 10;

        float d = 1;
        // очищаем буфер цвета
        Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);

        // очищаем текущую матрицу
        Gl.glLoadIdentity();
        // устанавливаем текущий цвет - красный
        Gl.glColor3f(255, 0, 0);

        float ww = 10;
        float hh = 10;

        // 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(x + 1 / ww * w, y + -6 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + -12 / hh * h);
        Gl.glVertex2d(x + 1 / ww * w, y + -11 / hh * h);
        Gl.glVertex2d(x + 1 / ww * w, y + -6 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + -2 / hh * h);
        Gl.glVertex2d(x + 4 / ww * w, y + -2 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + 0 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + 5 / ww * w, y + 3 / hh * h);
        Gl.glVertex2d(x + 3 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 2 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 0 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + -2 / hh * h);
        Gl.glVertex2d(x + 1 / ww * w, y + -4 / hh * h);
        Gl.glVertex2d(x + -2 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + -3 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + -2 / ww * w, y + -4 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 0 / hh * h);
        Gl.glVertex2d(x + -3 / ww * w, y + -1 / hh * h);
        Gl.glVertex2d(x + -6 / ww * w, y + -1 / hh * h);
        Gl.glVertex2d(x + -7 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + -6 / ww * w, y + 0 / hh * h);
        Gl.glVertex2d(x + -5 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + -3 / ww * w, y + 2 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + -4 / ww * w, y + 3 / hh * h);
        Gl.glVertex2d(x + -5 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + -6 / ww * w, y + 6 / hh * h);
        Gl.glVertex2d(x + -3 / ww * w, y + 6 / hh * h);
        Gl.glVertex2d(x + -2 / ww * w, y + 5 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 2 / hh * h);
        Gl.glVertex2d(x + -1 / ww * w, y + 5 / hh * h);
        Gl.glVertex2d(x + -1 / ww * w, y + 9 / hh * h);
        Gl.glVertex2d(x + 0 / ww * w, y + 10 / hh * h);
        Gl.glVertex2d(x + 1 / ww * w, y + 10 / hh * h);
        Gl.glVertex2d(x + 1 / ww * w, y + 9 / hh * h);
        Gl.glVertex2d(x + 3 / ww * w, y + 7 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + 4 / ww * w, y + 7 / hh * h);
        Gl.glVertex2d(x + 7 / ww * w, y + 8 / hh * h);
        Gl.glVertex2d(x + 8 / ww * w, y + 9 / hh * h);
        Gl.glVertex2d(x + 7 / ww * w, y + 6 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + 5 / ww * w, y + 3 / hh * h);
        Gl.glVertex2d(x + 9 / ww * w, y + 5 / hh * h);
        Gl.glVertex2d(x + 11 / ww * w, y + 5 / hh * h);
        Gl.glVertex2d(x + 13 / ww * w, y + 4 / hh * h);
        Gl.glVertex2d(x + 11 / ww * w, y + 2 / hh * h);
        Gl.glVertex2d(x + 9 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + 1 / hh * h);
        Gl.glVertex2d(x + 9 / ww * w, y + 0 / hh * h);
        Gl.glVertex2d(x + 12 / ww * w, y + -3 / hh * h);
        Gl.glVertex2d(x + 12 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + 11 / ww * w, y + -4 / hh * h);
        Gl.glVertex2d(x + 5 / ww * w, y + -1 / hh * h);
        Gl.glVertex2d(x + 8 / ww * w, y + -3 / hh * h);
        Gl.glVertex2d(x + 10 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + 10 / ww * w, y + -7 / hh * h);
        Gl.glVertex2d(x + 7 / ww * w, y + -6 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + -4 / hh * h);
        Gl.glVertex2d(x + 4 / ww * w, y + -2 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + 6 / ww * w, y + -7 / hh * h);
        Gl.glVertex2d(x + 4 / ww * w, y + -5 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + -4 / hh * h);
        Gl.glVertex2d(x + 2 / ww * w, y + -2 / hh * h);

        // завершаем режим рисования
        Gl.glEnd();
```

```
            // дожидаемся конца визуализации кадра
            Gl.glFlush();

            // посылаем сигнал перерисовки элемента AnT.
            AnT.Invalidate();
        }
    }
}
```

Скриншот




Файл Правка Вид Git Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Поиск (Ctrl+Q)
WindowsFormsApp1
Form1.Designer.cs
Form1.cs
Form1.cs [Конструктор]
WindowsFormsApp1.Form1
button3_Click(object sender, EventArgs e)
Gl.glVertex2d(x2 + 3.0 / 8.0 * w, y + 6.0 / 12.0 * h);
Gl.glVertex2d(x2 + 4.0 / 8.0 * w, y + 9.0 / 12.0 * h);
Gl.glVertex2d(x2 + 5.0 / 8.0 * w, y + 6.0 / 12.0 * h);
Form1
Буква
ФИО
Рисунок
Отмена
float ww = 10;
float hh = 10;
Видимые
Стек вызовов
Готово
Поиск