Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

Создание трехмерной сцены с применением OpenGL

Отчет о выполнении лабораторной работы
по дисциплине «Компьютерная графика»
Вариант №2

Выполнил:
студент группы 429-3
Бабец А. А.

Проверил:
Доцент каф. АОИ
ТУСУР,
канд. тех. наук
Т.О. Перемитина

2021

Содержание

Введение

Цель работы: получить навыки моделирования трехмерных объектов.

Постановка задачи:

• Изучение теоретических основ – Лекция "3D сцены с применением OpenGL" (стр. 72 – 85 и 122 – 125 учебного пособия https://edu.tusur.ru/training/publications/5613)

• Программная реализация построения трехмерной сцены согласно заданию.

Задание: согласно варианту задания построить трехмерную сцену с использованием двумерных примитивов OpenGl. Вращать объекты по таймеру.

Среда программирования

В качестве среды разработки был выбран Microsoft Visual Studio 2019. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, вебприложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Язык C# с применением компонента .NET Windows Forms включает в себя все компоненты, необходимые для выполнения задания лабораторной работы.

Для использования библиотеки OpenGL был подключен TaoFramework.

Метод решения задачи

Была инициализирована матрица для построения проволочного куба:

```
float[,] cube =

    {1f, 1.7f, 1f},
    {-1f, 1.7f, 1f},
    {-1f, 1.7f, -1f},
    {1f, 1.7f, -1f},

    {1f, -0.5f, 1f},
    {-1f, -0.5f, 1f},
    {-1f, -0.5f, -1f},
    {1f, -0.5f, -1f}
};
```

Далее была объявлена матрица для построения пирамиды:

```
float[,] pyramid =
{
    {0, -0.3f, -1f},
    {1f, -0.3f, -0.3f},
    {1f, -0.3f, 0.3f},
    {0, -0.3f, 1f},
    {-1f, -0.3f, 0.3f},
    {-1f, -0.3f, -0.3f},

    {0, 1.5f, 0}
};
```

Также был реализован метод drawFigure, который отрисовывает фигуры. Он содержит в себе следующее: поворот, отрисовка проволочного куба и отрисовка пирамиды. Далее приведен пример отрисовки проволочного куба:

```
//крышка куба
Gl.glColor3f(0.6f, 1, 0);
Gl.glBegin(Gl.GL_LINE_LOOP);
    for (int i = 0; i < 4; i++)
    {
        Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
    }
Gl.glEnd();

// дно куба
Gl.glBegin(Gl.GL_LINE_LOOP);
    for (int i = 4; i < 8; i++)
    {
        Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
    }
Gl.glEnd();

// боковые рёбра куба
for (int i = 0; i < 4; i++)
{
    Gl.glBegin(Gl.GL_LINE_STRIP);
        Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
        Gl.glVertex3f(cube[i+4, 0], cube[i+4, 1], cube[i+4, 2]);
    Gl.glEnd();
}
```

Дополнительно объявляем переменную для реализации вращения:

```
float yAngle = 0;
```

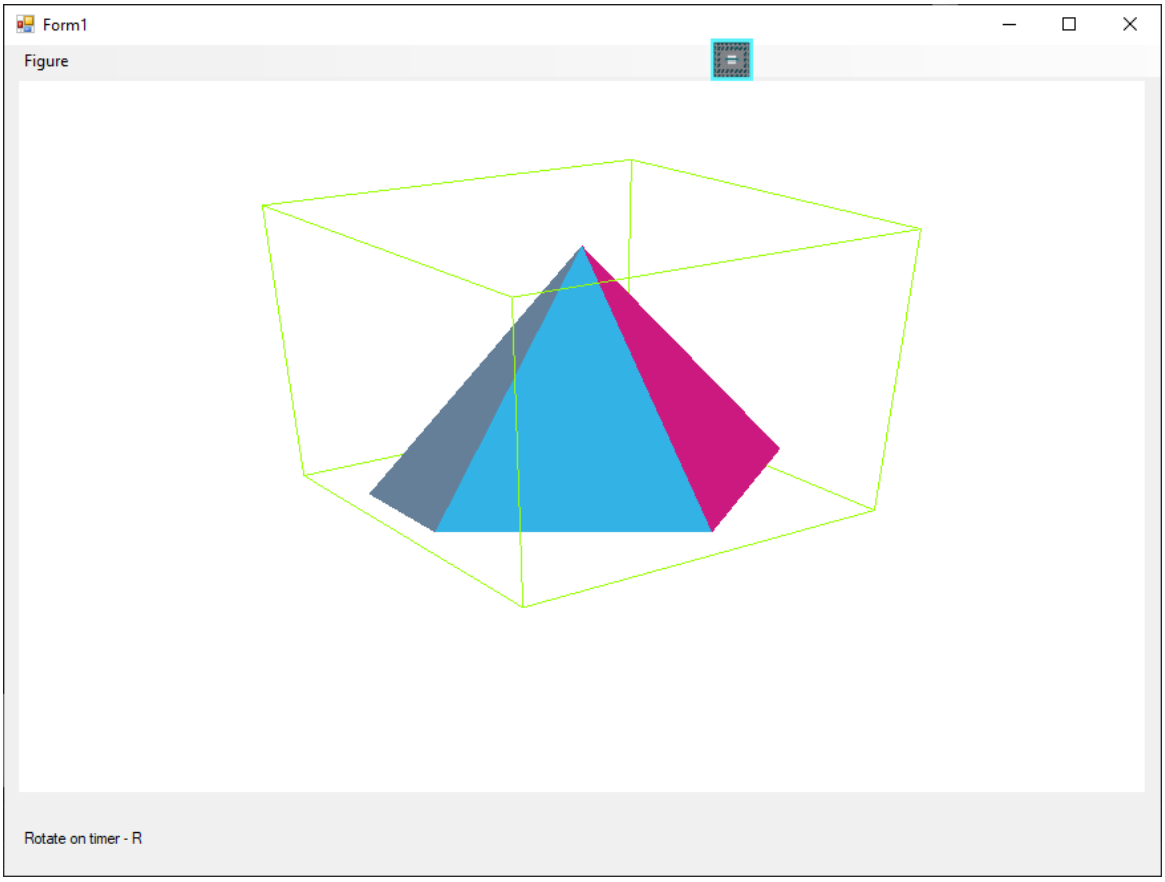
Добавляем объект таймер timer2 и в метод timer2_Tick записываем следующее:

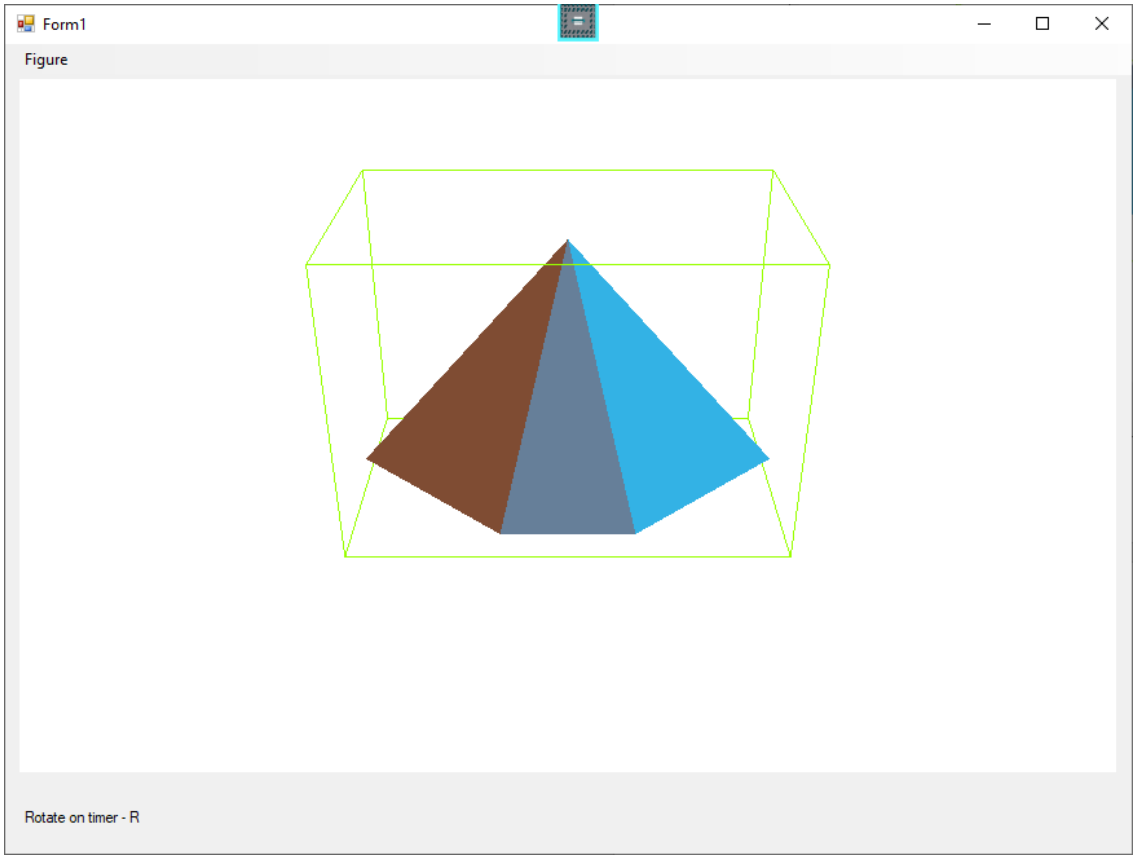
```
ссылка: 1
private void timer2_Tick(object sender, EventArgs e)
{
    yAngle = (yAngle + 5) % 360;
    DrawFigure();
}
```

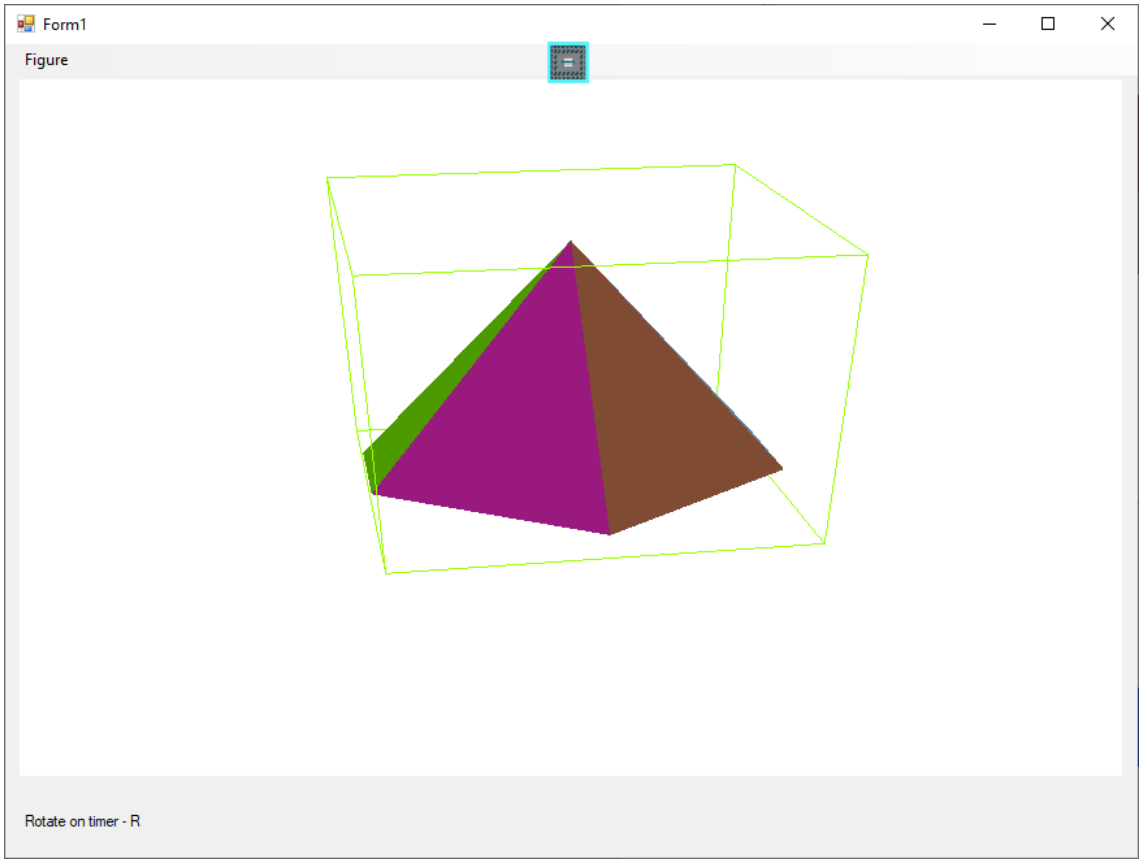
Теперь в метод Form1_Load добавляем следующие строки:

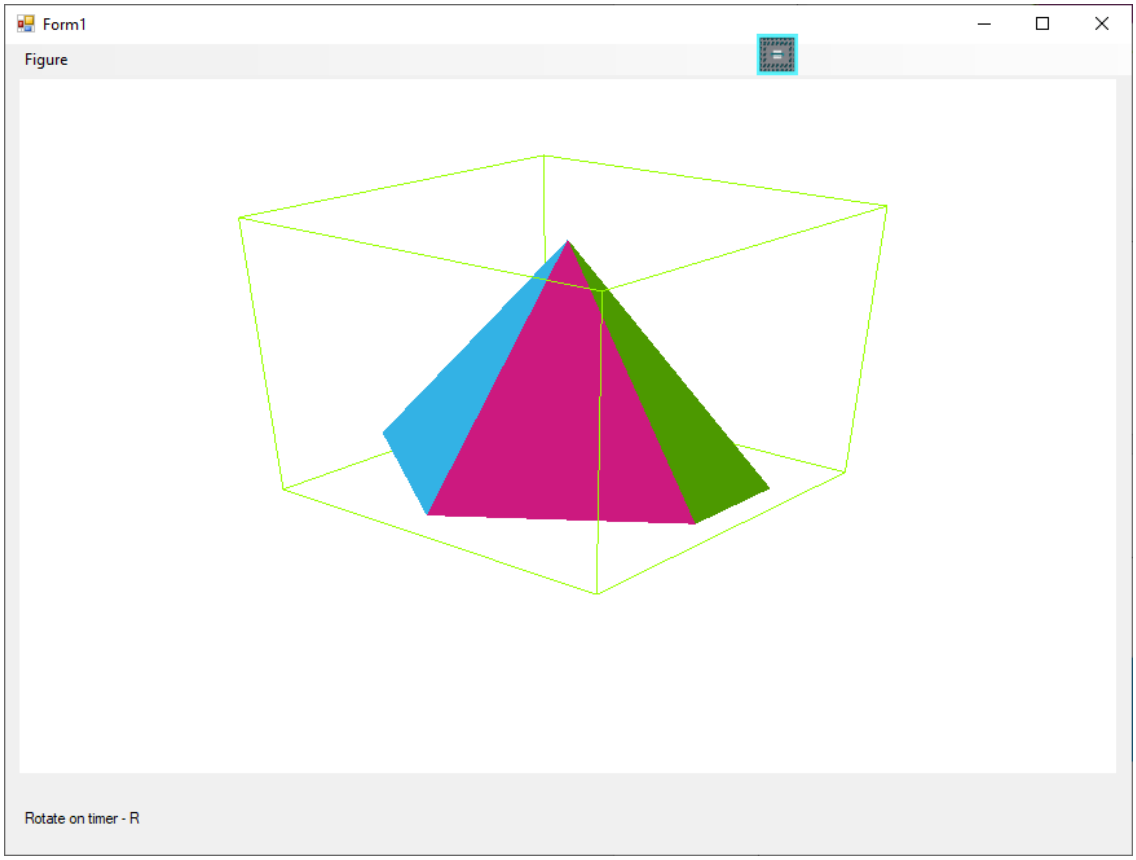
```
private void Form1_Load(object sender, EventArgs e)
{
    Gl.glEnable(Gl.GL_DEPTH_TEST);
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Gl.glFrustum(-1f, 1f, -1f, 1f, 3f, 10f);
    Gl.glMatrixMode(Gl.GL_MODELVIEW);
    Gl.glLoadIdentity();
    Gl.glTranslatef(0f, 0f, -8.0f);
    Gl.glRotatef(30, 1, 0, 0);
}
```

Что получили по итогу (представлено несколько скриншотов, сделанных во время выполнения программы):




Form1
Figure
Rotate on timer - R


Form1
Figure
Rotate on timer - R



С полным листингом программы можно ознакомиться в приложении 1.

Заключение

В процессе выполнения практической работы были изучены теоретические основы построения трехмерного изображения с помощью двухмерных примитивов библиотеки OpenGL, выбрана среда программирования, выполнена программная реализация построения трехмерной фигуры, вследствие чего были получены навыки моделирования трехмерных объектов при помощи 2D примитивов библиотеки OpenGL.

Приложения

Приложение 1.

Листинг 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 Tao.OpenGl;
using Tao.FreeGlut;
using Tao.Platform.Windows;

namespace prog
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            canvas.InitializeContexts();
            Gl.glViewport(0, 0, canvas.Width, canvas.Height);
            Gl.glClearColor(1f, 1f, 1f, 1);
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Gl.glEnable(Gl.GL_DEPTH_TEST);
            Gl.glMatrixMode(Gl.GL_PROJECTION);
            Gl.glLoadIdentity();
            Gl.glFrustum(-1f, 1f, -1f, 1f, 3f, 10f);
            Gl.glMatrixMode(Gl.GL_MODELVIEW);
            Gl.glLoadIdentity();
            Gl.glTranslatef(0f, 0f, -8.0f);
            Gl.glRotatef(30, 1, 0, 0);
        }
        float[,] cube =
{
            {1f, 1.7f, 1f},
            {-1f, 1.7f, 1f},
            {-1f, 1.7f, -1f},
            {1f, 1.7f, -1f},

            {1f, -0.5f, 1f},
            {-1f, -0.5f, 1f},
            {-1f, -0.5f, -1f},
            {1f, -0.5f, -1f}
        };

        float[,] pyramid =
        {
            {0, -0.3f, -1f},
            {1f, -0.3f, -0.3f},
            {1f, -0.3f, 0.3f},
```

```
            {0, -0.3f, 1f},
            {-1f, -0.3f, 0.3f},
            {-1f, -0.3f, -0.3f},

            {0, 1.5f, 0}
        };

        float yAngle = 0;

        private void stripToolStripMenuItem_Click(object sender, EventArgs e)
        {

            DrawFigure();
        }

        private void canvas_KeyDown(object sender, KeyEventArgs e)
        {
            if(e.KeyCode == Keys.R)
            {
                timer2.Enabled = !timer2.Enabled;
            }
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            yAngle = (yAngle + 5) % 360;
            DrawFigure();
        }

        private void polygonToolStripMenuItem_Click(object sender, EventArgs
e)
        {

        }

        public void DrawFigure()
        {
            Gl.glClearColor(1f, 1f, 1f, 1);
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
            Gl.glClear(Gl.GL_DEPTH_BUFFER_BIT);

            Gl.glPushMatrix();

            Gl.glRotatef(yAngle, 0, 1, 0);

            //крышка куба
            Gl.glColor3f(0.6f, 1, 0);
            Gl.glBegin(Gl.GL_LINE_LOOP);
                for (int i = 0; i < 4; i++)
                {
                    Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
                }
            Gl.glEnd();

            // дно куба
            Gl.glBegin(Gl.GL_LINE_LOOP);
                for (int i = 4; i < 8; i++)
                {
                    Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
                }
            Gl.glEnd();

            // боковые рёбра куба
```

```
            for (int i = 0; i < 4; i++)
            {
                Gl.glBegin(Gl.GL_LINE_STRIP);
                    Gl.glVertex3f(cube[i, 0], cube[i, 1], cube[i, 2]);
                    Gl.glVertex3f(cube[i+4, 0], cube[i+4, 1], cube[i+4, 2]);
                Gl.glEnd();
            }

            //дно пирамиды
            Gl.glColor3f(0.6f, 0.1f, 0.5f);
            Gl.glBegin(Gl.GL_POLYGON);
                for (int i = 0; i < 6; i++)
                {
                    Gl.glVertex3f(pyramid[i, 0], pyramid[i, 1], pyramid[i,
2]);
                }
            Gl.glEnd();


            //бока пирамиды
            double r = 0.6, g = 0.1, b = 0.5;

            Gl.glBegin(Gl.GL_TRIANGLES);
                Gl.glColor3d(r, g, b);
                Gl.glVertex3f(pyramid[0, 0], pyramid[0, 1], pyramid[0, 2]);
                Gl.glVertex3f(pyramid[1, 0], pyramid[1, 1], pyramid[1, 2]);
                Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
            Gl.glEnd();

            r = 0.3;
            g = 0.6;
            b = 0;
            Gl.glBegin(Gl.GL_TRIANGLES);
                Gl.glColor3d(r, g, b);
                Gl.glVertex3f(pyramid[2, 0], pyramid[2, 1], pyramid[2, 2]);
                Gl.glVertex3f(pyramid[1, 0], pyramid[1, 1], pyramid[1, 2]);
                Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
            Gl.glEnd();

            r = 0.8;
            g = 0.1;
            b = 0.5;
            Gl.glBegin(Gl.GL_TRIANGLES);
                Gl.glColor3d(r, g, b);
                Gl.glVertex3f(pyramid[2, 0], pyramid[2, 1], pyramid[2, 2]);
                Gl.glVertex3f(pyramid[3, 0], pyramid[3, 1], pyramid[3, 2]);
                Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
            Gl.glEnd();


            r = 0.2;
            g = 0.7;
            b = 0.9;
            Gl.glBegin(Gl.GL_TRIANGLES);
                Gl.glColor3d(r, g, b);
                Gl.glVertex3f(pyramid[3, 0], pyramid[3, 1], pyramid[3, 2]);
                Gl.glVertex3f(pyramid[4, 0], pyramid[4, 1], pyramid[4, 2]);
                Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
            Gl.glEnd();

            r = 0.4;
            g = 0.5;
            b = 0.6;
```

```
                Gl.glBegin(Gl.GL_TRIANGLES);
                    Gl.glColor3d(r, g, b);
                    Gl.glVertex3f(pyramid[4, 0], pyramid[4, 1], pyramid[4, 2]);
                    Gl.glVertex3f(pyramid[5, 0], pyramid[5, 1], pyramid[5, 2]);
                    Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
                Gl.glEnd();

                r = 0.5;
                g = 0.3;
                b = 0.2;
                Gl.glBegin(Gl.GL_TRIANGLES);
                    Gl.glColor3d(r, g, b);
                    Gl.glVertex3f(pyramid[0, 0], pyramid[0, 1], pyramid[0, 2]);
                    Gl.glVertex3f(pyramid[5, 0], pyramid[5, 1], pyramid[5, 2]);
                    Gl.glVertex3f(pyramid[6, 0], pyramid[6, 1], pyramid[6, 2]);
                Gl.glEnd();


                Gl.glPopMatrix();

                canvas.Invalidate();
            }
        }
    }
```