Московский авиационный институт

(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Компьютерная графика»

Лабораторная работа № 3

Тема: Основы построения фотореалистичных изображений

Студент: Пивницкий Даниэль Сергеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Используя результаты Л.Р.2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

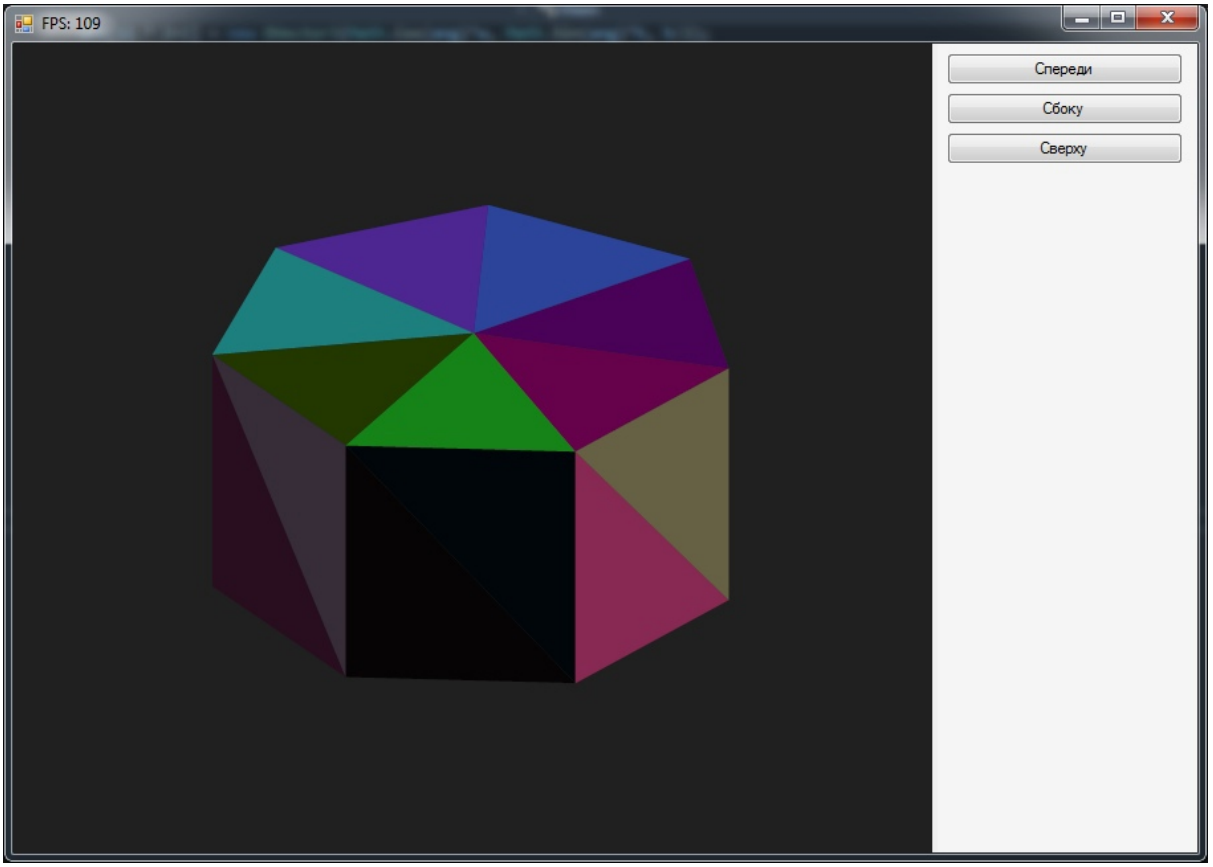
2. **Описание программы**

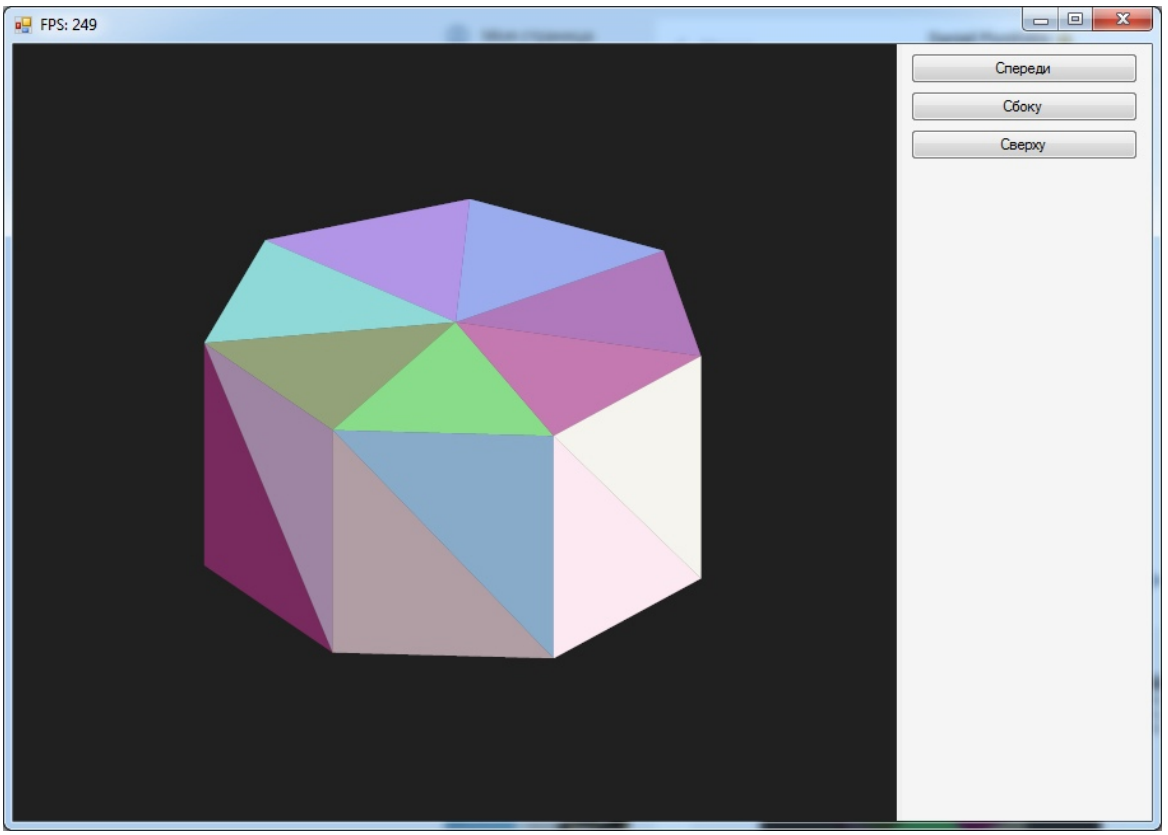
Код по своей сути ровно тоже самое, что в Л.Р.2, только теперь мы вершины не в ручную ставим, и вместо линий рисуем Graphics.FillPolygon. Есть новый глобальный вектор Sun в зависимости от угла между солнцем и нормалью, полигон либо затемняется, либо засветляется.

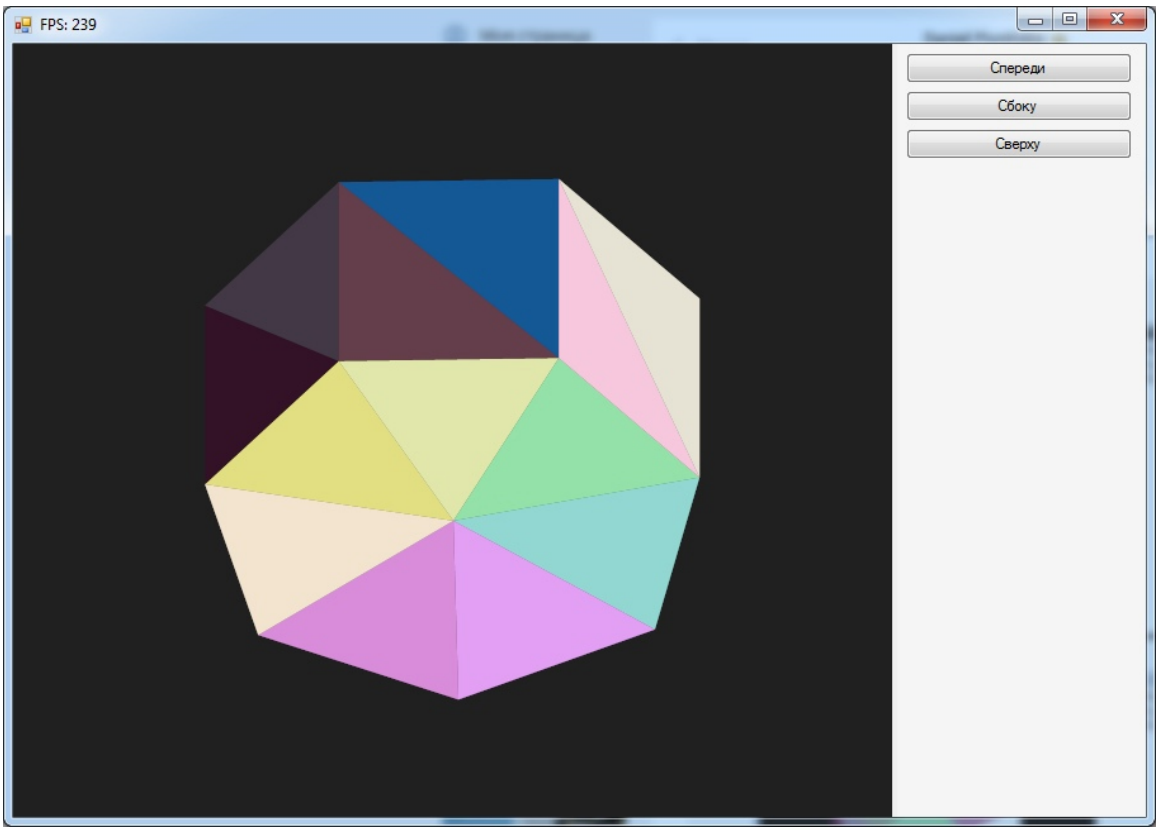
3. **Набор тестов**

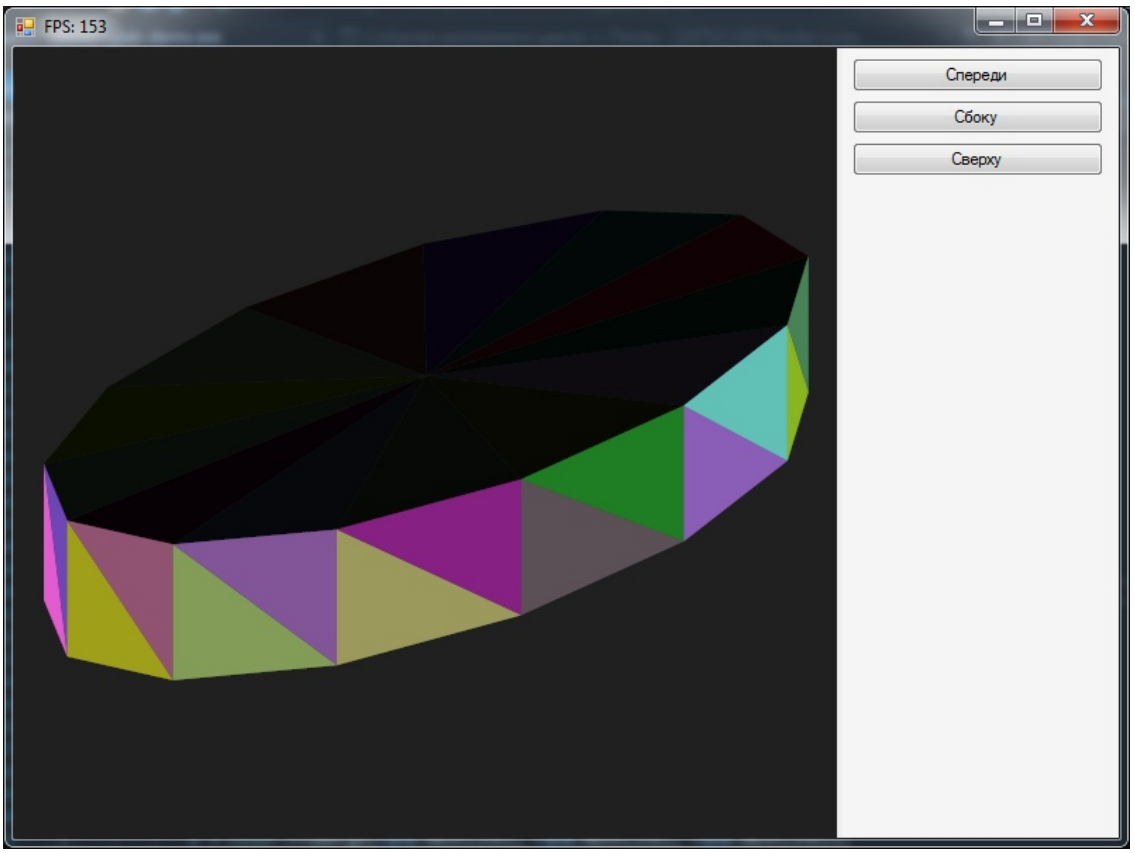
Фигуры под разными углами и освещением

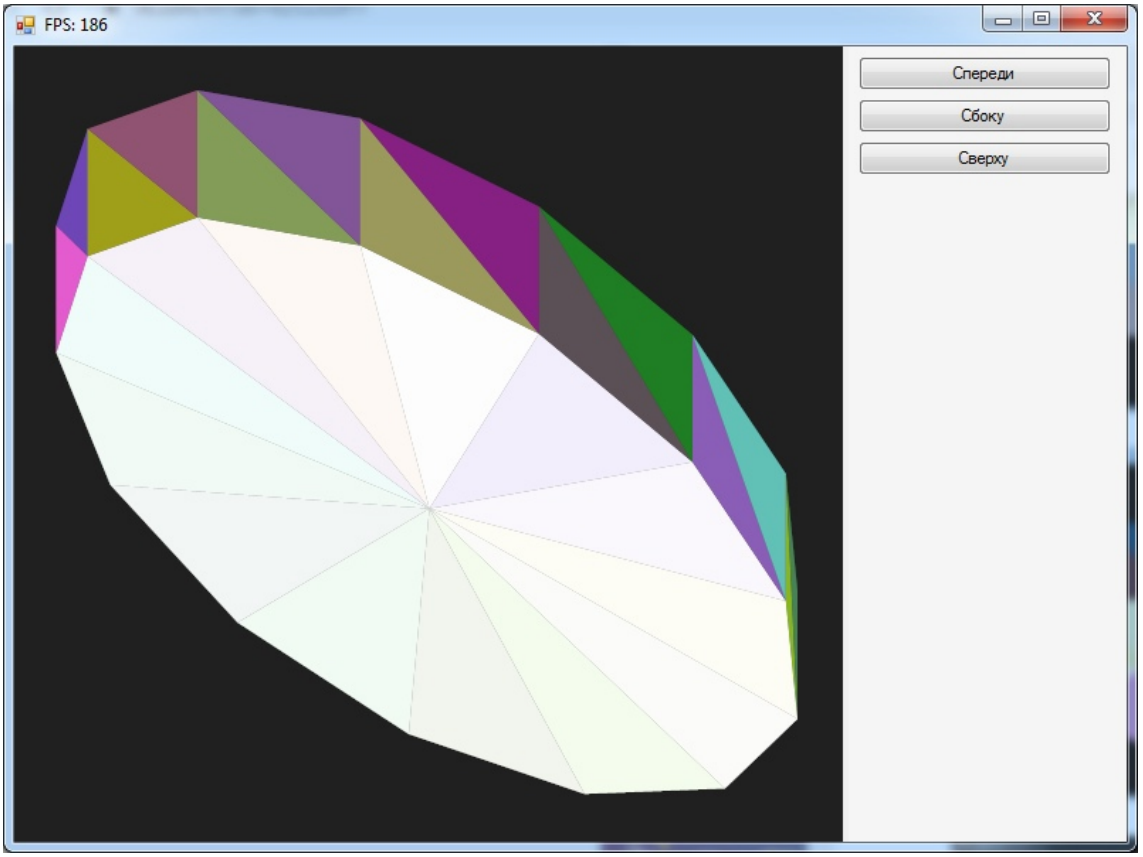
4. Результаты выполнения тестов




FPS: 249
Спереди
Сбоку
Сверху


FPS: 239
Спереди
Сбоку
Сверху


FPS: 153
Спереди
Сбоку
Сверху


FPS: 186
Спереди
Сбоку
Сверху

5. Листинг программы

```
using System;
using System.Linq;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Forms;
using System.ComponentModel;
using System.Collections.Generic;
using CGLabPlatform;

public class Polygon
{
    DVector3[] vertex;
    DVector3 normal;
    Color edge;
    public Polygon(Color inp)
    {
        vertex = new DVector3[3];
        edge = inp;
    }
    public void Set(DVector3 inpu, int i)
    {
        if ((i <= 2) && (i >= 0))
        {
            vertex[i] = inpu;
        }
    }

    public void Sett(DVector3 inpu1, DVector3 inpu2,
DVector3 inpu3)
    {
        vertex[0] = inpu1;
        vertex[1] = inpu2;
        vertex[2] = inpu3;
```

```
    }
    public void CalcNormal(bool invert)
    {
          normal = DVector3.CrossProduct(vertex[1] -
vertex[0], vertex[2] - vertex[0]);
        if (normal.GetLengthSquared() == 0)
        {
            return;
        }
        normal.Normalize();
         if (DVector3.DotProduct(normal, vertex[1] +
vertex[0] + vertex[2]) <0)//(invert)
        {
            normal *= -1;
        }
    }
     DVector3 ApplyTransform(DVector3 inp, DVector2
ang)
    {
        DVector3 rez = new DVector3(0, 0, 0), tmp;
        rez = inp;
            rez.X = inp.X * Math.Cos(ang.X / 180 *
Math.PI) - inp.Y * Math.Sin(ang.X / 180 * Math.PI);
            rez.Z = inp.Y * Math.Cos(ang.X / 180 *
Math.PI) + inp.X * Math.Sin(ang.X / 180 * Math.PI);
        rez.Y = -inp.Z;
        tmp.X = rez.X;
            tmp.Z = rez.Z * Math.Cos(ang.Y / 180 *
Math.PI) + rez.Y * Math.Sin(ang.Y / 180 * Math.PI);
            tmp.Y = rez.Y * Math.Cos(ang.Y / 180 *
Math.PI) - rez.Z * Math.Sin(ang.Y / 180 * Math.PI);


        return tmp;
    }
```

```
    DVector2 to2d(DVector3 inp)
    {
        DVector2 rez = new DVector2(inp.X, inp.Y);
        rez *= 100;// 0 / (10 + inp.Z);
        return rez;
    }

    DVector3 camVector(DVector2 inp)
    {
         DVector3 rez = new DVector3(0, 0, 0), temp =
new DVector3(0, 0, 0);

        temp.Z = Math.Sin(inp.Y / 180 * Math.PI);
        temp.Y = Math.Cos(inp.Y / 180 * Math.PI);
        temp.X = 0;

            rez.X = temp.X * Math.Cos(inp.X / 180 *
Math.PI) + temp.Y * Math.Sin(inp.X / 180 * Math.PI);
            rez.Y = temp.Y * Math.Cos(inp.X / 180 *
Math.PI) - temp.X * Math.Sin(inp.X / 180 * Math.PI);
        rez.Z = -temp.Z;
        return rez;
    }

     public void draw(DVector2 camr, DVector2 sunn,
DVector2 offs, GDIDeviceUpdateArgs e)
    {
        if (normal.GetLengthSquared() == 0)
        {
            return;
        }
        DVector3 cam = camVector(camr);
        DVector3 soon = camVector(sunn);
        DVector2[] tvertex = new DVector2[3];
```

```
        for (int i = 0; i < 3; i++)
        {
                                        tvertex[i]   =
to2d(ApplyTransform(vertex[i], camr))+offs;
        }

        if (DVector3.DotProduct(cam, normal) > 0)
        {//backfacing

        }
        else
        {
            PointF[] ttvertex = new PointF[3];
            for(int j = 0; j < 3; j++)
            {
                ttvertex[j].X = (float)tvertex[j].X;
                ttvertex[j].Y = (float)tvertex[j].Y;
            }
               double brg = DVector3.DotProduct(soon,
normal);
            Color shade;
            if (brg < 0)
            {
                brg += 1;
                shade = Color.FromArgb((int)(edge.R *
(brg)),   (int)(edge.G   *   (brg)),   (int)(edge.B   *
(brg)));
            }
            else
            {
                brg = 1 - brg;
                 double rr = 255 - edge.R; rr *= brg;
rr = 255 - rr;
                 double gg = 255 - edge.G; gg *= brg;
gg = 255 - gg;
```

```
                double bb = 255 - edge.B; bb *= brg;
bb = 255 - bb;
                    shade = Color.FromArgb((int)rr,
(int)gg, (int)bb);
            }
                        SolidBrush broosh = new
SolidBrush(shade);
            e.Graphics.FillPolygon(broosh, ttvertex);
             //e.Surface.DrawLine(edge.ToArgb(), offs
+ tvertex[0], offs + tvertex[1]);
             //e.Surface.DrawLine(edge.ToArgb(), offs
+ tvertex[0], offs + tvertex[2]);
             //e.Surface.DrawLine(edge.ToArgb(), offs
+ tvertex[2], offs + tvertex[1]);
        }
    }
}
//Сектор эллипсоида
//Прямой эллиптический цилиндр.

public class Kleen
{
    public int nuu;
    public Polygon[] polys;
    public Kleen(int nu,double w,double d, double h)
    {
        double ang;
        DVector3[] vrt = new DVector3[nu*2+2];
        vrt[nu*2] = new DVector3(0, 0, -h/2);
        vrt[nu*2+1] = new DVector3(0, 0, h/2);
        for(int ii = 0; ii < nu; ii++)
        {
            ang = (double)ii;
            ang /= (double)nu;
            ang*= Math.PI * 2;
```

```
                                vrt[ii * 2] = new
DVector3(Math.Cos(ang)*w, Math.Sin(ang)*d, -h/2);
                                vrt[ii * 2+1] = new
DVector3(Math.Cos(ang)*w, Math.Sin(ang)*d, h/2);
        }
        nuu = nu*4;
        polys = new Polygon[nuu];
        var rand = new Random();
        Color c;
        for (int ii = 0; ii < nu ; ii++)
        {
                c = Color.FromArgb(rand.Next(256),
rand.Next(256), rand.Next(256));
            polys[ii*4] = new Polygon(c);
                c = Color.FromArgb(rand.Next(256),
rand.Next(256), rand.Next(256));
            polys[ii*4+1] = new Polygon(c);
                c = Color.FromArgb(rand.Next(256),
rand.Next(256), rand.Next(256));
            polys[ii * 4+2] = new Polygon(c);
                c = Color.FromArgb(rand.Next(256),
rand.Next(256), rand.Next(256));
            polys[ii * 4 + 3] = new Polygon(c);

              polys[ii * 4].Sett(vrt[nu*2], vrt[ii *
2], vrt[(ii * 2 + 2) % (nu*2)]);

                polys[ii * 4+1].Sett(vrt[nu * 2+1],
vrt[(ii * 2 + 1) % (nu * 2)], vrt[(ii * 2 + 3) % (nu
* 2)]);

            polys[ii * 4 + 2].Sett(vrt[(ii * 2 + 1) %
(nu * 2)], vrt[(ii * 2) % (nu * 2)], vrt[(ii * 2 + 2)
% (nu * 2)]);
```

```
            polys[ii * 4 + 3].Sett(vrt[(ii * 2 + 1) %
(nu * 2)], vrt[(ii * 2+3) % (nu * 2)], vrt[(ii * 2 +
2) % (nu * 2)]);

            polys[ii * 4].CalcNormal(true);
            polys[ii * 4+1].CalcNormal(false);
            polys[ii * 4+2].CalcNormal(true);
            polys[ii * 4+3].CalcNormal(false);

        }

    }
}
public      abstract      class      CGLab01      :
GFXApplicationTemplate<CGLab01>
{
    #region Инициализация

    [STAThread]
    static void Main()
    {

        RunApplication();
    }

    public abstract DVector2 FirstPoint { get; set; }
    public abstract DVector2 Sun { get; set; }

    public Kleen k;

     protected override void OnMainWindowLoad(object
sender, EventArgs args)
    {
        base.RenderDevice.BufferBackCol = 0x20;
        base.RenderDevice.MouseMoveWithLeftBtnDown +=
```

```
(s, e) =>
        {
              FirstPoint += new DVector2(e.MovDeltaX,
e.MovDeltaY);
             if (FirstPoint.X > 360) { FirstPoint -=
new DVector2(360, 0); }
            if (FirstPoint.X < 0) { FirstPoint += new
DVector2(360, 0); }
            if (FirstPoint.Y > 90) { FirstPoint = new
DVector2(FirstPoint.X, 90); }
              if (FirstPoint.Y < -90) { FirstPoint =
new DVector2(FirstPoint.X, -90); ; }
        };

          base.RenderDevice.MouseMoveWithRightBtnDown
+= (s, e) =>
        {
                    Sun += new DVector2(e.MovDeltaX,
e.MovDeltaY);
                    if (Sun.X > 360) { Sun -= new
DVector2(360, 0); }
            if (Sun.X < 0) { Sun += new DVector2(360,
0); }
                     if (Sun.Y > 90) { Sun = new
DVector2(Sun.X, 90); }
                    if (Sun.Y < -90) { Sun = new
DVector2(Sun.X, -90); ; }
        };

        MainWindow.Shown += (s, e) =>
        {
                 var btnX = new Button() { Text =
"Спереди" };
            btnX.Click += (cs, ce) => {
                FirstPoint = new DVector2(0, 0);
```

```
            };
            AddControll(btnX, 30);
             var btnY = new Button() { Text = "Сбоку"
};
            btnY.Click += (cs, ce) => {
                FirstPoint = new DVector2(90, 0);
            };
            AddControll(btnY, 30);
            var btnZ = new Button() { Text = "Сверху"
};
            btnZ.Click += (cs, ce) => {
                FirstPoint = new DVector2(0, 90);
            };
            AddControll(btnZ, 30);
        };




    }


    #endregion

     protected override void OnDeviceUpdate(object s,
GDIDeviceUpdateArgs e)
    {//do stuff here
        if (k == null)
        {
            k = new Kleen(13,2,5,1);
        }
          DVector2 Center = new DVector2(e.Width / 2,
e.Heigh / 2);
        for (int i = 0; i < k.nuu; i++)
        {
```

```
            k.polys[i].draw(FirstPoint, Sun, Center,
e);
        }
    }


    private void AddControll(Control ctrl, int heigh)
    {
                                    var    layout    =
ValueStorage.Controls[0].Controls[0]              as
TableLayoutPanel;

        layout.SuspendLayout();
        ctrl.Dock = DockStyle.Fill;
        layout.Parent.Height += heigh;
         layout.RowStyles.Insert(layout.RowCount - 1,
new RowStyle(SizeType.Absolute, heigh));
         layout.Controls.Add(ctrl, 0, layout.RowCount
- 1);
        layout.SetColumnSpan(ctrl, 2);
        layout.RowCount++;
        layout.ResumeLayout(true);
    }

    private void AddControl(Control ctrl, int heigh,
string InsertBeforeProperty)
    {
                                    var    layout    =
ValueStorage.Controls[0].Controls[0]              as
TableLayoutPanel;
        layout.SuspendLayout();
        ctrl.Dock = DockStyle.Fill;
        layout.Parent.Height += heigh;
                                var    beforectrl    =
ValueStorage.GetControlForProperty(InsertBeforeProper
```

```
ty);
                                    var    position    =
layout.GetPositionFromControl(beforectrl).Row + 1;
          for (int r = layout.RowCount; position <=
r--;)
        {
               for (int c = layout.ColumnCount; 0 !=
c--;)
            {
                                        var   control   =
layout.GetControlFromPosition(c, r);
                                  if  (control  !=  null)
layout.SetRow(control, r + 1);
            }
        }
           layout.RowStyles.Insert(position - 1, new
RowStyle(SizeType.Absolute, heigh));
        layout.Controls.Add(ctrl, 0, position - 1);
        layout.SetColumnSpan(ctrl, 2);
        layout.RowCount++;
        layout.ResumeLayout(true);

    }

}
```

6. Литература

1. Документация Graphics.FillPolygon: https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.graphics.fillpolygon?view=dotnet-plat-ext-6.0
2. Документация Color argb: https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.color.fromargb?view=net-6.0