Московский авиационный институт

(Национальный исследовательский университет)

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

Лабораторная работа № 2

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий

Студент: Пивницкий Даниэль Сергеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

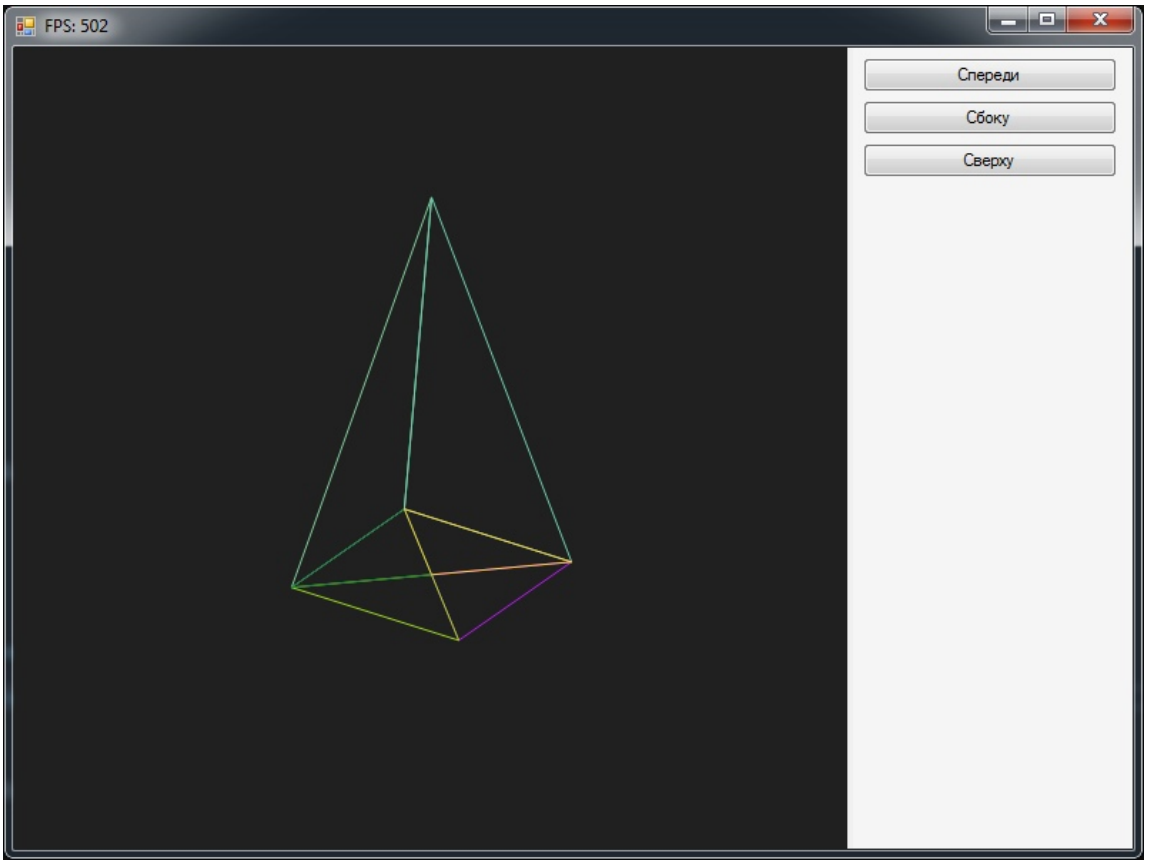
Вариант: 16

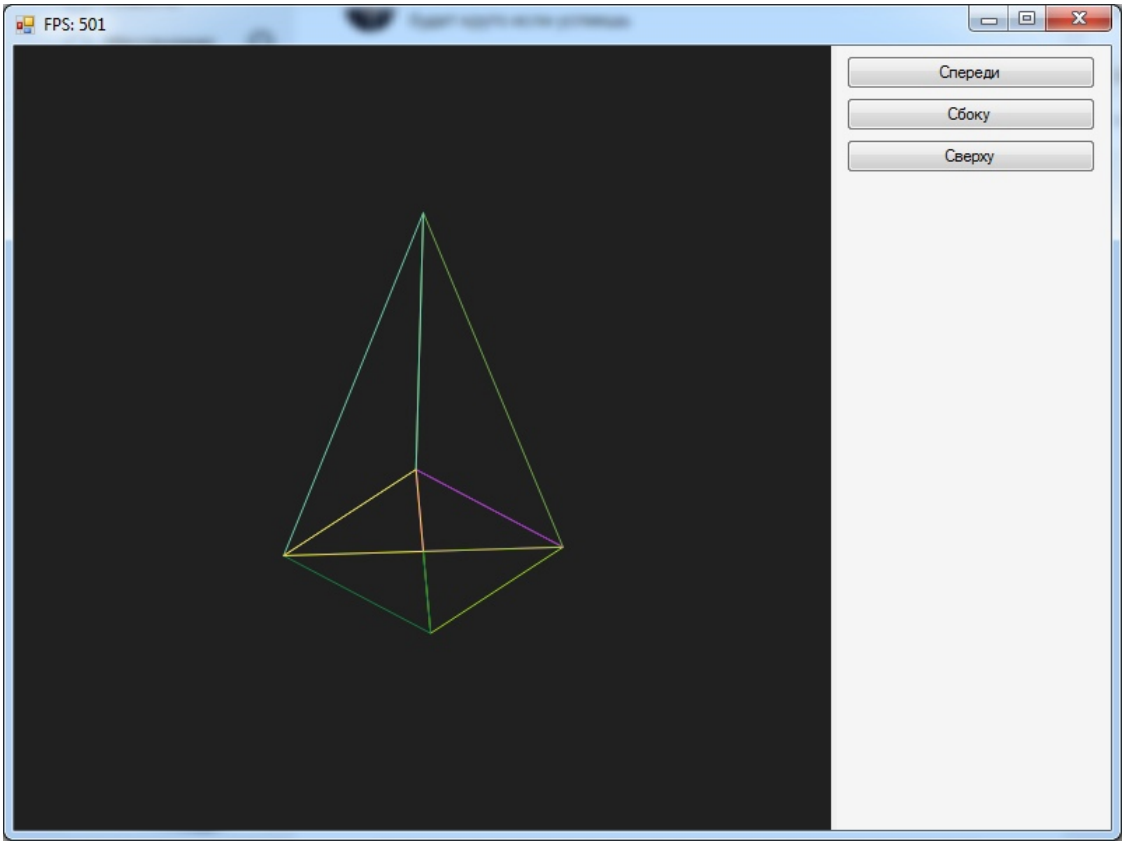
Гранная прямая правильная пирамида

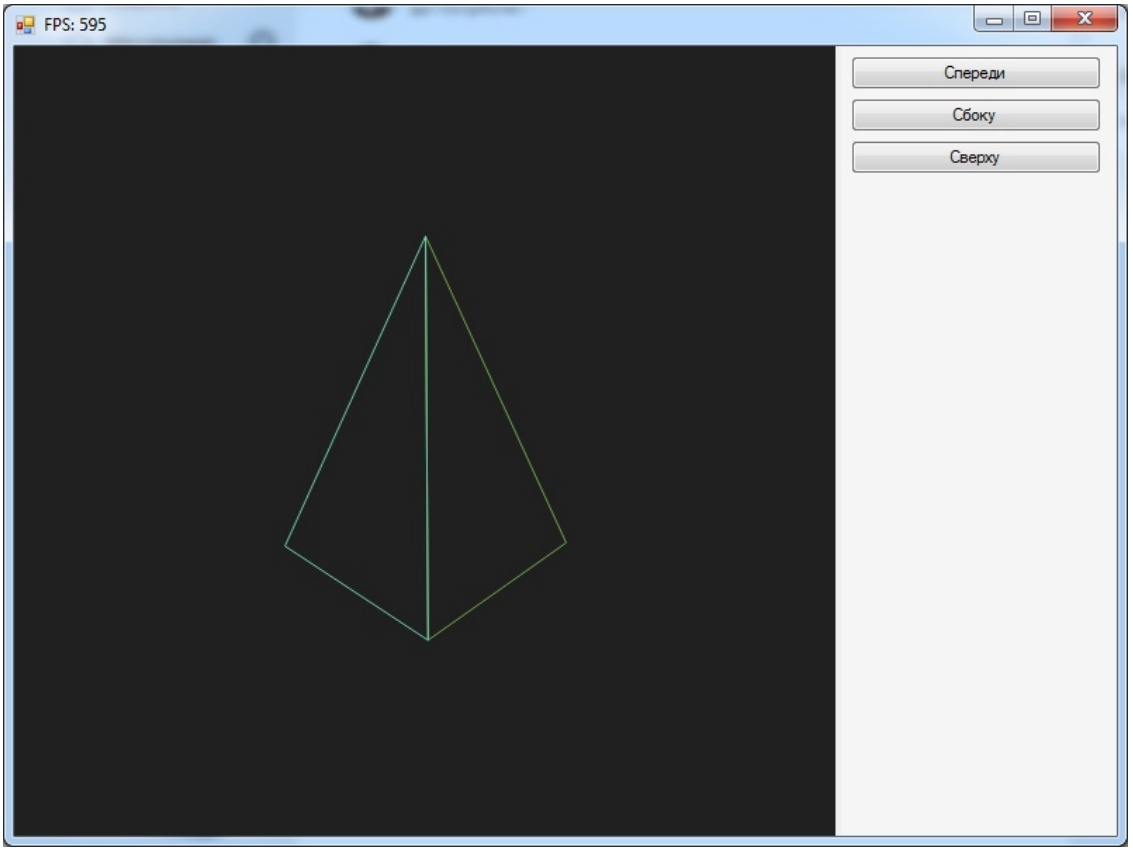
2. **Описание программы**

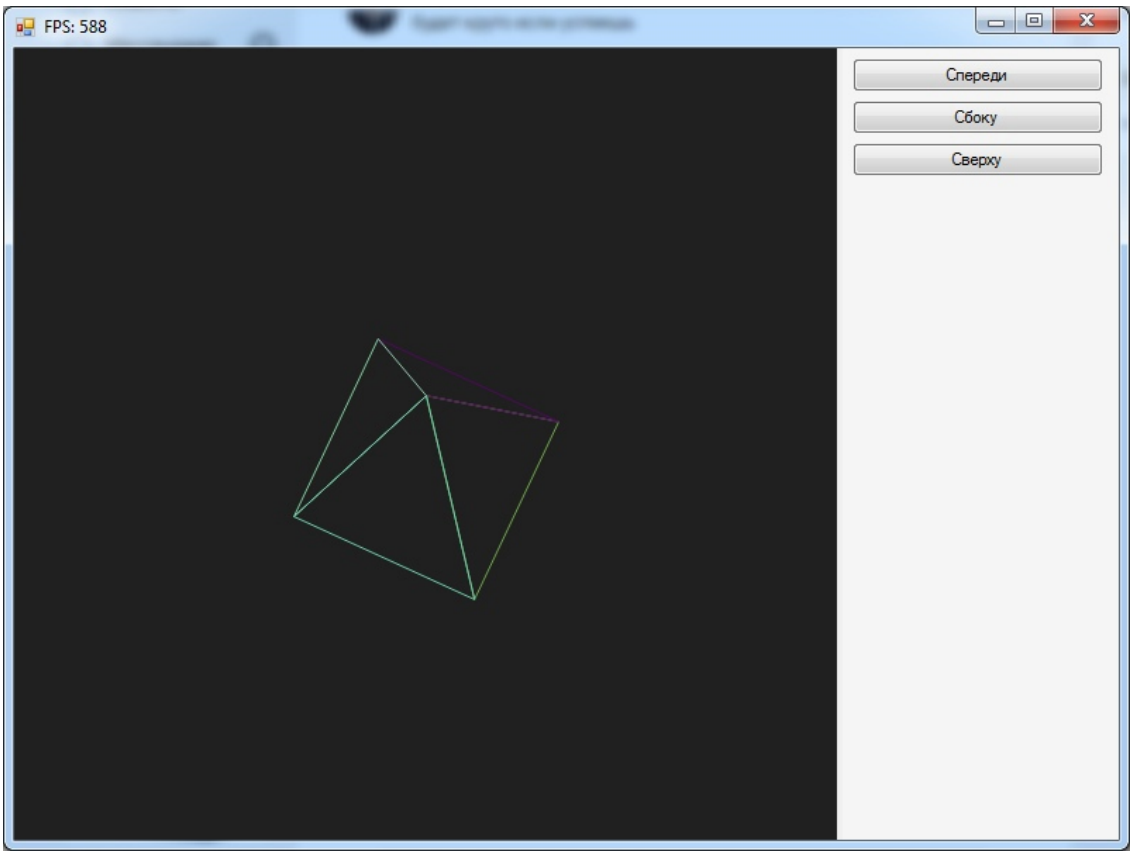
Основной частью программы является система поворота камеры. Камера вращается на определённом расстоянии от начала координат. Углы ее поворота используются для вращения вершин клина. Клин составлен из плоскостей, ограниченных треугольниками. У каждого треугольника вычисляется нормаль. С помощью скалярного произведения вектора нормали и вектора камеры определяется, стоит ли отображать данный треугольник. Нормаль вычисляется как векторное произведение двух сторон треугольников.

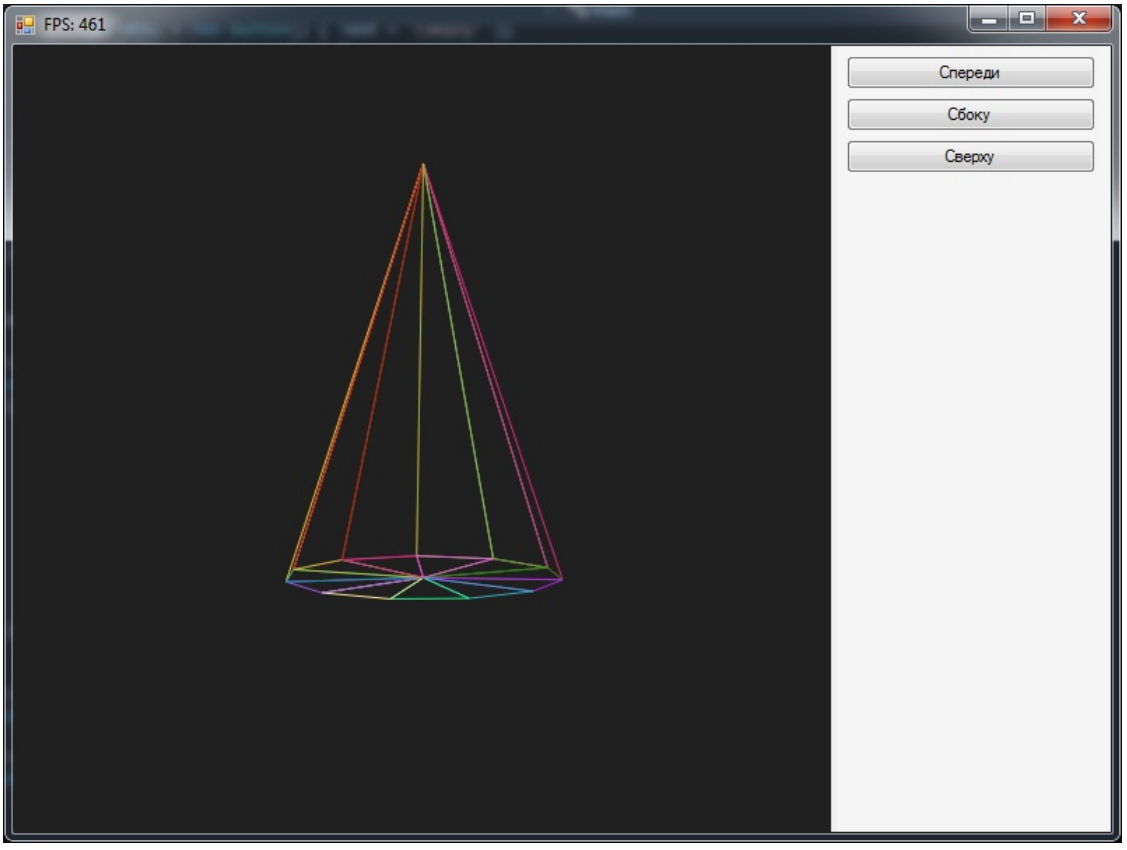
3. Набор тестов




FPS: 501
Спереди
Сбоку
Сверху


FPS: 595
Спереди
Сбоку
Сверху


FPS: 588
Спереди
Сбоку
Сверху


FPS: 461
Спереди
Сбоку
Сверху

4. Результаты выполнения тестов

Поскольку все поверхности обладают разными цветами, хорошо видно, что в зависимости от угла поворота, некоторые треугольники перестают отрисовываться. Ортогональные проекции получаются с помощью кнопок справа.

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 CalcNormal(bool invert)
    {
```

```
          normal = DVector3.CrossProduct(vertex[1] -
vertex[0], vertex[2] - vertex[0]);
        normal.Normalize();
        if (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 offs,
GDIDeviceUpdateArgs e)
    {
        DVector3 cam = camVector(camr);
        DVector2[] tvertex = new DVector2[3];
        for (int i = 0; i < 3; i++)
        {
                                       tvertex[i]   =
to2d(ApplyTransform(vertex[i], camr));
        }
        if (DVector3.DotProduct(cam, normal) > 0)
        {

        }
        else
        {
             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]);
        }
    }
}

public class Kleen
{
    public Polygon[] polys;
    public Kleen()
    {
        int nu = 4;
        double ang;
        DVector3[] vrt = new DVector3[nu+2];
        vrt[nu] = new DVector3(0, 0, 2);
        vrt[nu+1] = new DVector3(0, 0, -1);
        for(int ii = 0; ii < nu; ii++)
        {
            ang = (double)ii;
            ang /= (double)nu;
            ang*= Math.PI * 2;
               vrt[ii] = new DVector3(Math.Cos(ang),
Math.Sin(ang),-1);
        }

        polys = new Polygon[nu*2];
        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*2] = new Polygon(c);
```

```
                c = Color.FromArgb(rand.Next(256),
rand.Next(256), rand.Next(256));
            polys[ii*2+1] = new Polygon(c);
            polys[ii*2].Set(vrt[nu], 0);
            polys[ii*2].Set(vrt[ii], 1);
            polys[ii*2].Set(vrt[(ii+1) % nu], 2);

            polys[2*ii+1].Set(vrt[nu+1], 0);
            polys[2*ii+1].Set(vrt[ii], 1);
               polys[2*ii+1].Set(vrt[(ii + 1) % nu],
2);

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

        }

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

    [STAThread]
    static void Main()
    {

        RunApplication();
    }

     public abstract DVector2 FirstPoint { 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); ; }
        };

        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)
    {
        if (k == null)
        {
            k = new Kleen();
        }
         DVector2 Center = new DVector2(e.Width / 2,
e.Heigh / 2);
        for (int i = 0; i < 8; i++)
        {
            k.polys[i].draw(FirstPoint, 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(InsertBeforePrope
rty);
                                    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 Random class:
http://dir.by/developer/csharp/random/
2. Документация Color argb:
https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.color.fromargb?view=net-6.0