```
namespace WindowsFormsApp2
{
    public partial class Form1 : Form
    {
        IntPtr Handle3D;
        IntPtr HDC3D;
        IntPtr HRC3D;

        //контролируют область камеры
        float r = 20;
        float fi = 35;
        float psi = 35;

        //контролирует движение
        float dt = 0;

        //угловые скорости
        float wearth = 1f;
        float wmoon = 5f;

        //регулирует прозрачность
        float[] vis = new float[] { 1f, 1f, 1f };


        uint Texture;
        uint Texture2;
        uint Texture3;

        //шрифт
        int Font3D = 0;

        float[] orbita = new float[] { 4, 1, 0 };
        float[] radius = new float[] { 1, 0.6f, 0.2f };
        float[] meshfi = new float[] { 32, 24, 20 };
        float[] meshtet = new float[] { 32, 24, 20 };

        // Конструктор
        public Form1()
        {
            InitializeComponent();
            // Для рисования выбираем форму
            Handle3D = Handle;
            HDC3D = User.GetDC(Handle3D);
            Gdi.PIXELFORMATDESCRIPTOR PFD = new Gdi.PIXELFORMATDESCRIPTOR();
            PFD.nVersion = 1;
            PFD.nSize = (short)Marshal.SizeOf(PFD);
            PFD.dwFlags = Gdi.PFD_DRAW_TO_WINDOW | Gdi.PFD_SUPPORT_OPENGL |
Gdi.PFD_DOUBLEBUFFER;
            PFD.iPixelType = Gdi.PFD_TYPE_RGBA;
            PFD.cColorBits = 24;
            PFD.cDepthBits = 32;
            PFD.iLayerType = Gdi.PFD_MAIN_PLANE;

            int nPixelFormat = Gdi.ChoosePixelFormat(HDC3D, ref PFD);
            Gdi.SetPixelFormat(HDC3D, nPixelFormat, ref PFD);
            HRC3D = Wgl.wglCreateContext(HDC3D);
            Wgl.wglMakeCurrent(HDC3D, HRC3D);
            Form1_Resize(null, null);


            Gl.glEnable(Gl.GL_DEPTH_TEST);
            //натягивание текстуры
            Texture = LoadTexture("Sun1.bmp");
            Texture2 = LoadTexture("Earth1.bmp");
            Texture3 = LoadTexture("moon.bmp");
```

```
            CreateFont3D(Font);

            //настройки таймера
            timer1.Interval = 100; // миллисекунды
            timer1.Enabled = true;
            timer1.Tick += timer1_Tick;

            //всплывающие подсказки
            System.Windows.Forms.ToolTip btt = new System.Windows.Forms.ToolTip();
            btt.ToolTipTitle = "Значение угловой скорости Земли";
            btt.UseAnimation = true;
            btt.IsBalloon = true;
            btt.ShowAlways = true;
            btt.AutoPopDelay = 3000;
            btt.InitialDelay = 500;
            btt.ReshowDelay = 500;
            btt.SetToolTip(trackBar_wearth, "Изменяется в диапазоне [ -10; 10 ],
знак задает направление вращения.");

            System.Windows.Forms.ToolTip btt1 = new System.Windows.Forms.ToolTip();
            btt1.ToolTipTitle = "Значение угловой скорости Луны";
            btt1.UseAnimation = true;
            btt1.IsBalloon = true;
            btt1.ShowAlways = true;
            btt1.AutoPopDelay = 3000;
            btt1.InitialDelay = 500;
            btt1.ReshowDelay = 500;
            btt1.SetToolTip(trackBar_wmoon, "Изменяется в диапазоне [ -15; 15 ],
знак задает направление вращения.");

            System.Windows.Forms.ToolTip btt2 = new System.Windows.Forms.ToolTip();
            btt2.ToolTipTitle = "Замечание";
            btt2.UseAnimation = true;
            btt2.IsBalloon = true;
            btt2.ShowAlways = true;
            btt2.AutoPopDelay = 3000;
            btt2.InitialDelay = 500;
            btt2.ReshowDelay = 500;
            btt2.SetToolTip(checkBox3, "Нельзя снять флажок, если включена
текстура!");

            textBoxEarth.Text = orbita[0].ToString();
            textBoxMoon.Text = orbita[1].ToString();

            textBox_meshS_fi.Text = meshfi[0].ToString();
            textBox_meshE_fi.Text = meshfi[1].ToString();
            textBox_meshM_fi.Text = meshfi[2].ToString();

            textBox_meshS_tet.Text = meshtet[0].ToString();
            textBox_meshE_tet.Text = meshtet[1].ToString();
            textBox_meshM_tet.Text = meshtet[2].ToString();

            textBox_wE.Text = wearth.ToString();
            textBox_wM.Text = wmoon.ToString();
        }



        //определяет область вывода
        private void Form1_Resize(object sender, EventArgs e)
        {
            Gl.glMatrixMode(Gl.GL_PROJECTION);
            Gl.glLoadIdentity();
```

```
            int w = ClientRectangle.Width - panel1.Width;
            int h = ClientRectangle.Height;
            Glu.gluPerspective(30, (double)w / h, 2, 20000);
            //область вывода
            Gl.glViewport(0, 0, w, h);
        }



        //вызов отрисовки
        private void Form1_Paint(object sender, PaintEventArgs e)
        {

            //цвет фона
            Gl.glClearColor(0, 0, 0, 1);
            //очистка буферов
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);

            //начальное положение камеры
            Gl.glMatrixMode(Gl.GL_MODELVIEW);
            Gl.glLoadIdentity();
            Gl.glTranslatef(0, 0, -r);
            Gl.glRotatef(-90, 0, 1f, 0);
            Gl.glRotatef(-90, 1f, 0, 0);
            Gl.glRotatef(fi, 0, 1f, 0);//широта
            Gl.glRotatef(psi, 0, 0, 1f);//долгота

            //Оси
            DrawScene();

            //Планеты
            DrawingOrder();

            Gl.glFinish();
            Gdi.SwapBuffers(HDC3D);
        }

    //Рисование осей
        void DrawScene()
        {

            Gl.glColor3f(0, 0, 0);
            Gl.glPointSize(15);
            Gl.glEnable(Gl.GL_POINT_SMOOTH);

            Gl.glBegin(Gl.GL_POINTS);
            Gl.glVertex3f(0, 0, 0);
            Gl.glEnd();

            Gl.glBegin(Gl.GL_LINES);
            Gl.glColor3f(1, 0, 0);
            Gl.glVertex3f(0, 0, 0);
            Gl.glVertex3f(2, 0, 0);
            Gl.glColor3f(0, 1, 0);
            Gl.glVertex3f(0, 0, 0);
            Gl.glVertex3f(0, 2, 0);
            Gl.glColor3f(0, 0, 1);
            Gl.glVertex3f(0, 0, 0);
            Gl.glVertex3f(0, 0, 2);
            Gl.glEnd();
            Gl.glColor3f(1, 0, 0);

            //название осей
            OutText3D(2, 0, 0, "OX");
            Gl.glColor3f(0, 1, 0);
            OutText3D(0, 2, 0, "OY");
```

```
            Gl.glColor3f(0, 0, 1);
            OutText3D(0, 0, 2, "OZ");
        }

        void DrawSphere(float r, float nx, float ny)
        {
            int ix, iy;
            float x, y, z, siny, cosy, siny1, cosy1, sinx, cosx;


            Gl.glBegin(Gl.GL_QUAD_STRIP);
            for (iy = 0; iy < ny; iy++)
            {

                //угол тета [0;pi]
                siny = (float)Math.Sin(iy * (float)Math.PI / ny);
                cosy = (float)Math.Cos(iy * (float)Math.PI / ny);

                siny1 = (float)Math.Sin(iy * (float)Math.PI / ny + (float)Math.PI /
ny);
                cosy1 = (float)Math.Cos(iy * (float)Math.PI / ny + (float)Math.PI /
ny);


                for (ix = 0; ix <= nx; ix++)
                {
                    //угол фи [0;2pi]
                    sinx = (float)Math.Sin(2 * ix * (float)Math.PI / nx);
                    cosx = (float)Math.Cos(2 * ix * (float)Math.PI / nx);

                    x = r * siny * cosx;
                    y = r * siny * sinx;
                    z = r * cosy;
                    Gl.glNormal3f(x, y, z);
                    Gl.glTexCoord2f(-ix / nx, -iy / ny);
                    Gl.glVertex3f(x, y, z);

                    x = r * siny1 * cosx;
                    y = r * siny1 * sinx;
                    z = r * cosy1;
                    Gl.glNormal3f(x, y, z);
                    Gl.glTexCoord2f(-ix / nx, -(iy / ny + 1 / ny));
                    Gl.glVertex3f(x, y, z);
                }
            }
            Gl.glEnd();
        }

        void SpherePlanet(float r, float nx, float ny, int name)
        {
            Color[] colplan = new Color[] { Color.Gold, Color.RoyalBlue,
Color.LightSteelBlue };
            Color[] colgrid = new Color[] { Color.Goldenrod, Color.DarkBlue,
Color.Gray };
            Color[] colname = new Color[] { Color.DarkOrange, Color.Blue,
Color.LightSlateGray };
            Color[] colorb = new Color[] { Color.Purple, Color.ForestGreen,
Color.ForestGreen };
            //скорость вращение вокруг своей оси
            float[] rotspeed = new float[] { 100, 400, 400 };



            Gl.glColor3f((float)colname[name].R / 255, (float)colname[name].G / 255,
(float)colname[name].B / 255);
```

```
            switch (name)
            {
                case 0:
                    OutText3D(1f, 0.5f, 0f, "Sun");
                    break;
                case 1:
                    Gl.glTranslatef((float)Math.Cos(2 * 3.14159 * (dt * wearth + 30)
/ 100.0) * orbita[name - 1], (float)Math.Sin(2 * 3.14159 * (dt * wearth + 30) /
100.0) * orbita[name - 1], 0);
                    OutText3D(0.4f, 0.4f, 0.4f, "Earth");
                    break;
                case 2:
                    Gl.glTranslatef((float)Math.Cos(2 * 3.14159 * (dt * wearth + 30)
/ 100.0) * orbita[name - 2], (float)Math.Sin(2 * 3.14159 * (dt * wearth + 30) /
100.0) * orbita[name - 2], 0);
                    Gl.glTranslatef(-(float)Math.Cos(2 * 3.14159 * (dt * wmoon + 30)
/ 100.0) * orbita[name - 1], -(float)Math.Sin(2 * 3.14159 * (dt * wmoon + 30) /
100.0) * orbita[name - 1], 0);
                    OutText3D(0.2f, 0.1f, 0.0f, "Moon");
                    break;
            }


            //рисование орбит
            Gl.glColor3f((float)colorb[name].R / 255, (float)colorb[name].G / 255,
(float)colorb[name].B / 255);
            Gl.glPointSize(1f);
            Gl.glBegin(Gl.GL_POINTS);
            for (int i = 0; i < 100; i = i + 2)
            {
                Gl.glVertex3f((float)Math.Cos(2 * 3.14159 * i / 100.0) *
orbita[name], (float)Math.Sin(2 * 3.14159 * i / 100.0) * orbita[name], 0);
            }
            Gl.glEnd();


            //сетка планет
            switch (checkBox1.CheckState)
            {
                case CheckState.Checked:
                    Gl.glColor3f((float)colgrid[name].R / 255,
(float)colgrid[name].G / 255, (float)colgrid[name].B / 255);
                    Gl.glPolygonMode(Gl.GL_FRONT_AND_BACK, Gl.GL_LINE);
                    Gl.glRotatef((float)(3.14159 * (dt * rotspeed[name]) / 180.0),
0, 0, 1f);
                    DrawSphere(r, nx, ny);
                    Gl.glRotatef(-(float)(3.14159 * (dt * rotspeed[name]) / 180.0),
0, 0, 1f);
                    break;
                case CheckState.Unchecked:
                    break;
            }

//текстура планет
            switch (checkBox4.CheckState)
            {
                case CheckState.Checked:
                    Gl.glEnable(Gl.GL_TEXTURE_2D);
                    Gl.glTexEnvf(Gl.GL_TEXTURE_ENV, Gl.GL_TEXTURE_ENV_MODE,
Gl.GL_DECAL);
                    switch (name)
                    {
                        case 0:
                            Gl.glBindTexture(Gl.GL_TEXTURE_2D, Texture);
                            break;
                        case 1:
```

```
                            Gl.glBindTexture(Gl.GL_TEXTURE_2D, Texture2);
                            break;
                        case 2:
                            Gl.glBindTexture(Gl.GL_TEXTURE_2D, Texture3);
                            break;
                    }
                    break;
                case CheckState.Unchecked:
                    break;
            }

            //свет вкл
            switch (checkBox2.CheckState)
            {
                case CheckState.Checked:
                    Gl.glEnable(Gl.GL_LIGHTING);
                    Gl.glEnable(Gl.GL_LIGHT0);
                    Gl.glEnable(Gl.GL_NORMALIZE);
                    Gl.glEnable(Gl.GL_COLOR_MATERIAL);
                    Gl.glLightModeli(Gl.GL_LIGHT_MODEL_TWO_SIDE, 1);
                    break;
                case CheckState.Unchecked:
                    break;
            }

            //Внутренность планеты
            switch (checkBox3.CheckState)
            {
                case CheckState.Checked:
                    Gl.glEnable(Gl.GL_POLYGON_OFFSET_FILL);
                    Gl.glPolygonOffset(1f, 1f);
                    Gl.glPolygonMode(Gl.GL_FRONT_AND_BACK, Gl.GL_FILL);

                    Gl.glEnable(Gl.GL_BLEND);
                    Gl.glBlendFunc(Gl.GL_SRC_ALPHA, Gl.GL_ONE_MINUS_SRC_ALPHA);

                    Gl.glColor4f((float)colplan[name].R / 255,
(float)colplan[name].G / 255, (float)colplan[name].B / 255, vis[name]);
                    //Gl.glColor3f((float)colplan[name].R / 255,
(float)colplan[name].G / 255, (float)colplan[name].B / 255);
                    Gl.glRotatef((float)(3.14159 * (dt * rotspeed[name]) / 180.0),
0, 0, 1f);
                    DrawSphere(r, nx, ny);
                    Gl.glRotatef(-(float)(3.14159 * (dt * rotspeed[name]) / 180.0),
0, 0, 1f);
                    break;
                case CheckState.Unchecked:
                    break;
            }


            //выкл свет
            switch (checkBox2.CheckState)
            {
                case CheckState.Checked:
                    Gl.glDisable(Gl.GL_COLOR_MATERIAL);
                    Gl.glDisable(Gl.GL_NORMALIZE);
                    Gl.glDisable(Gl.GL_LIGHT0);
                    Gl.glDisable(Gl.GL_LIGHTING);
                    break;
                case CheckState.Unchecked:
                    break;
            }

            Gl.glDisable(Gl.GL_TEXTURE_2D);
```

```
            Gl.glDisable(Gl.GL_POLYGON_OFFSET_FILL);
            Gl.glDisable(Gl.GL_BLEND);
            //обратное смещение в (0,0,0)
            switch (name)
            {
                case 1:
                    Gl.glTranslatef(-(float)Math.Cos(2 * 3.14159 * (dt * wearth +
30) / 100.0) * orbita[name - 1],-(float)Math.Sin(2 * 3.14159 * (dt * wearth + 30) /
100.0) * orbita[name - 1], 0);
                    break;
                case 2:
                    Gl.glTranslatef(-(float)Math.Cos(2 * 3.14159 * (dt * wearth +
30) / 100.0) * orbita[name - 2],-(float)Math.Sin(2 * 3.14159 * (dt * wearth + 30) /
100.0) * orbita[name - 2], 0);
                    Gl.glTranslatef((float)Math.Cos(2 * 3.14159 * (dt * wmoon + 30)
/ 100.0) * orbita[name - 1], (float)Math.Sin(2 * 3.14159 * (dt * wmoon + 30) /
100.0) * orbita[name - 1], 0);
                    break;
            }
        }

        void DrawingOrder()
        {
            //Солнце 0
            //Земля  1
            //Луна   2

            // Прозрачность вынуждает использовать данную конструкцию, для
корректной отрисовки сначала менее прозрачных элементов,
            // затем более прозрачных
            if (vis[0] >= vis[1])
            {
                if (vis[0] <= vis[2])
                {
                    SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                    SpherePlanet(radius[0], meshfi[1], meshtet[1], 0);
                    SpherePlanet(radius[1], meshfi[0], meshtet[0], 1);
                }
                else
                {
                    if (vis[1] >= vis[2])
                    {
                        SpherePlanet(radius[0], meshfi[0], meshtet[0], 0);
                        SpherePlanet(radius[1], meshfi[1], meshtet[1], 1);
                        SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                    }
                    else
                    {
                        SpherePlanet(radius[0], meshfi[0], meshtet[0], 0);
                        SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                        SpherePlanet(radius[1], meshfi[1], meshtet[1], 1);
                    }
                }
            }
            else
            {
                if (vis[0] >= vis[2])
                {
                    SpherePlanet(radius[1], meshfi[1], meshtet[1], 1);
                    SpherePlanet(radius[0], meshfi[0], meshtet[0], 0);
                    SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                }
                else
                {
                    if (vis[1] >= vis[2])
```

```
                    {
                        SpherePlanet(radius[1], meshfi[1], meshtet[1], 1);
                        SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                        SpherePlanet(radius[0], meshfi[0], meshtet[0], 0);
                    }
                    else
                    {
                        SpherePlanet(radius[2], meshfi[2], meshtet[2], 2);
                        SpherePlanet(radius[1], meshfi[1], meshtet[1], 1);
                        SpherePlanet(radius[0], meshfi[0], meshtet[0], 0);
                    }

                }
            }
        }

//инициализация шрифта
        void CreateFont3D(Font font)
        {
            Gdi.SelectObject(HDC3D, font.ToHfont());
            Font3D = Gl.glGenLists(256);
            Wgl.wglUseFontBitmapsA(HDC3D, 0, 256, Font3D);
        }

        //определяет вывод текста
        void OutText3D(float x, float y, float z, string Text)
        {
            Gl.glRasterPos3f(x, y, z);
            Gl.glPushAttrib(Gl.GL_LIST_BIT);
            Gl.glListBase(Font3D);
            byte[] bText = MyGL.RussianEncoding.GetBytes(Text);
            Gl.glCallLists(Text.Length, Gl.GL_UNSIGNED_BYTE, bText);
            Gl.glPopAttrib();
        }

        //удаление шрифта
        void DeleteFont3D()
        {
            if (Font3D != 0)
            {
                Gl.glDeleteLists(Font3D, 256);
                Font3D = 0;
            }
        }




        //вызов перерисовки
        void InvalidateRect()
        {
            MyGL.InvalidateRect(Handle, IntPtr.Zero, false);
        }

        //ловит сообщения об перерисовке
        protected override void WndProc(ref Message m)
        {
            base.WndProc(ref m);
            if (m.Msg == MyGL.WM_ERASEBKGND)
            {
                m.Result = IntPtr.Zero;
                InvalidateRect();
            }
        }
```

```
        //считывание текстуры из файла
        static uint LoadTexture(string fileName)
        {
            uint texObject = 0;
            try
            {
                Bitmap bmp = new Bitmap(fileName);
                bmp.RotateFlip(RotateFlipType.RotateNoneFlipY);
                BitmapData bmpdata = bmp.LockBits(new Rectangle(0, 0, bmp.Width,
bmp.Height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
                texObject = MakeGLTexture(bmpdata.Scan0, bmpdata.Width,
bmpdata.Height);
                bmp.UnlockBits(bmpdata);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);

            }
            return texObject;
        }
        //создает текстуру в памяти
        static uint MakeGLTexture(IntPtr pixels, int w, int h)
        {
            uint texObject;
            Gl.glGenTextures(1, out texObject);
            Gl.glPixelStorei(Gl.GL_UNPACK_ALIGNMENT, 1);
            Gl.glBindTexture(Gl.GL_TEXTURE_2D, texObject);
            Gl.glTexParameteri(Gl.GL_TEXTURE_2D, Gl.GL_TEXTURE_MAG_FILTER,
Gl.GL_NEAREST);
            Gl.glTexParameteri(Gl.GL_TEXTURE_2D, Gl.GL_TEXTURE_MIN_FILTER,
Gl.GL_NEAREST);

            Gl.glTexImage2D(Gl.GL_TEXTURE_2D, 0, Gl.GL_RGB, w, h, 0, Gl.GL_BGR,
Gl.GL_UNSIGNED_BYTE, pixels);
            return texObject;
        }



//поворот осей (наклон)
        private void trackBar_fi_Scroll(object sender, EventArgs e)
        {
            fi = trackBar_fi.Value;
            InvalidateRect();
        }

        //поворот осей (поворот)
        private void trackBar_psi_Scroll(object sender, EventArgs e)
        {
            psi = trackBar_psi.Value;
            InvalidateRect();
        }

        //область просмотра
        private void trackBar_r_Scroll(object sender, EventArgs e)
        {
            r = trackBar_r.Value * 0.6f;
            InvalidateRect();
        }

        //регулятор движения
        private void timer1_Tick(object sender, EventArgs e)
        {
            dt = dt + 0.1f;
```

```
            InvalidateRect();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (timer1.Enabled == true)
            {
                timer1.Stop();
            }
            else
            {
                timer1.Start();
            }
        }


        //сетка, свет, планета, текстура
        private void checkBox1_CheckedChanged(object sender, EventArgs e) {
InvalidateRect(); }

        private void checkBox2_CheckedChanged(object sender, EventArgs e) {
InvalidateRect(); }

        private void checkBox3_CheckedChanged(object sender, EventArgs e)
        {
            switch (checkBox4.CheckState)
            {
                case CheckState.Checked:
                    {
                        checkBox3.CheckState = CheckState.Checked;
                        break;
                    };
                case CheckState.Unchecked:
                    {

                        break;
                    };
            }
            InvalidateRect();
        }

        private void checkBox4_CheckedChanged(object sender, EventArgs e)
        {
            switch (checkBox4.CheckState)
            {
                case CheckState.Checked:
                    {
                        checkBox3.CheckState = CheckState.Checked;
                        break;
                    };
                case CheckState.Unchecked:
                    {

                        break;
                    };
            }
            InvalidateRect();
        }


        //угловые скорости
        private void trackBar_wearth_Scroll(object sender, EventArgs e)
        {
            wearth = trackBar_wearth.Value;
            textBox_wE.Text = wearth.ToString();
            InvalidateRect();
```

```
}
private void trackBar_wmoon_Scroll(object sender, EventArgs e)
{
    wmoon = trackBar_wmoon.Value;
    textBox_wM.Text = wmoon.ToString();
    InvalidateRect();
}


//замена значений орбит и разбиения сетки планет
private void button2_Click(object sender, EventArgs e)
{
    orbita[0] = float.Parse(textBoxEarth.Text);
    orbita[1] = float.Parse(textBoxMoon.Text);

    meshfi[0]= float.Parse(textBox_meshS_fi.Text);
    meshfi[1]= float.Parse(textBox_meshE_fi.Text);
    meshfi[2]= float.Parse(textBox_meshM_fi.Text);

    meshtet[0]=float.Parse(textBox_meshS_tet.Text);
    meshtet[1] = float.Parse(textBox_meshE_tet.Text);
    meshtet[2] = float.Parse(textBox_meshM_tet.Text);

    InvalidateRect();
}


//регулятор прозрачности
private void trackBar_VisSun_Scroll(object sender, EventArgs e)
{
    vis[0] = trackBar_VisSun.Value * 0.1f;
    InvalidateRect();
}
private void trackBar_VisEarth_Scroll(object sender, EventArgs e)
{
    vis[1] = trackBar_VisEarth.Value * 0.1f;
    InvalidateRect();
}
private void trackBar_VisMoon_Scroll(object sender, EventArgs e)
{
    vis[2] = trackBar_VisMoon.Value * 0.1f;
    InvalidateRect();
}


//регулятор радиусов
private void trackBar_rS_Scroll(object sender, EventArgs e)
{
    radius[0] = trackBar_rS.Value * 0.1f;
    InvalidateRect();
}
private void trackBar_rE_Scroll(object sender, EventArgs e)
{
    radius[1] = trackBar_rE.Value * 0.1f;
    InvalidateRect();
}
private void trackBar_rM_Scroll(object sender, EventArgs e)
{
    radius[2] = trackBar_rM.Value * 0.1f;
    InvalidateRect();
}
```

```
        //отложенная перерисовка
        class MyGL
        {
            internal const int WM_ERASEBKGND = 0x0014;
            [DllImport("user32.dll")]
            internal static extern bool InvalidateRect(IntPtr hWnd, IntPtr lpRect,
bool Erase);

            internal static Encoding RussianEncoding = Encoding.GetEncoding(1251);
        }


        //корректное завершение работы программмы
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            DeleteFont3D();
            Wgl.wglMakeCurrent(IntPtr.Zero, IntPtr.Zero);
            //уничтожение контекста устройства
            Wgl.wglDeleteContext(HRC3D);
            //уничтожение контекста воспроизведения
            User.ReleaseDC(Handle3D, HDC3D);
        }

        private void Form1_Load(object sender, EventArgs e) { }

    }
}
```