Министерство образовании и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Белгородский государственный технологический университет им. В.Г. Шухова»

(БГТУ им. В.Г. Шухова)

Кафедра Программного обеспечения вычислительной техники

и автоматизированных систем

РГЗ  
по дисциплине: Компьютерная графика

***Создание трехмерного окружения с использованием OpenGL***

Выполнил: студент группы ПВ-41

Черенков Д. М.

Проверил: Осипов О.В.

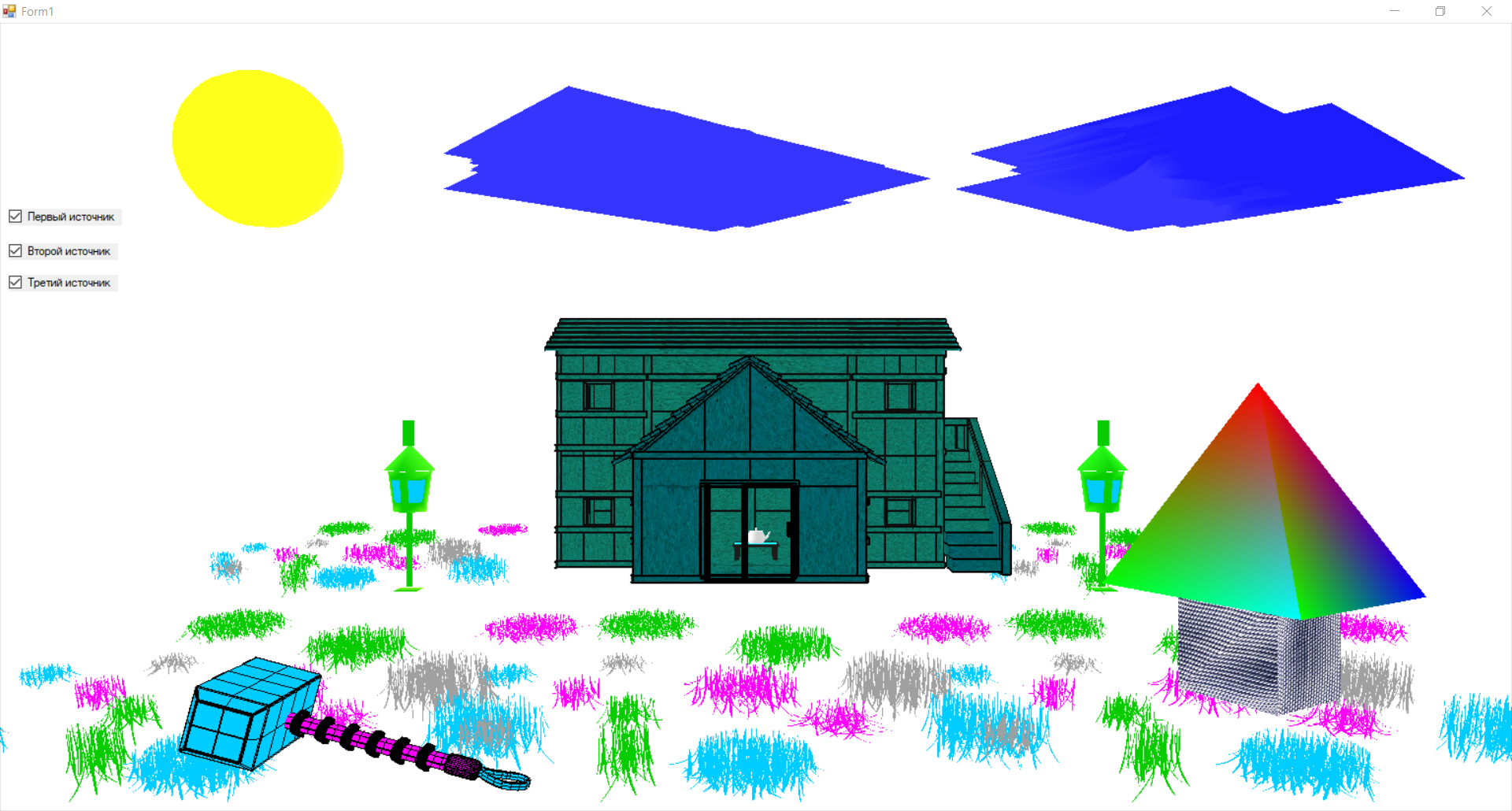
Белгород 2017 г.

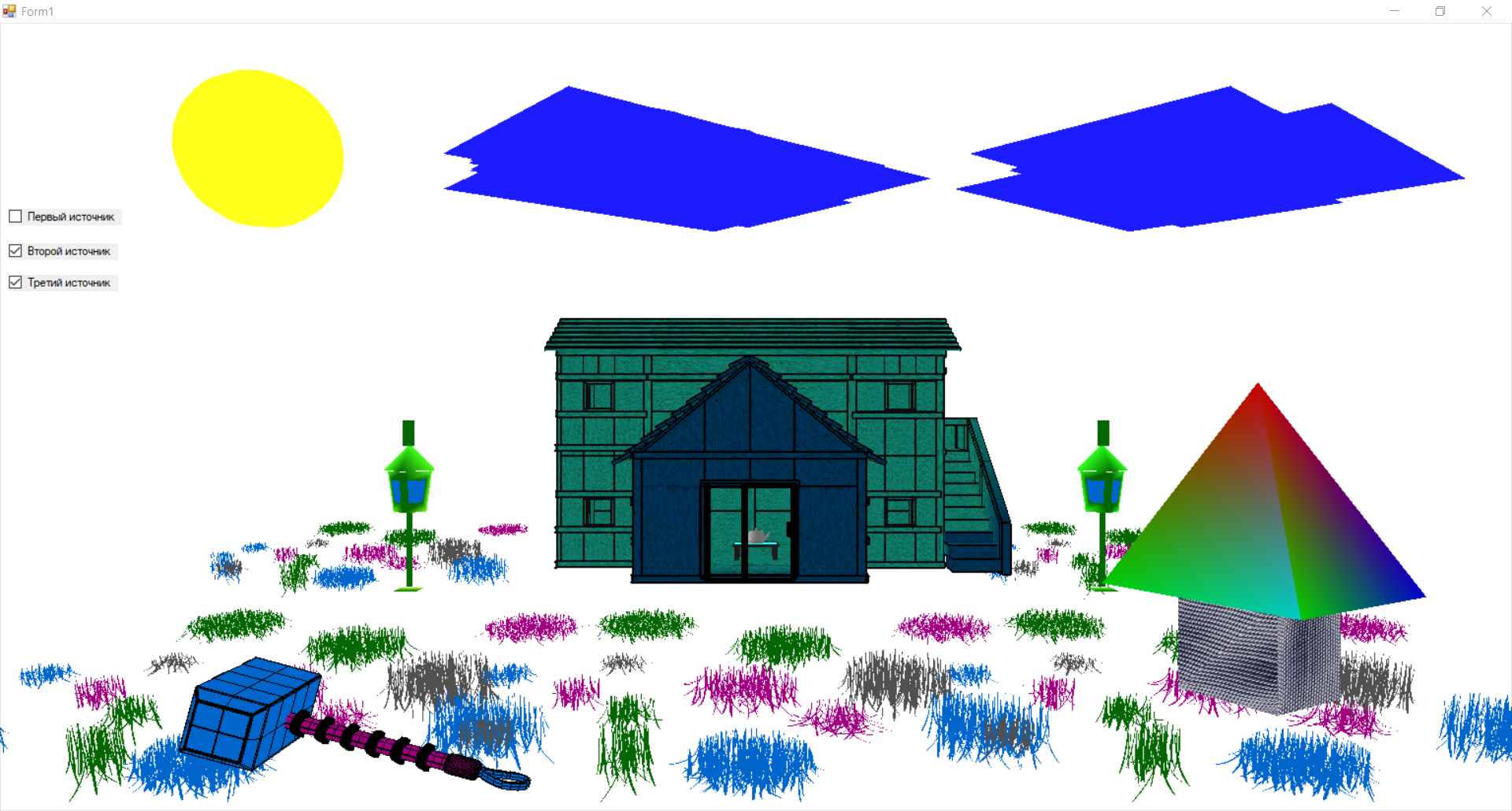
**Цель работы:** разработать изображение для визуализации трехмерных объектов внутри окружения с использованием OpenGL.

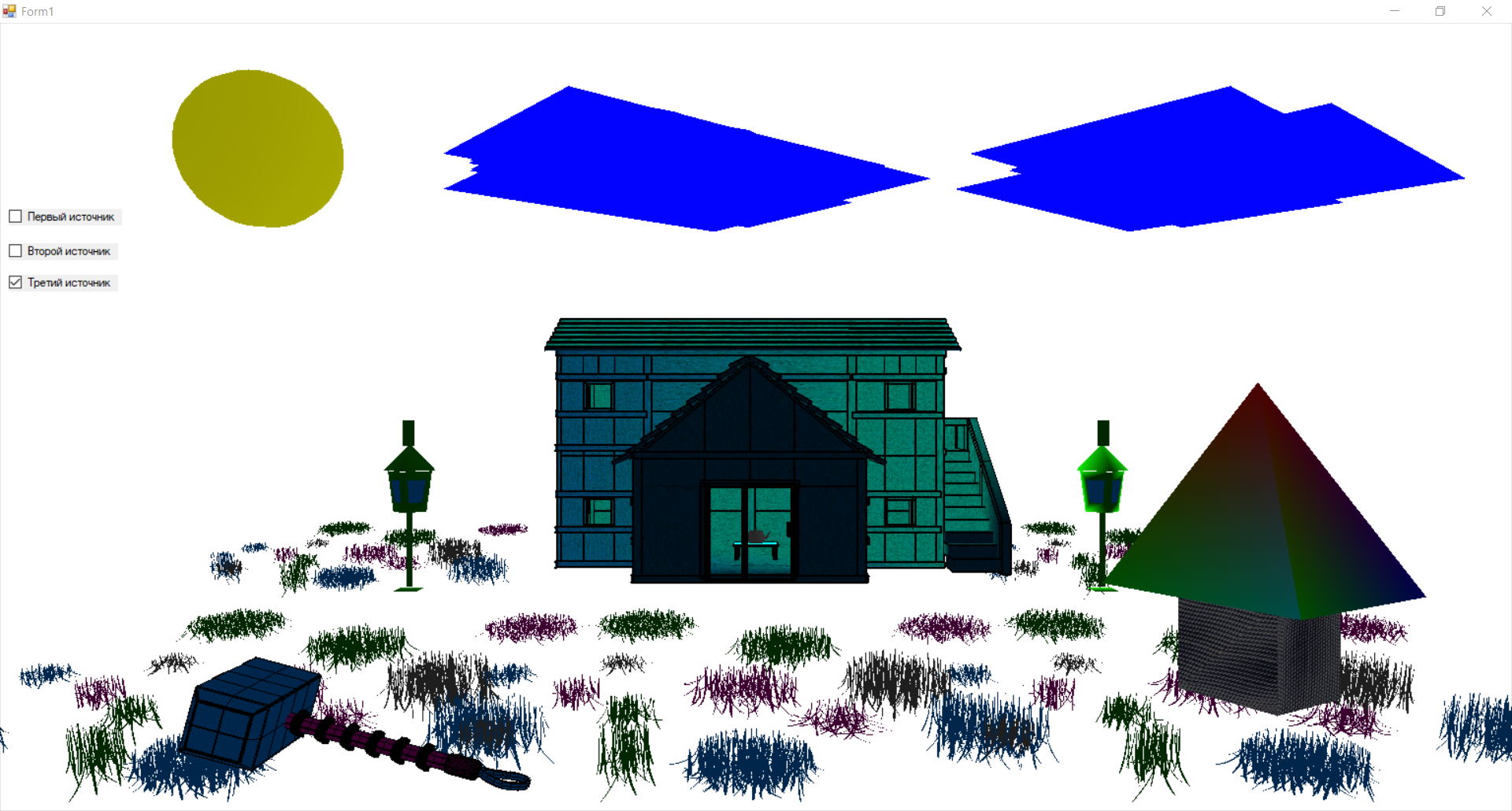
**Требования к программе**

1. Изобразить на сцене следующие объекты:
   1. Дом
   2. Объекты внутри дома (мебель)
   3. Окружение дома смоделировать по собственные усмотрения. Дверь растения, фонарь и т.д.
2. Поверхности должны иметь текстуры за исключением некоторых объектов
3. Обязательные элементы:
   1. Текстурирование
   2. Освещение
4. Предоставить возможность пользователю включать и выключать источники света (не менее 3).
5. Изобразить на сцене объекты из 6 и 7 лабораторных работ. Внутри дома, рядом с домом.
6. То, что не оговорено в задании, можно делать по собственному усмотрению.

**Скриншоты**







**Листинг программы**

**Form1.cs**

**using** System.Drawing;  
**using** System.Windows.Forms;  
**using** SharpGL;  
**using** SharpGL\_RGZ.figures;  
  
**namespace** SharpGL\_RGZ  
{  
 **public** partial **class** Form1 : Form  
 {  
 **public** Form1()  
 {  
 InitializeComponent();  
 }  
   
 **private readonly** FigureHouse \_figureHouse = **new** FigureHouse();  
 **private readonly** FigureHummer \_figureHummer = **new** FigureHummer();  
 **private readonly** FigureTable \_figureTable = **new** FigureTable();  
 **private readonly** FigureDver \_figureDver = **new** FigureDver();  
 **private readonly** FigureParallelepiped \_figureParallelepiped = **new** FigureParallelepiped();  
 **private readonly** FigureTriangle \_figureTriangle = **new** FigureTriangle();  
   
 **private** Point \_currentLocation = **new** Point(0, 0);  
 **private float** \_angleX;  
 **private float** \_angleY;  
 **private float** z = -1f;  
  
 **private float**[] \_black = {0f, 0f, 0f, 1f};  
   
 **private readonly** FigureSphere \_figureSphere1 = **new** FigureSphere();  
 **private readonly** FigureSphere \_figureSphere2 = **new** FigureSphere();  
 **private readonly** FigureSphere \_figureSphere3 = **new** FigureSphere();  
  
 **private readonly** FigureFonar \_figureFonar1 = **new** FigureFonar();  
 **private readonly** FigureFonar \_figureFonar2 = **new** FigureFonar();  
  
 **private readonly** FigureTrava \_figureTrava1 = **new** FigureTrava();  
 **private readonly** FigureCloud \_figureCloud = **new** FigureCloud();  
   
 **private float**[] \_l0pos = {-1.3f, 0.7f, -2.5f, 1.0f};  
 **private float**[] \_l1pos = {-0.34f, -0.08f, -0.95f, 1.0f};  
 **private float**[] \_l2pos = {0.34f, -0.08f, -0.95f, 1.0f};  
   
 **private float**[] \_light0Scpecular = {1.0f, 1.0f, 1.0f, 1.0f};  
 **private float**[] \_light0Ambient = {0.8f, 0.8f, 0.8f, 1f};  
 **private float**[] \_light0Diffuse = {1f, 1f, 1f, 1f};  
   
 **private float**[] \_light1Scpecular = {1f, 1f, 1f, 1f};  
 **private float**[] \_light1Ambient = {0.5f, 0.5f, 0.5f, 1f};  
 **private float**[] \_light1Diffuse = {0.6f, 0.6f, 0.6f, 0.6f};  
   
 **private float**[] \_light2Scpecular = {0.1f, 0.1f, 0.1f, 0.5f};  
 **private float**[] \_light2Ambient = {0.1f, 0.1f, 0.1f, 0.5f};  
 **private float**[] \_light2Diffuse = {0.1f, 0.1f, 0.1f, 0.5f};  
   
   
 **private float**[] \_sRef = {0.1f, 0.1f, 0.1f, 1.0f};  
  
 **public void** init()  
 {  
 \_l2pos = **new**[] {0.34f + x1, -0.08f + y1, -0.95f + z1, 1.0f};  
 }  
   
 **private void** openGLControl1\_OpenGLDraw(**object** sender, RenderEventArgs args)  
 {  
 init();  
 var gl = openGLControl1.OpenGL;  
   
 gl.Enable(OpenGL.GL\_LIGHTING);  
*// gl.LightModel(OpenGL.GL\_LIGHT\_MODEL\_AMBIENT, \_black);* gl.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT);  
 gl.ClearColor(1f, 1f, 1f, 1f);  
   
 gl.Enable(OpenGL.GL\_BLEND); *// Разрешить прозрачность.* gl.Enable(OpenGL.GL\_POINT\_SMOOTH); *// Разрешить сглаживание точек.* gl.Enable(OpenGL.GL\_COLOR\_MATERIAL); *// Отключить перелевание цвета.* gl.Enable(OpenGL.GL\_TEXTURE\_2D);  
 gl.PointSize(16); *// Размер точки.* gl.LineWidth(2); *// Толщина линий.* gl.LoadIdentity();  
 gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_POSITION, \_l0pos);  
 \_figureSphere1.Draw(gl, \_l0pos[0], \_l0pos[1], \_l0pos[2], 0, 0, 2f, 0);  
 gl.LoadIdentity();  
   
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.GL\_POSITION, \_l1pos);  
*// \_figureSphere2.Draw(gl, \_l1pos[0], \_l1pos[1], \_l1pos[2], 0, 0, 0.1f, 0);  
// gl.LoadIdentity();* \_figureFonar1.Draw(gl, \_l1pos[0] - 0.006f, \_l1pos[1] - 0.1f, \_l1pos[2], 0, 0, 0.03f, 0);  
 gl.LoadIdentity();  
  
 gl.Light(OpenGL.GL\_LIGHT2, OpenGL.GL\_POSITION, \_l2pos);  
*// \_figureSphere3.Draw(gl, \_l2pos[0], \_l2pos[1], \_l2pos[2], 0, 0, 0.1f, 0);  
// gl.LoadIdentity();* \_figureFonar2.Draw(gl, \_l2pos[0] + 0.006f, \_l2pos[1] - 0.1f, \_l2pos[2], 0, 0, 0.03f, 0);  
 gl.LoadIdentity();  
   
 gl.LoadIdentity();  
 \_figureCloud.Draw(gl, -0.5f, 1.5f, -5.55f);  
 gl.LoadIdentity();  
 \_figureCloud.Draw(gl, 2.5f, 1.5f, -5.55f);  
   
   
 **if** (checkBox1.Checked)  
 {  
 gl.Enable(OpenGL.GL\_LIGHT0);  
 }  
 **else** {  
 gl.Disable(OpenGL.GL\_LIGHT0);   
 }  
  
 **if** (checkBox2.Checked)  
 {  
 gl.Enable(OpenGL.GL\_LIGHT1);  
 }  
 **else** {  
 gl.Disable(OpenGL.GL\_LIGHT1);   
 }  
  
 **if** (checkBox3.Checked)  
 {  
 gl.Enable(OpenGL.GL\_LIGHT2);   
 }  
 **else** {  
 gl.Disable(OpenGL.GL\_LIGHT2);   
 }  
   
 gl.LoadIdentity();  
 \_figureHouse.Draw(gl, 0, 0, z, \_angleX, \_angleY, 1f, 0);  
 gl.LoadIdentity();  
 \_figureTable.Draw(gl, 0, 0, z, \_angleX, \_angleY, 1f, 0);  
 gl.LoadIdentity();  
 \_figureDver.Draw(gl, 0, 0, 0.16f + z, \_angleX, \_angleY, 1f, 0);  
 gl.LoadIdentity();  
 \_figureParallelepiped.Draw(gl, 0.29f, -0.13f, -0.55f);  
 gl.LoadIdentity();  
 \_figureTriangle.Draw(gl, 0.29f, -0.04f, -0.55f);   
 gl.LoadIdentity();  
 \_figureHummer.Draw(gl, -0.29f, -0.17f, -0.55f);  
   
 gl.LoadIdentity();  
 \_figureTrava1.Draw(gl, 0, -0.55f, z - 1f, \_angleX, 0, 0.5f, 0);  
 gl.LoadIdentity();  
 \_figureTrava1.Draw(gl, 1f, -0.55f, z - 1f, \_angleX, 0, 0.5f, 0);  
 gl.LoadIdentity();  
 \_figureTrava1.Draw(gl, -1f, -0.55f, z - 1f, \_angleX, 0, 0.5f, 0);  
 gl.LoadIdentity();  
 \_figureTrava1.Draw(gl, -1.6f, -0.55f, z - 3f, \_angleX, 0, 0.5f, 0);  
 gl.LoadIdentity();  
 \_figureTrava1.Draw(gl, 1.6f, -0.55f, z - 3f, \_angleX, 0, 0.5f, 0);  
   
   
 gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_DIFFUSE, \_light0Diffuse);  
 gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_AMBIENT, \_light0Ambient);  
 gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_SPECULAR, \_light0Scpecular);  
   
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.GL\_DIFFUSE, \_light1Diffuse);  
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.GL\_AMBIENT, \_light1Ambient);  
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.GL\_SPECULAR, \_light1Scpecular);  
   
 gl.Light(OpenGL.GL\_LIGHT2, OpenGL.GL\_DIFFUSE, \_light2Diffuse);  
 gl.Light(OpenGL.GL\_LIGHT2, OpenGL.GL\_AMBIENT, \_light2Ambient);  
 gl.Light(OpenGL.GL\_LIGHT2, OpenGL.GL\_SPECULAR, \_light2Scpecular);  
   
 gl.Color(0.5f, 0.5f, 0.5f);  
 gl.LoadIdentity();  
 gl.Translate(0, -0.125f, z);  
 var tea = **new** SharpGL.SceneGraph.Primitives.Teapot();  
 tea.Draw(gl, 14, 0.01, OpenGL.GL\_FILL);  
   
   
 gl.ColorMaterial(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_AMBIENT\_AND\_DIFFUSE);  
 gl.Material(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_SPECULAR, \_sRef);  
 gl.Material(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_SHININESS, 64);  
   
 gl.ShadeModel(OpenGL.GL\_SMOOTH);  
 }  
   
 **private void** openGLControl1\_MouseMove(**object** sender, MouseEventArgs e)  
 {  
 **if** (e.Button != MouseButtons.Left) **return**;  
 \_angleY += (e.Location.X - \_currentLocation.X) / 1.0f;  
 \_angleX += (e.Location.Y - \_currentLocation.Y) / 1.0f;  
 \_currentLocation = e.Location;  
 }  
   
 **private void** openGLControl1\_MouseDown(**object** sender, MouseEventArgs e)  
 {  
 **if** (e.Button != MouseButtons.Left) **return**;  
 \_currentLocation = e.Location;  
 }  
   
 **protected override void** OnMouseWheel(MouseEventArgs e)  
 {  
 z += e.Delta > 0 ? 0.1f : -0.1f;  
 **base**.OnMouseWheel(e);  
 }  
   
 **private static float** x1 = 0;  
 **private static float** y1 = 0;  
   
 **private static float** x2 = 0;  
 **private static float** y2 = 0;  
   
 **private static float** x3 = 0;  
 **private static float** y3 = 0;  
   
 **private static float** z1 = 0;  
 **private static float** z2 = 0;  
   
 **private void** openGLControl1\_KeyDown(**object** sender, KeyEventArgs e)  
 {   
 **switch** (e.KeyCode)  
 {  
 **case** Keys.Q:  
 **if** (e.Shift)  
 {  
 x1 += 0.01f;  
 }  
 **else** {  
 x1 -= 0.01f;  
 }  
 **break**;  
 **case** Keys.W:  
 **if** (e.Shift)  
 {  
 y1 += 0.01f;  
 }  
 **else** {  
 y1 -= 0.01f;  
 }  
 **break**;  
 **case** Keys.R:  
 **if** (e.Shift)  
 {  
 z1 += 0.01f;  
 }  
 **else** {  
 z1 -= 0.01f;  
 }  
 **break**;  
 }  
 }  
  
 }  
}

**FigureSphere.cs** (аналогичные классы: «FigureCloud», « FigureDver», « FigureFonar», «FigureHouse», « FigureHummer», « FigureSphere», « FigureTable», « FigureTrava»)

**using** System.Collections.Generic;  
**using** SharpGL;  
**using** SharpGL.SceneGraph.Assets;  
**using** SharpGL\_6;  
**using** SharpGL\_6.figures;  
  
**namespace** SharpGL\_7.figures  
{  
 **public class** FigureHummer  
 {  
 **private readonly** List<Polygon> **\_polygons**;  
  
 **public** FigureHummer()  
 {  
 **\_polygons** = LoadPrimitive.Load(  
 **"C:\\Users\\User\\RiderProjects\\SharpGL\_7\\SharpGL\_7\\obj\_files\\sphere.obj"**);  
 }  
  
 **private** Texture \_texture = **new** Texture();  
  
 **public void** Draw(OpenGL gl, **float** ta, **float** ty, **float** tz, **float** angleX, **float** angleY, **float** scale, **float** z)  
 {  
*// \_texture.Create(gl, "C:\\Users\\User\\RiderProjects\\SharpGL\_7\\SharpGL\_7\\files\\Eiche.bmp");* scale = 0.1f \* scale;  
  
 gl.Translate(ta, ty, tz);  
 gl.Scale(scale, scale, scale);  
*// \_texture.Bind(gl);* **foreach** (**var** polygon **in \_polygons**)  
 {  
 gl.Begin(OpenGL.***GL\_POLYGON***);  
 gl.Color(polygon.**color**.**R**, polygon.**color**.**G**, polygon.**color**.**B**);  
 **var** i = 0;  
 **foreach** (**var** points **in** polygon.**list**)  
 {  
*// gl.TexCoord(i == 0 || i == 1 ? 0f : 1f, i == 0 || i == 3 ? 0f : 1f);* gl.Vertex(points.**Item1**, points.**Item2**, points.**Item3**);  
 i += 1;  
 }  
 gl.End();  
 }  
*// \_texture.Destroy(gl);* }  
 }  
}

**Polygon.cs**

**using** System;  
**using** System.Collections.Generic;  
**using** System.Drawing;  
  
**namespace** SharpGL\_6  
{  
 **public class** Polygon  
 {  
 **public** Color **color**;  
 **public readonly** List<Tuple<**float**, **float**, **float**>> **list** = **new** List<Tuple<**float**, **float**, **float**>>();  
  
 **public** Polygon(Color color)  
 {  
 **this**.**color** = color;  
 }  
  
 **public void** AddPoint(Tuple<**float**, **float**, **float**> point)  
 {  
 **list**.Add(point);  
 }  
 **public void** AddPoint(**float** x, **float** y, **float** z)  
 {  
 AddPoint(**new** Tuple<**float**, **float**, **float**>(x, y, z));  
 }  
   
 }  
}

**LoadPrimitive.cs**

**using** System.Collections.Generic;  
**using** System.Drawing;  
**using** System.IO;  
**using** ObjLoader.Loader.Loaders;  
  
**namespace** SharpGL\_6.figures  
{  
 **public static class** LoadPrimitive  
 {   
 **public static** List<Polygon> Load(**string** path)  
 {  
 **var** polygons = **new** List<Polygon>();  
   
 **var** objLoaderFactory = **new** ObjLoaderFactory();  
 **var** objLoader = objLoaderFactory.Create();  
 **var** fileStream = **new** FileStream(path, FileMode.***Open***);  
 **var** loadedObj = objLoader.Load(fileStream);  
  
 **var** colors = **new**[]  
 {  
 Color.FromArgb(255, 255, 128, 64),  
 Color.FromArgb(255, 5, 128, 0),  
 Color.FromArgb(255, 255, 0, 64),  
 Color.FromArgb(255, 200, 0, 160),  
 Color.FromArgb(255, 60, 0, 64),  
 Color.FromArgb(255, 0, 128, 255),  
 Color.FromArgb(255, 100, 100, 100),  
 Color.**Crimson** };  
 **var** k = -1;  
 **foreach** (**var** g **in** loadedObj.**Groups**)  
 {  
 k = (k + 1) % colors.**Length**;  
 **var** color = colors[k++];  
 **foreach** (**var** f **in** g.**Faces**)  
 {  
 **var** p = **new** Polygon(color);  
 **for** (**var** i = 0; i < f.**Count**; i++)  
 {  
 p.AddPoint(  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**X**,  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**Y**,  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**Z** );  
 }  
 polygons.Add(p);  
 }  
 }  
  
 fileStream.Close();  
 **return** polygons;  
 }  
 }  
}

**FigureParallelepiped.cs**

**using** SharpGL;  
**using** SharpGL.SceneGraph.Assets;  
  
**namespace** SharpGL\_RGZ.figures  
{  
 **public class** FigureParallelepiped  
 {  
 **private float a** = 1, **b** = 1, **c** = 1;  
  
 **public** FigureParallelepiped(**float** a, **float** b, **float** c)  
 {  
 **this**.**a** = a;  
 **this**.**b** = b;  
 **this**.**c** = c;  
 }  
  
 **public** FigureParallelepiped()  
 {  
 }  
   
 **private** Texture **\_texture** = **new** Texture();  
   
 **public void** Draw(OpenGL gl, **float** ta, **float** ty, **float** tz)  
 {  
 **\_texture**.Create(gl, **"C:\\Users\\User\\RiderProjects\\SharpGL\_7\\SharpGL\_7\\files\\12387996.jpg"**);  
 **var** scale = 0.06f \* 1;  
   
 gl.Translate(ta, ty, tz);  
 gl.Rotate(30, 0, 1f, 0);  
*// gl.LookAt(0, 0, z, 0, 0, z + 10, 0 , 1, 0);* gl.Scale(scale, scale, scale);  
  
 gl.Normal(0f, 0f, 1f);  
 gl.Begin(OpenGL.***GL\_QUAD\_STRIP***);  
 gl.Color(1f, 1f, 1f);   
 **\_texture**.Bind(gl);  
 gl.TexCoord(0f, 0f); gl.Vertex(-**a** / 2, -**b** / 2, -**c** / 2);  
 gl.TexCoord(1f, 0f); gl.Vertex(-**a** / 2, -**b** / 2, **c** / 2);  
 gl.TexCoord(1f, 1f); gl.Vertex(-**a** / 2, **b** / 2, -**c** / 2);  
 gl.TexCoord(0f, 1f); gl.Vertex(-**a** / 2, **b** / 2, **c** / 2);  
  
 gl.TexCoord(0f, 0f); gl.Vertex(**a** / 2, **b** / 2, -**c** / 2);  
 gl.TexCoord(1f, 0f); gl.Vertex(**a** / 2, **b** / 2, **c** / 2);  
 gl.TexCoord(1f, 1f);  
 gl.TexCoord(0f, 1f);  
  
 gl.TexCoord(0f, 0f);   
 gl.TexCoord(1f, 0f); gl.Vertex(**a** / 2, -**b** / 2, -**c** / 2);  
 gl.TexCoord(1f, 1f); gl.Vertex(**a** / 2, -**b** / 2, **c** / 2);  
 gl.TexCoord(0f, 1f);   
   
 gl.TexCoord(0f, 0f);   
 gl.TexCoord(1f, 0f);   
 gl.TexCoord(1f, 1f); gl.Vertex(-**a** / 2, -**b** / 2, -**c** / 2);  
 gl.TexCoord(0f, 1f); gl.Vertex(-**a** / 2, -**b** / 2, **c** / 2);  
 gl.End();   
  
 gl.Begin(OpenGL.***GL\_QUADS***);  
  
 gl.TexCoord(0f, 0f); gl.Vertex(-**a** / 2, -**b** / 2, **c** / 2);  
 gl.TexCoord(1f, 0f); gl.Vertex(-**a** / 2, **b** / 2, **c** / 2);  
 gl.TexCoord(1f, 1f); gl.Vertex(**a** / 2, **b** / 2, **c** / 2);  
 gl.TexCoord(0f, 1f); gl.Vertex(**a** / 2, -**b** / 2, **c** / 2);  
  
 gl.TexCoord(0f, 0f); gl.Vertex(-**a** / 2, -**b** / 2, -**c** / 2);  
 gl.TexCoord(1f, 0f); gl.Vertex(-**a** / 2, **b** / 2, -**c** / 2);  
 gl.TexCoord(1f, 1f); gl.Vertex(**a** / 2, **b** / 2, -**c** / 2);  
 gl.TexCoord(0f, 1f); gl.Vertex(**a** / 2, -**b** / 2, -**c** / 2);  
   
  
 gl.End();  
 **\_texture**.Destroy(gl);  
 }  
   
 }  
}

**FigureTriangle.cs**

**using** SharpGL;  
  
**namespace** SharpGL\_RGZ.figures  
{  
 **public class** FigureTriangle  
 {  
 **public void** Draw(OpenGL gl, **float** ta, **float** ty, **float** tz)  
 {  
 **var** scale = 0.06f \* 1f;  
  
 gl.Translate(ta, ty, tz);  
 gl.Rotate(30, 0, 1f, 0);  
 gl.Scale(scale, scale, scale);  
 gl.Begin(OpenGL.***GL\_TRIANGLES***);  
  
 gl.Color(1f, 0, 0);  
 gl.Vertex(0.0f, 1.0f, 0f);  
 gl.Color(0, 1f, 0);  
 gl.Vertex(-1.0f, -1.0f, 1.0f);  
 gl.Color(0, 0, 1f);  
 gl.Vertex(1.0f, -1.0f, 1.0f);  
  
 gl.Color(1f, 0, 0);  
 gl.Vertex(0.0f, 1.0f, 0f);  
 gl.Color(0, 0, 1f);  
 gl.Vertex(1.0f, -1.0f, 1.0f);  
 gl.Color(0, 1f, 0);  
 gl.Vertex(1.0f, -1.0f, -1.0f);  
  
 gl.Color(1f, 0, 0);  
 gl.Vertex(0.0f, 1.0f, 0f);  
 gl.Color(0, 1f, 0);  
 gl.Vertex(1.0f, -1.0f, -1.0f);  
 gl.Color(0, 0, 1f);  
 gl.Vertex(-1.0f, -1.0f, -1.0f);  
  
 gl.Color(1f, 0, 0);  
 gl.Vertex(0.0f, 1.0f, 0f);  
 gl.Color(0, 1f, 0);  
 gl.Vertex(-1.0f, -1.0f, -1.0f);  
 gl.Color(0, 1f, 1f);  
 gl.Vertex(-1.0f, -1.0f, 1.0f);  
  
 gl.End();  
  
 gl.Begin(OpenGL.***GL\_QUADS***);  
 gl.Color(1f, 0, 0);  
 gl.Vertex(-1.0f, -1.0f, -1.0f);  
 gl.Color(0, 1f, 0);  
 gl.Vertex(-1.0f, -1.0f, 1.0f);  
 gl.Color(0, 1f, 1f);  
 gl.Vertex(1.0f, -1.0f, 1.0f);  
 gl.Color(0, 1f, 1f);  
 gl.Vertex(1.0f, -1.0f, -1.0f);  
 gl.End();  
 }  
 }  
}

**LoadPrimitive.cs**

**using** System.Collections.Generic;  
**using** System.Drawing;  
**using** System.IO;  
**using** ObjLoader.Loader.Loaders;  
  
**namespace** SharpGL\_RGZ.figures  
{  
 **public static class** LoadPrimitive  
 {   
 **public static** List<Polygon> Load(**string** path)  
 {  
 **var** polygons = **new** List<Polygon>();  
   
 **var** objLoaderFactory = **new** ObjLoaderFactory();  
 **var** objLoader = objLoaderFactory.Create();  
 **var** fileStream = **new** FileStream(path, FileMode.***Open***);  
 **var** loadedObj = objLoader.Load(fileStream);  
  
 **var** colors = **new**[]  
 {  
 Color.FromArgb(255, 0, 128, 255),  
 Color.FromArgb(255, 255, 215, 0),  
 Color.FromArgb(255, 5, 128, 0),  
 Color.FromArgb(255, 255, 0, 64),  
 Color.FromArgb(255, 200, 0, 160),  
 Color.FromArgb(255, 60, 0, 64),  
 Color.FromArgb(255, 100, 100, 100),  
 Color.Crimson  
 };  
 **var** k = -1;  
 **foreach** (**var** g **in** loadedObj.**Groups**)  
 {  
 k = (k + 1) % colors.**Length**;  
 **var** color = colors[k++];  
 **foreach** (**var** f **in** g.**Faces**)  
 {  
 **var** p = **new** Polygon(color);  
 **for** (**var** i = 0; i < f.**Count**; i++)  
 {  
 p.AddPoint(  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**X**,  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**Y**,  
 loadedObj.**Vertices**[f[i].**VertexIndex** - 1].**Z** );  
 }  
  
 polygons.Add(p);  
 }  
 }  
  
 fileStream.Close();  
 **return** polygons;  
 }  
 }  
}

**LoadPrimitive.cs**

**using** System;  
**using** System.Collections.Generic;  
**using** System.Drawing;  
  
**namespace** SharpGL\_RGZ.figures  
{  
 **public class** Polygon  
 {  
 **public** Color **color**;  
 **public readonly** List<Tuple<**float**, **float**, **float**>> **list** = **new** List<Tuple<**float**, **float**, **float**>>();  
  
 **public** Polygon(Color color)  
 {  
 **this**.**color** = color;  
 }  
  
 **public void** AddPoint(Tuple<**float**, **float**, **float**> point)  
 {  
 **list**.Add(point);  
 }  
   
 **public void** AddPoint(**float** x, **float** y, **float** z)  
 {  
 AddPoint(**new** Tuple<**float**, **float**, **float**>(x, y, z));  
 }  
   
 }  
}