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

Лабораторная работа № 7  
по дисциплине: Компьютерная графика

***Освещение и текстурированние OpenGL***

Выполнил: студент группы ПВ-41

Черенков Д. М.

Проверил: Осипов О.В.

Белгород 2017 г.

**Цель работы:** создание эффектов освещение и текстурированние простых геометрических объектов в OpenGL.

**Требования к программе**

1. Создать 3 источника направленного света и 3 источника точечного света.
2. Предоставить возможность пользователю перемещать точечный источник света и изменять цвет всех источников.
3. Предоставит возможность пользователю менять все параметры материала для объектов сцены.
4. Наложить на объекты текстуры (по желанию пользователя).
5. Нарисовать положение источника света в виде сферы.

**Скриншоты**



**Листинг программы**

**Form1.cs**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
**using** SharpGL;  
**using** SharpGL.SceneGraph.Primitives;  
**using** SharpGL\_6.figures;  
**using** SharpGL\_7.figures;  
  
**namespace** SharpGL\_7  
{  
 **public partial class** Form1 : Form  
 {  
 **public** Form1()  
 {  
 InitializeComponent();  
 }  
  
 **private readonly** FigureParallelepiped \_figureKub = **new** FigureParallelepiped();  
 **private readonly** FigureHummer **\_figureHummer1** = **new** FigureHummer();  
 **private readonly** FigureHummer **\_figureHummer2** = **new** FigureHummer();  
 **private readonly** FigureHummer **\_figureHummer3** = **new** FigureHummer();  
  
 **private float** \_angleX;  
 **private float** \_angleY;  
  
 **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** Point **\_currentLocation** = **new** Point(0, 0);  
  
 **private readonly float**[] \_globalAmbient = {0.0f, 0.0f, 0.0f, 1f};  
   
 **private float**[] **\_light0Scpecular** = {1.0f, 1.0f, 1.0f, 1.0f};  
 **private float**[] **\_light0Ambient** = {0.8f, 0.8f, 0.8f, 1f};  
 **private float**[] **\_light1Ambient** = {0.4f, 0.4f, 0.4f, 1f};  
 **private float**[] **\_light2Ambient** = {0.1f, 0.1f, 0.1f, 1f};  
 **private float**[] **\_light0Diffuse** = {0.5f, 0.5f, 0.5f, 0.5f};  
 **private float**[] \_light1Diffuse = {1.0f, 1.0f, 1.0f, 1.0f};  
   
 **private float**[] **\_l0pos** = {-0.1f + **x1**, 0f + **y1**, -4.0f, 1.0f};  
 **private float**[] **\_l1pos** = {0.2f + **x2**, 0f + **y2**, -4.0f, 1.0f};  
 **private float**[] **\_l2pos** = {0.2f + **x2**, 0f + **y2**, -4.0f, 1.0f};  
   
   
 **private float**[] **\_sRef** = {1.0f, 1.0f, 1.0f, 1.0f};  
 **private float**[] \_s0pos = {-0.1f + **x1**, 0f + **y1**, -4.0f, 1.0f};  
 **private float**[] **\_black** = {0f, 0f, 0f, 1f};  
  
 **public void** Init()  
 {  
 **try** {  
 **var** tbR = **int**.Parse(**textBoxR**.**Text**);  
 **var** tbG = **int**.Parse(**textBoxR**.**Text**);  
 **var** tbB = **int**.Parse(**textBoxR**.**Text**);  
  
 **\_l0pos** = **new**[] {-0.1f + **x1**, 0f + **y1**, -4.0f + **z1**, 1.0f};  
 **\_l1pos** = **new**[] {0.2f + **x2**, 0f + **y2**, -4.0f + **z2**, 1.0f};  
 **\_l2pos** = **new**[] {0.1f + **x3**, 0.2f + **y3**, -4.0f, 1.0f};  
  
 **\_sRef** = **new**[] {tbR / 255f, tbG / 255f, tbB / 255f, 1f};  
 }  
 **catch** (Exception e)  
 {  
 *// ignored* }  
 }  
   
 **private void** openGLControl1\_OpenGLDraw\_1(**object** sender, RenderEventArgs args)  
 {  
 Init();  
 **var** gl = **openGLControl1**.**OpenGL**;  
 gl.Clear(OpenGL.***GL\_COLOR\_BUFFER\_BIT*** | OpenGL.***GL\_DEPTH\_BUFFER\_BIT***);  
  
 gl.Enable(OpenGL.***GL\_LIGHTING***);  
 gl.ClearColor(0.5f, 0.5f, 0.5f, 0.5f);  
   
 gl.Enable(OpenGL.***GL\_COLOR\_MATERIAL***);  
 gl.Enable(OpenGL.***GL\_DEPTH\_TEST***);  
 gl.Enable(OpenGL.***GL\_TEXTURE\_2D***);  
   
 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.LoadIdentity();  
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.***GL\_POSITION***, **\_l0pos**);  
 **\_figureHummer1**.Draw(gl, **\_l0pos**[0], **\_l0pos**[1], **\_l0pos**[2], 0, 0, 1f, 0);  
 gl.LoadIdentity();  
   
 gl.Light(OpenGL.***GL\_LIGHT0***, OpenGL.***GL\_POSITION***, **\_l1pos**);  
 **\_figureHummer2**.Draw(gl, **\_l1pos**[0], **\_l1pos**[1], **\_l1pos**[2], 0, 0, 1f, 0);  
 gl.LoadIdentity();  
   
 gl.Light(OpenGL.***GL\_LIGHT2***, OpenGL.***GL\_POSITION***, **\_l2pos**);  
 **\_figureHummer3**.Draw(gl, **\_l2pos**[0], **\_l2pos**[1], **\_l2pos**[2], 0, 0, 1f, 0);  
 gl.LoadIdentity();  
   
 gl.Enable(OpenGL.***GL\_LIGHT0***);  
 gl.Enable(OpenGL.GL\_LIGHT1);  
 gl.Enable(OpenGL.***GL\_LIGHT2***);  
  
 gl.LightModel(OpenGL.***GL\_LIGHT\_MODEL\_AMBIENT***, **\_black**);  
   
 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***, **\_light0Diffuse**);  
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.***GL\_AMBIENT***, **\_light1Ambient**);  
 gl.Light(OpenGL.GL\_LIGHT1, OpenGL.***GL\_SPECULAR***, **\_light0Scpecular**);  
 gl.Light(OpenGL.***GL\_LIGHT2***, OpenGL.***GL\_DIFFUSE***, **\_light0Diffuse**);  
 gl.Light(OpenGL.***GL\_LIGHT2***, OpenGL.***GL\_AMBIENT***, **\_light2Ambient**);  
 gl.Light(OpenGL.***GL\_LIGHT2***, OpenGL.***GL\_SPECULAR***, **\_light0Scpecular**);  
  
   
 gl.LoadIdentity();gl.Translate(0, 0, -5f);  
 **var** teapot = **new** Teapot();gl.Color(0.5f, 0.5f, 0.1f);  
 teapot.Draw(gl, 14, 1, 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)  
 {  
 **z1** += e.**Delta** > 0 ? 0.01f : -0.01f;  
*// z2 += e.Delta > 0 ? 0.01f : -0.01f;* **base**.OnMouseWheel(e);  
 }  
   
 **private void** openGLControl1\_KeyDown(**object** sender, KeyEventArgs e)  
 {   
 **switch** (e.**KeyCode**)  
 {  
 **case** Keys.***Q***:  
 **if** (e.**Shift**)  
 {  
 **x1** += 0.05f;  
 }  
 **else** {  
 **x1** -= 0.05f;  
 }  
 **break**;  
 **case** Keys.***W***:  
 **if** (e.**Shift**)  
 {  
 **y1** += 0.05f;  
 }  
 **else** {  
 **y1** -= 0.05f;  
 }  
 **break**;  
 **case** Keys.***A***:  
 **if** (e.**Shift**)  
 {  
 **x2** += 0.05f;  
 }  
 **else** {  
 **x2** -= 0.05f;  
 }  
 **break**;  
 **case** Keys.***S***:  
 **if** (e.**Shift**)  
 {  
 **y2** += 0.05f;  
 }  
 **else** {  
 **y2** -= 0.05f;  
 }  
 **break**;  
 **case** Keys.***Z***:  
 **if** (e.**Shift**)  
 {  
 **x3** += 0.05f;  
 }  
 **else** {  
 **x3** -= 0.05f;  
 }  
 **break**;  
 **case** Keys.***X***:  
 **if** (e.**Shift**)  
 {  
 **y3** += 0.05f;  
 }  
 **else** {  
 **y3** -= 0.05f;  
 }  
 **break**;  
 }  
 }  
  
 }  
}

**FigureSphere.cs**

**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;  
 }  
 }  
}