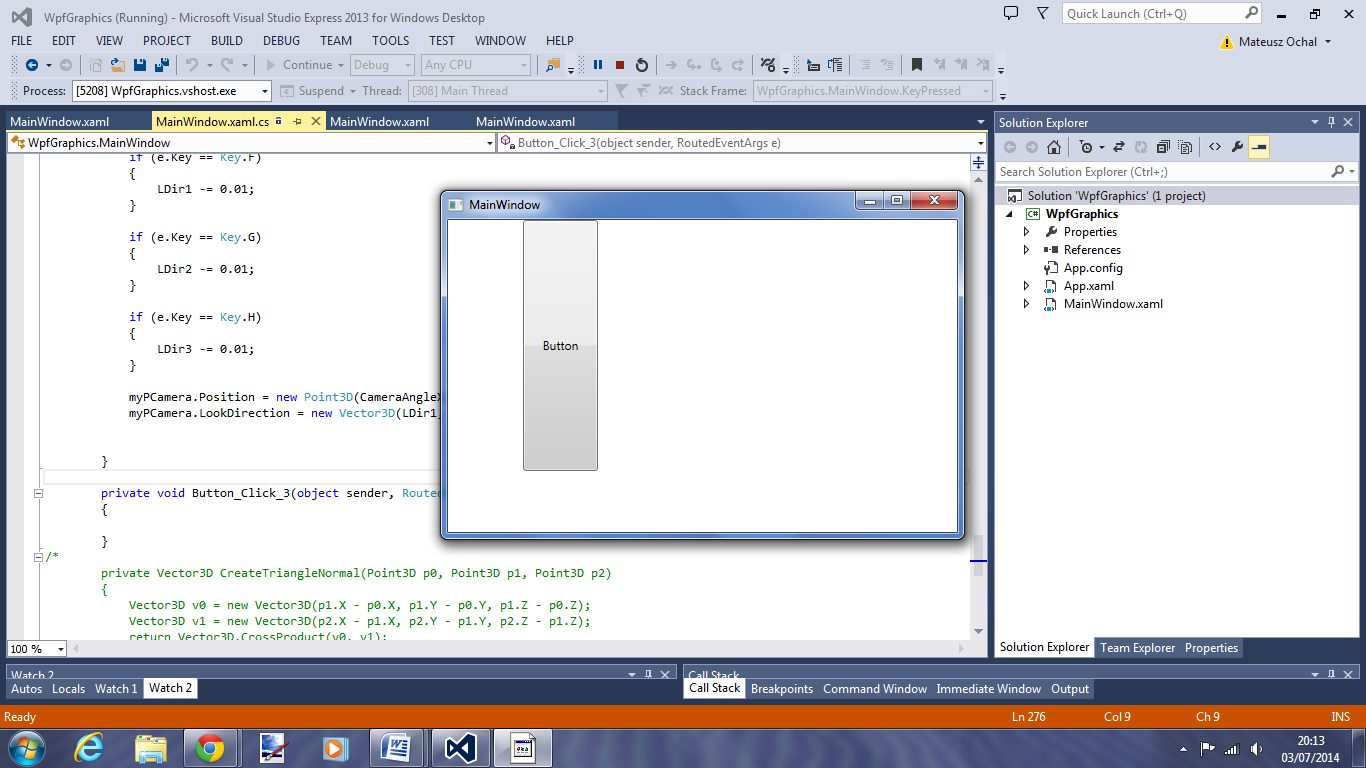
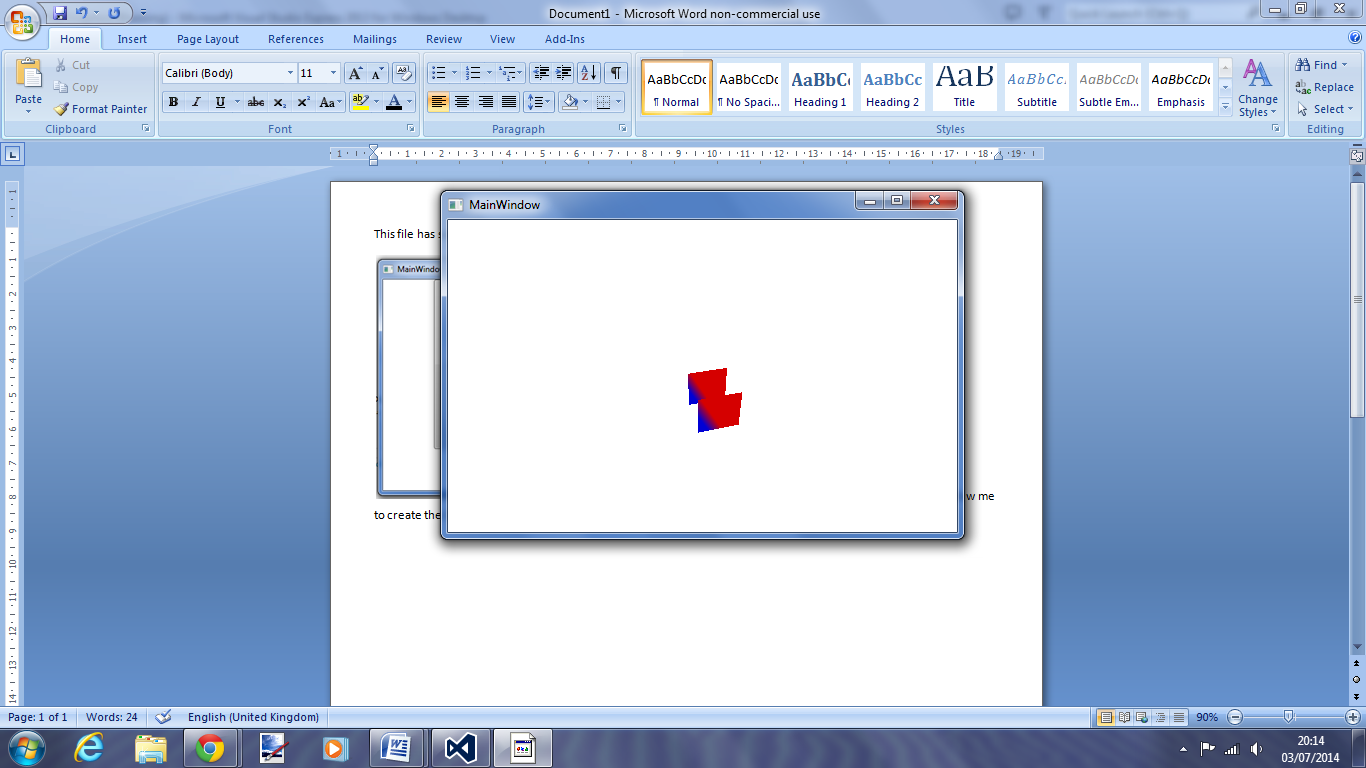
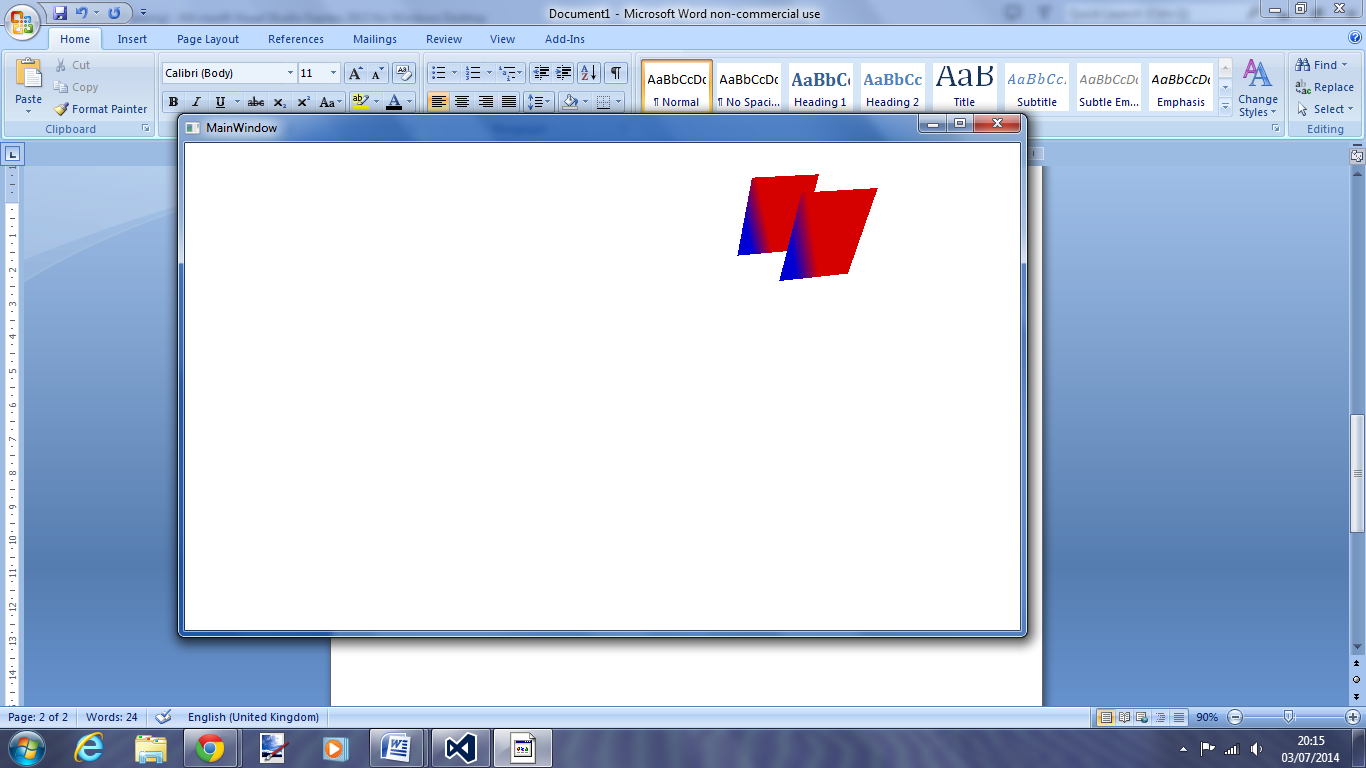
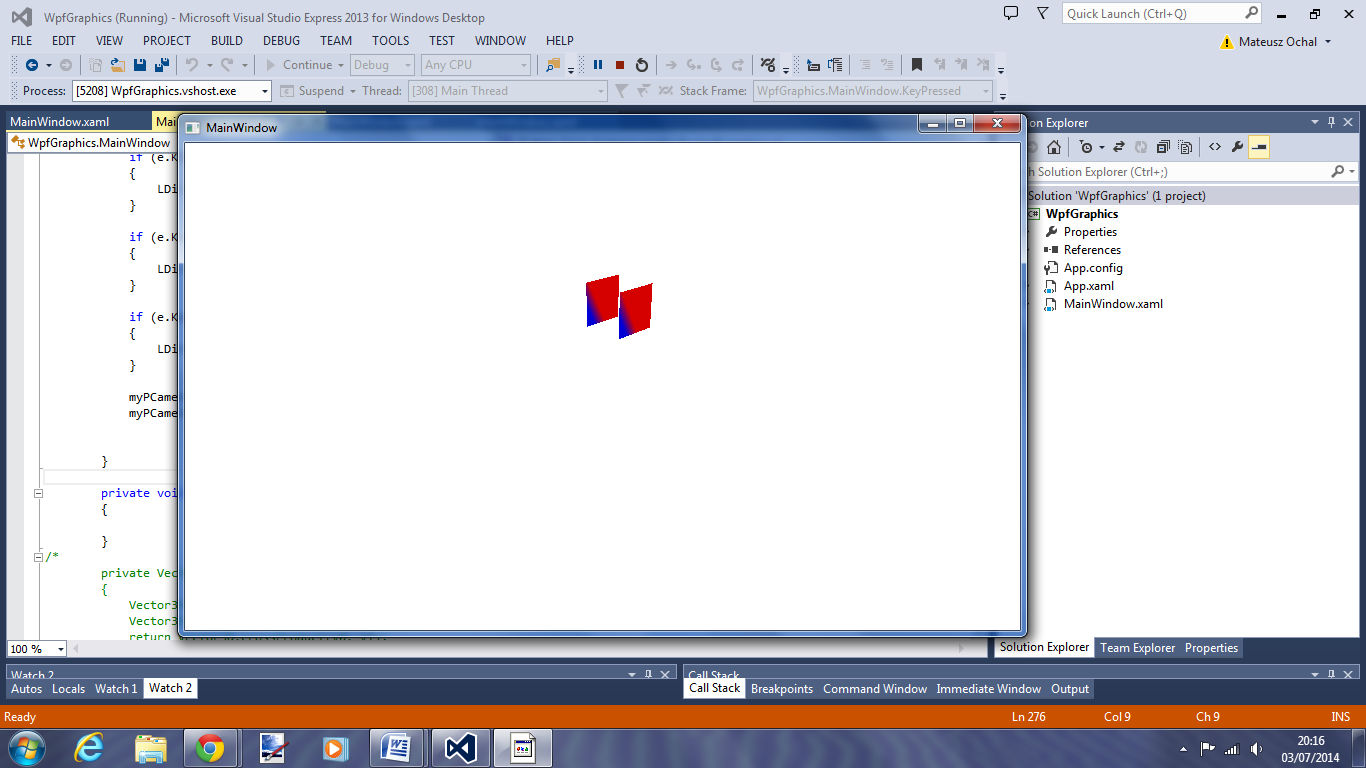
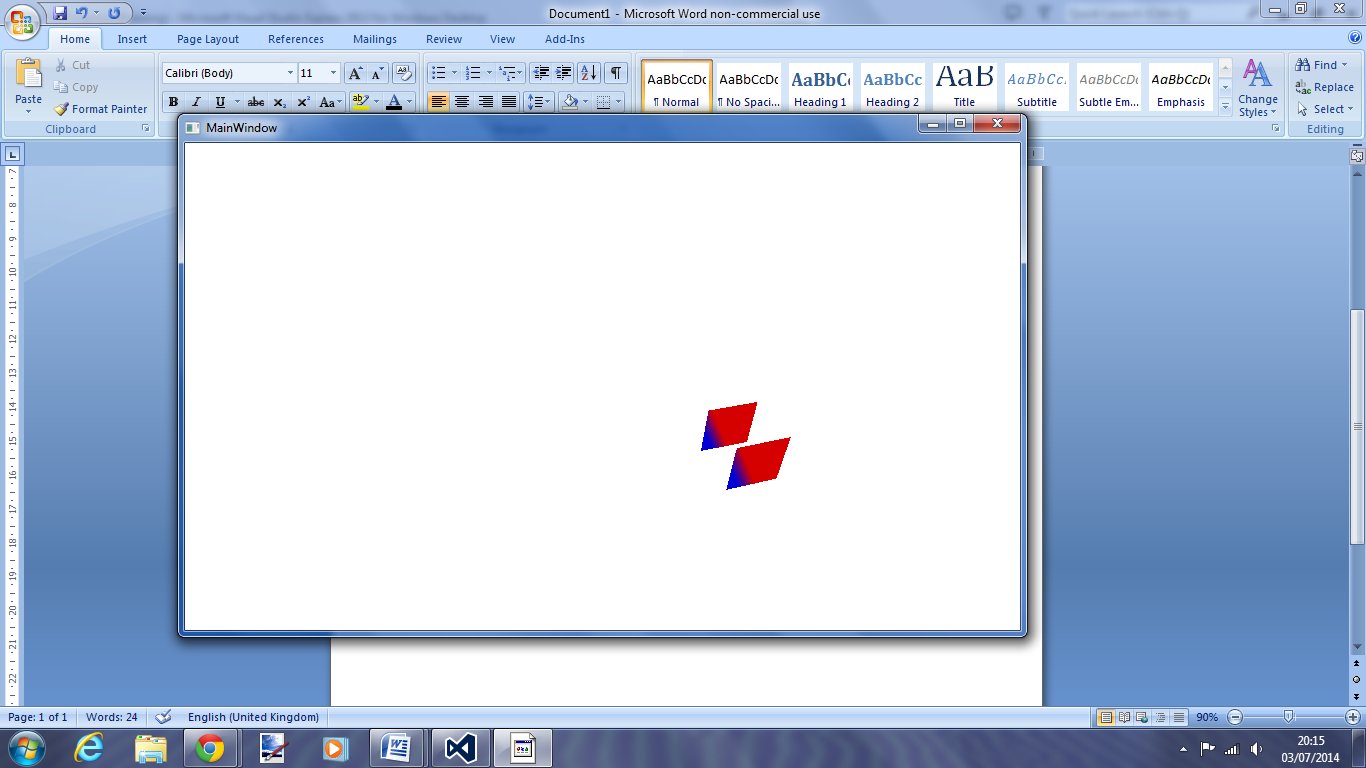
This file has screen shots of the WFP program:

 First very basic screen with a button to allow me to create the 3D object

Here is what is displayed after the button is clicked



The following screen shots are just to show how the camera direction and position change as I click the keys on the key board.

Here is the code I wrote with the example code used from this website: <http://msdn.microsoft.com/en-us/library/system.windows.controls.viewport3d.aspx>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Media.Media3D;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfGraphics

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Viewport3D myViewport3D;

Model3DGroup myModel3DGroup;

GeometryModel3D myGeometryModel;

ModelVisual3D myModelVisual3D;

PerspectiveCamera myPCamera;

DirectionalLight myDirectionalLight;

double CameraAngleX = 4;

double CameraAngleY = 4;

double CameraAngleZ = 4;

double LDir1 = -1;

double LDir2 = -1;

double LDir3 = -1;

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

myViewport3D = new Viewport3D();

myModel3DGroup = new Model3DGroup();

myGeometryModel = new GeometryModel3D();

myModelVisual3D = new ModelVisual3D();

// Defines the camera used to view the 3D object. In order to view the 3D object,

// the camera must be positioned and pointed such that the object is within view

// of the camera.

myPCamera = new PerspectiveCamera();

// Specify where in the 3D scene the camera is.

//myPCamera.Position = new Point3D(4, 4, 4); //2

myPCamera.Position = new Point3D(CameraAngleX, CameraAngleY, CameraAngleZ);

// Specify the direction that the camera is pointing.

myPCamera.LookDirection = new Vector3D(LDir1, LDir2, LDir3);

// Define camera's horizontal field of view in degrees.

myPCamera.FieldOfView = 90; //60

// Asign the camera to the viewport

myViewport3D.Camera = myPCamera;

// Define the lights cast in the scene. Without light, the 3D object cannot

// be seen. Note: to illuminate an object from additional directions, create

// additional lights.

myDirectionalLight = new DirectionalLight();

myDirectionalLight.Color = Colors.White;

myDirectionalLight.Direction = new Vector3D(-0.61, -0.5, -0.61);

myModel3DGroup.Children.Add(myDirectionalLight);

// The geometry specifes the shape of the 3D plane. In this sample, a flat sheet

// is created.

MeshGeometry3D myMeshGeometry3D = new MeshGeometry3D();

/\*\*/

// Create a collection of normal vectors for the MeshGeometry3D.

Vector3DCollection myNormalCollection = new Vector3DCollection();

/\*myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myNormalCollection.Add(new Vector3D(0, 0, 1));

myMeshGeometry3D.Normals = myNormalCollection;

/\*\*/

// Create a collection of vertex positions for the MeshGeometry3D.

Point3DCollection myPositionCollection = new Point3DCollection();

myPositionCollection.Add(new Point3D(-0.5, -0.5, 0.5));

myPositionCollection.Add(new Point3D(0.5, -0.5, 0.5));

myPositionCollection.Add(new Point3D(0.5, 0.5, 0.5));

myPositionCollection.Add(new Point3D(0.5, 0.5, 0.5));

myPositionCollection.Add(new Point3D(-0.5, 0.5, 0.5));

myPositionCollection.Add(new Point3D(-0.5, -0.5, 0.5));

myPositionCollection.Add(new Point3D(-0.5, -0.5, 1.5));

myPositionCollection.Add(new Point3D(0.5, -0.5, 1.5));

myPositionCollection.Add(new Point3D(0.5, 0.5, 1.5));

myPositionCollection.Add(new Point3D(0.5, 0.5, 1.5));

myPositionCollection.Add(new Point3D(-0.5, 0.5, 1.5));

myPositionCollection.Add(new Point3D(-0.5, -0.5, 1.5));

myMeshGeometry3D.Positions = myPositionCollection;

// Create a collection of texture coordinates for the MeshGeometry3D.

PointCollection myTextureCoordinatesCollection = new PointCollection();

myTextureCoordinatesCollection.Add(new Point(0, 0));

myTextureCoordinatesCollection.Add(new Point(1, 0));

myTextureCoordinatesCollection.Add(new Point(1, 1));

myTextureCoordinatesCollection.Add(new Point(1, 1));

myTextureCoordinatesCollection.Add(new Point(0, 1));

myTextureCoordinatesCollection.Add(new Point(0, 0));

myTextureCoordinatesCollection.Add(new Point(0, 0));

myTextureCoordinatesCollection.Add(new Point(1, 0));

myTextureCoordinatesCollection.Add(new Point(1, 1));

myTextureCoordinatesCollection.Add(new Point(1, 1));

myTextureCoordinatesCollection.Add(new Point(0, 1));

myTextureCoordinatesCollection.Add(new Point(0, 0));

myMeshGeometry3D.TextureCoordinates = myTextureCoordinatesCollection;

// Create a collection of triangle indices for the MeshGeometry3D.

Int32Collection myTriangleIndicesCollection = new Int32Collection();

myTriangleIndicesCollection.Add(0);

myTriangleIndicesCollection.Add(1);

myTriangleIndicesCollection.Add(2);

myTriangleIndicesCollection.Add(3);

myTriangleIndicesCollection.Add(4);

myTriangleIndicesCollection.Add(5);

myTriangleIndicesCollection.Add(6);

myTriangleIndicesCollection.Add(7);

myTriangleIndicesCollection.Add(8);

myTriangleIndicesCollection.Add(9);

myTriangleIndicesCollection.Add(10);

myTriangleIndicesCollection.Add(11);

myMeshGeometry3D.TriangleIndices = myTriangleIndicesCollection;

// Apply the mesh to the geometry model.

myGeometryModel.Geometry = myMeshGeometry3D;

// The material specifies the material applied to the 3D object. In this sample a

// linear gradient covers the surface of the 3D object.

// Create a horizontal linear gradient with four stops.

LinearGradientBrush myHorizontalGradient = new LinearGradientBrush();

myHorizontalGradient.StartPoint = new Point(0, 0.5);

//myHorizontalGradient.EndPoint = new Point(1, 0.5);

myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Blue, 0.0));

myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Red, 0.25));

//myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Blue, 0.75));

//myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.LimeGreen, 1.0));

// Define material and apply to the mesh geometries.

DiffuseMaterial myMaterial = new DiffuseMaterial(myHorizontalGradient);

myGeometryModel.Material = myMaterial;

// Apply a transform to the object. In this sample, a rotation transform is applied,

// rendering the 3D object rotated.

RotateTransform3D myRotateTransform3D = new RotateTransform3D();

AxisAngleRotation3D myAxisAngleRotation3d = new AxisAngleRotation3D();

myAxisAngleRotation3d.Axis = new Vector3D(0, 3, 0);

myAxisAngleRotation3d.Angle = 60;

myRotateTransform3D.Rotation = myAxisAngleRotation3d;

myGeometryModel.Transform = myRotateTransform3D;

// Add the geometry model to the model group.

myModel3DGroup.Children.Add(myGeometryModel);

// Add the group of models to the ModelVisual3d.

myModelVisual3D.Content = myModel3DGroup;

//

myViewport3D.Children.Add(myModelVisual3D);

// Apply the viewport to the page so it will be rendered.

this.Content = myViewport3D;

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

}

private void KeyPressed(object sender, KeyEventArgs e)

{

if (e.Key == Key.A)

{

//myModel3DGroup.Children.Remove(myDirectionalLight);

CameraAngleX += - 0.1;

//myModel3DGroup.Children.Add(myDirectionalLight);

}

if (e.Key == Key.Q)

{

//myModel3DGroup.Children.Remove(myDirectionalLight);

CameraAngleX += 0.1;

//myModel3DGroup.Children.Add(myDirectionalLight);

}

if (e.Key == Key.S)

{

CameraAngleY += -0.1;

}

if (e.Key == Key.W)

{

CameraAngleY += 0.1;

}

if (e.Key == Key.D)

{

CameraAngleZ += -0.1;

}

if (e.Key == Key.E)

{

CameraAngleZ += 0.1;

}

//----------------------------

if (e.Key == Key.R)

{

LDir1 += 0.01;

}

if (e.Key == Key.T)

{

LDir2 += 0.01;

}

if (e.Key == Key.Y)

{

LDir3 += 0.01;

}

if (e.Key == Key.F)

{

LDir1 -= 0.01;

}

if (e.Key == Key.G)

{

LDir2 -= 0.01;

}

if (e.Key == Key.H)

{

LDir3 -= 0.01;

}

myPCamera.Position = new Point3D(CameraAngleX, CameraAngleY, CameraAngleZ);

myPCamera.LookDirection = new Vector3D(LDir1, LDir2, LDir3);

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

}

/\*

private Vector3D CreateTriangleNormal(Point3D p0, Point3D p1, Point3D p2)

{

Vector3D v0 = new Vector3D(p1.X - p0.X, p1.Y - p0.Y, p1.Z - p0.Z);

Vector3D v1 = new Vector3D(p2.X - p1.X, p2.Y - p1.Y, p2.Z - p1.Z);

return Vector3D.CrossProduct(v0, v1);

}\*/

}

}