

Course 3D_WPF: 3D-Computer Graphics with C# + WPF

Chapter C4: The complete code of Sphere



Copyright © by V. Miszalok, last update: 2010-01-08

- ↓ [Preliminaries](#)
- ↓ [MainWindow.xaml](#)
- ↓ [MainWindow.xaml.cs](#)

Preliminaries

Guidance for **Visual C# 2010 Express**:

- 1) Main Menu after start of Visual C# 2010 Express: File → New Project... → WPF Application → Name: sphere1 → OK.
- 2) File → Save All → C:\temp. Check whether the project arrived in C:\temp\sphere1\!
- 3) Download and store 5 images to directory C:\temp\sphere1\ :
 - 3.1 www.miszalok.de/C_3D_WPF/C4_Sphere/Images/mesh.bmp
 - 3.2 www.miszalok.de/C_3D_WPF/C4_Sphere/Images/randomStripes.bmp
 - 3.3 www.miszalok.de/C_3D_WPF/C4_Sphere/Images/earth.bmp
 - 3.4 www.miszalok.de/C_3D_WPF/C4_Sphere/Images/lena256.bmp
 - 3.5 www.miszalok.de/C_3D_WPF/C4_Sphere/Images/boom.gif

Check whether these 5 images are correctly stored in directory C:\temp\sphere1\ and whether they carry their proper extensions *.bmp or *.gif.

MainWindow.xaml

Replace the default code of MainWindow.xaml by the following code:

```
<Window x:Class="sphere1.MainWindow" x:Name="window"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="sphere1" Width="400" Height="650">
<Window.Resources>
  <BitmapImage x:Key="meshImage" UriSource="C:\temp\sphere1\mesh.bmp" />
  <BitmapImage x:Key="ballImage" UriSource="C:\temp\sphere1\randomStripes.bmp" />
  <BitmapImage x:Key="earthImage" UriSource="C:\temp\sphere1\earth.bmp" />
  <BitmapImage x:Key="faceImage" UriSource="C:\temp\sphere1\lena256.bmp" />
  <BitmapImage x:Key="boomImage" UriSource="C:\temp\sphere1\boom.gif" />
  <!--alternative (but slower) image sources:
  <BitmapImage x:Key="meshImage"
    UriSource="http://www.miszalok.de/C_3D_WPF/C4_Sphere/Images/mesh.bmp" />
  <BitmapImage x:Key="ballImage"
    UriSource="http://www.miszalok.de/C_3D_WPF/C4_Sphere/Images/randomStripes.bmp" />
  <BitmapImage x:Key="earthImage"
    UriSource="http://www.miszalok.de/C_3D_WPF/C4_Sphere/Images/earth.bmp" />
  <BitmapImage x:Key="faceImage"
    UriSource="http://www.miszalok.de/C_3D_WPF/C4_Sphere/Images/lena256.bmp" />
  <BitmapImage x:Key="boomImage"
    UriSource="http://www.miszalok.de/C_3D_WPF/C4_Sphere/Images/boom.gif" />
  -->
  <ImageBrush x:Key="meshBrush" ImageSource="{ StaticResource meshImage }"/>
  <ImageBrush x:Key="ballBrush" ImageSource="{ StaticResource ballImage }"/>
  <ImageBrush x:Key="earthBrush" ImageSource="{ StaticResource earthImage }"/>
  <ImageBrush x:Key="faceBrush" ImageSource="{ StaticResource faceImage }"/>
  <SolidColorBrush x:Key="solidColorBrush" Color="Red"/>
  <DiffuseMaterial x:Key="backMaterial" Brush="{ StaticResource SolidColorBrush }"/>
```

```

<Style TargetType="{x:Type StackPanel}">
  <Setter Property="Orientation" Value="Horizontal"/>
  <Setter Property="HorizontalAlignment" Value="Center"/>
  <Setter Property="Margin" Value="5 5 5 5"/>
</Style>
<Style TargetType="{x:Type RadioButton}">
  <Setter Property="Width" Value="70"/>
  <Setter Property="FontSize" Value="9"/>
  <EventSetter Event="Click" Handler="on_radioButton_clicked"/>
</Style>
<Style TargetType="{x:Type CheckBox}">
  <Setter Property="Width" Value="70"/>
  <Setter Property="FontSize" Value="9"/>
  <EventSetter Event="Click" Handler="on_checkbox_clicked"/>
</Style>
<Style TargetType="{x:Type DockPanel}">
  <Setter Property="Background" Value="LightGray"/>
  <Setter Property="LastChildFill" Value="True"/>
  <Setter Property="Margin" Value="5 1 5 1"/>
</Style>
<Style TargetType="{x:Type Label}">
  <Setter Property="FontSize" Value="10"/>
  <Setter Property="Width" Value="90"/>
  <Setter Property="DockPanel.Dock" Value="Left"/>
</Style>
<Style TargetType="{x:Type Slider}">
  <Setter Property="HorizontalAlignment" Value="Center"/>
  <Setter Property="Width" Value="200"/>
  <EventSetter Event="ValueChanged" Handler="on_slider_value_changed"/>
</Style>
<Style TargetType="{x:Type TextBox}">
  <Setter Property="DockPanel.Dock" Value="Right"/>
  <Setter Property="Width" Value="35"/>
</Style>
<Style TargetType="{x:Type Button}">
  <Setter Property="Width" Value="60"/>
  <Setter Property="FontSize" Value="8"/>
  <EventSetter Event="Click" Handler="on_button_clicked"/>
</Style>
</Window.Resources>

<StackPanel Orientation="Vertical" Margin="0 0 0 0">
  <StackPanel>
    <RadioButton x:Name="mesh" Content="Mesh" IsChecked="True"/>
    <RadioButton x:Name="ball" Content="BeachBall"/>
    <RadioButton x:Name="earth" Content="Earth"/>
    <RadioButton x:Name="face" Content="Face"/>
  </StackPanel>
  <StackPanel Margin="5,5,5,1">
    <CheckBox x:Name="X_Rotate" Content="X-Rotate"/>
    <CheckBox x:Name="Y_Rotate" Content="Y-Rotate"/>
    <CheckBox x:Name="Z_Rotate" Content="Z-Rotate"/>
  </StackPanel>
</StackPanel/>
<DockPanel>
  <Label Content="No. Latitudes"/>
  <TextBox x:Name="latitudes_slider_textBox" DockPanel.Dock="Right"/>
  <Slider x:Name="latitudes_slider" Minimum="3" Maximum="100" Value="30"/>
</DockPanel>
<DockPanel>
  <Label Content="No. Longitudes"/>
  <TextBox x:Name="longitudes_slider_textBox" DockPanel.Dock="Right"/>
  <Slider x:Name="longitudes_slider" Minimum="3" Maximum="100" Value="30"/>
</DockPanel>
<DockPanel>
  <Label Content="Light Direction"/>
  <TextBox x:Name="light_move_slider_textBox" DockPanel.Dock="Right"/>
  <Slider x:Name="light_move_slider" Minimum="-180" Maximum="180" Value="45"/>
</DockPanel>
<DockPanel>
  <Label Content="Quake Magnitude"/>
  <TextBox x:Name="quake_magnitude_slider_textBox" DockPanel.Dock="Right"/>
  <Slider x:Name="quake_magnitude_slider" Minimum="1" Maximum="10" Value="5"/>
</DockPanel>

```

```

<StackPanel>
  <Button x:Name="slices" Content="Cut off Slices" /><Label Width="10"/>
  <Button x:Name="north" Content="Cut from North" /><Label Width="10"/>
  <Button x:Name="south" Content="Cut from South" /><Label Width="10"/>
  <Button x:Name="equator" Content="Cut from Equator"/><Label Width="10"/>
  <Button x:Name="stacks" Content="Cut off Stacks" />
</StackPanel>
<StackPanel>
  <Button x:Name="earthquake" Content="Earthquake" FontSize="10"/>
  <Label Width="10"/>
  <Button x:Name="boom" Height="40"><Image Source="{StaticResource boomImage}"/></Button>
  <Label Width="10"/>
  <Button x:Name="reset" Content="Reset" FontSize="10"/>
</StackPanel>
<!--Viewport3D is a drawing canvas which resizes its Content automatically-->
<Viewport3D x:Name="viewport">
  <Viewport3D.Camera>
    <PerspectiveCamera x:Name="perspectiveCamera"
      Position=" 0 0 3" LookDirection=" 0 0 -1" UpDirection=" 0 1 0"/>
  </Viewport3D.Camera>
  <!--Any 3D-content must be packed in a ModelVisual3D-object-->
  <ModelVisual3D>
    <ModelVisual3D.Content>
      <!--Only one Content is allowed. Thus we use a Model3DGroup as envelope for our
      two lights and all further GeometryModel3Ds.-->
      <Model3DGroup x:Name="model3DGroup">
        <AmbientLight Color="#444444"/>
        <DirectionalLight x:Name="directionalLight" Color="#ffffff" Direction="-1 -1 -1" />
        <!--A lot of GeometryModel3Ds will be inserted here.-->
      </Model3DGroup>
    </ModelVisual3D.Content>
  </ModelVisual3D>
</Viewport3D>
</StackPanel><!--end of the uppermost StackPanel which contains everything-->
</Window>

```

MainWindow.xaml.cs

Replace the default code of MainWindow.xaml.cs by the following code:

```

using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Media.Media3D;

namespace sphere1
{
  public partial class MainWindow : Window
  {
    private const int maxLongitudes = 100; //maximum
    private const int maxLatitudes = 100; //maximum
    private Point3D[,] position = new Point3D[maxLongitudes+1,maxLatitudes];
    private Point [,] texture = new Point [maxLongitudes+1,maxLatitudes];
    private DiffuseMaterial[] frontMaterial = new DiffuseMaterial[maxLatitudes-1];
    private int longitudes; //actual <= maximum
    private int latitudes; //actual <= maximum
    private int emptySlices = 0; //cuts slices out of the apple
    private int IndexOfFirstGeometryModel3DInModel3DGroup; //= no of lights = 2
    private Matrix3D matrix = Matrix3D.Identity;
    private MatrixTransform3D matrixTransform3D;
    private Quaternion qX = new Quaternion( new Vector3D(1,0,0), 1 ); //rotations around X-axis
    private Quaternion qY = new Quaternion( new Vector3D(0,1,0), 1 ); //rotations around Y-axis
    private Quaternion qZ = new Quaternion( new Vector3D(0,0,1), 1 ); //rotations around Z-axis
    private System.Windows.Threading.DispatcherTimer timer =
      new System.Windows.Threading.DispatcherTimer();
  }
}

```

```

public MainWindow() //constructor
{
    InitializeComponent();
    IndexOfFirstGeometryModel3DInModel3DGroup = model3DGroup.Children.Count;
    longitudes = Convert.ToInt32( longitudes_slider.Value );
    latitudes = Convert.ToInt32( latitudes_slider.Value );
    longitudes_slider_textBox.Text = Convert.ToInt32( longitudes ).ToString();
    latitudes_slider_textBox.Text = Convert.ToInt32( latitudes ).ToString();
    quake_magnitude_slider_textBox.Text = Convert.ToInt32( quake_magnitude_slider.Value ).ToString();
    light_move_slider_textBox.Text = Convert.ToInt32( light_move_slider.Value ).ToString();
    GenerateImageMaterials();
    GenerateSphere( longitudes, latitudes );
    GenerateAllCylinders();
    timer.Interval = TimeSpan.FromMilliseconds( 1 );
    timer.Tick += TimerOnTick;
}

protected override void OnRenderSizeChanged( SizeChangedEventArgs sizeInfo )
{
    try { viewport.Width = viewport.Height = window.ActualWidth; } catch {}
}

private void GenerateImageMaterials()
{
    ImageBrush imageBrush;
    double flatThickness = 1.0/(latitudes-1);
    double minus = (double)(longitudes-emptySlices);
    if ( (bool)mesh.IsChecked )
    {
        imageBrush = (ImageBrush)Resources["meshBrush"];
        imageBrush.Viewport = new Rect( 0, 0, 1.0/minus, 1 );
        imageBrush.TileMode = TileMode.Tile;
        for ( int lat=0; lat < latitudes-1; lat++ )
            frontMaterial[lat] = new DiffuseMaterial( imageBrush );
    }
    else if ( (bool)ball.IsChecked )
    {
        imageBrush = (ImageBrush)Resources["ballBrush"];
        imageBrush.Viewbox = new Rect( 0, 0, minus/maxLongitudes, flatThickness );
        for ( int lat=0; lat < latitudes-1; lat++ )
            frontMaterial[lat] = new DiffuseMaterial( imageBrush );
    }
    else if ( (bool)earth.IsChecked )
    {
        for ( int lat=0; lat < latitudes-1; lat++ )
        {
            imageBrush = new ImageBrush( (BitmapImage)Resources["earthImage"] );
            imageBrush.Viewbox = new Rect( 0, lat*flatThickness, minus/longitudes, flatThickness );
            frontMaterial[lat] = new DiffuseMaterial( imageBrush );
        }
    }
    else if ( (bool)face.IsChecked )
    {
        for ( int lat=0; lat < latitudes-1; lat++ )
        {
            imageBrush = new ImageBrush( (BitmapImage)Resources["faceImage"] );
            imageBrush.Viewbox = new Rect( 0, lat*flatThickness, minus/longitudes, flatThickness );
            frontMaterial[lat] = new DiffuseMaterial( imageBrush );
        }
    }
}

private void GenerateSphere( int longitudes, int latitudes )
{
    double latitudeArcusIncrement = Math.PI / (latitudes-1);
    double longitudeArcusIncrement = 2.0*Math.PI / longitudes;
    for ( int lat=0; lat < latitudes; lat++ )
    {
        double latitudeArcus = lat * latitudeArcusIncrement;
        double radius = Math.Sin( latitudeArcus );
        //if ( lat == latitudes/2 ) radius *= 1.3;
        double y = Math.Cos( latitudeArcus );
        double textureY = (double)lat / (latitudes-1);
        for ( int lon=0; lon <= longitudes; lon++ )
        {
            double longitudeArcus = lon * longitudeArcusIncrement;
            position[lon,lat].X = radius * Math.Cos( longitudeArcus );
            position[lon,lat].Y = y;
            position[lon,lat].Z = -radius * Math.Sin( longitudeArcus );
            texture [lon,lat].X = (double)lon / longitudes;
            texture [lon,lat].Y = textureY;
        }
    }
}

```

```

private void GenerateAllCylinders()
{ //At first delete all existing flats beginning with the last one
  for ( int i=model3DGroup.Children.Count-1;
        i >= IndexOfFirstGeometryModel3DInModel3DGroup; i-- )
    model3DGroup.Children.Remove( (GeometryModel3D)model3DGroup.Children[i] );
  for ( int lat=0; lat < latitudes-1; lat++ )
  { GeometryModel3D geometryModel3D = new GeometryModel3D();
    geometryModel3D.Geometry = GenerateCylinder( lat );
    geometryModel3D.Material = frontMaterial[lat];
    geometryModel3D.BackMaterial = (DiffuseMaterial)Resources["backMaterial"];
    model3DGroup.Children.Add( geometryModel3D );
  }
}

private MeshGeometry3D GenerateCylinder( int lat )
{ MeshGeometry3D meshGeometry3D = new MeshGeometry3D();
  for ( int lon = 0; lon <= longitudes - emptySlices; lon++ )
    //create a zigzagging point collection
    { Point3D p0 = position[lon,lat ] ; //on the ceiling
      Point3D p1 = position[lon,lat+1]; //on the floor
      meshGeometry3D.Positions.Add( p0 ); //on the ceiling
      meshGeometry3D.Positions.Add( p1 ); //on the floor
      meshGeometry3DNormals.Add( (Vector3D)p0 ); //ceiling normal
      meshGeometry3DNormals.Add( (Vector3D)p1 ); //floor normal
      meshGeometry3D.TextureCoordinates.Add( texture[lon,lat ] ); //on the ceiling
      meshGeometry3D.TextureCoordinates.Add( texture[lon,lat+1] ); //on the floor
    }
  for ( int lon = 1; lon < meshGeometry3D.Positions.Count - 2; lon+=2)
  { //first triangle = left upper part of a rectangle
    meshGeometry3D.TriangleIndices.Add( lon-1 ); //left upper point
    meshGeometry3D.TriangleIndices.Add( lon ); //left lower point
    meshGeometry3D.TriangleIndices.Add( lon+1 ); //right upper point
    //second triangle = right lower part of the rectangle
    meshGeometry3D.TriangleIndices.Add( lon+1 ); //right upper point
    meshGeometry3D.TriangleIndices.Add( lon ); //left lower point
    meshGeometry3D.TriangleIndices.Add( lon+2 ); //right lower point
  }
  return meshGeometry3D;
}

private void on_radioButton_clicked(object sender, EventArgs e)
{ GenerateImageMaterials();
  for ( int i=IndexOfFirstGeometryModel3DInModel3DGroup, j=0;
        i < model3DGroup.Children.Count; i++, j++ )
    ((GeometryModel3D)model3DGroup.Children[i]).Material = frontMaterial[j];
}

private void on_checkbox_clicked(object sender, EventArgs e)
{ timer.Stop();
  matrix = Matrix3D.Identity;
  if ( (bool)X_Rotate.IsChecked ) timer.Start();
  else if ( (bool)Y_Rotate.IsChecked ) timer.Start();
  else if ( (bool)Z_Rotate.IsChecked ) timer.Start();
}

private void TimerOnTick( Object sender, EventArgs args )
{ if ( (bool)X_Rotate.IsChecked ) matrix.Rotate( qX );
  if ( (bool)Y_Rotate.IsChecked ) matrix.Rotate( qY );
  if ( (bool)Z_Rotate.IsChecked ) matrix.Rotate( qZ );
  matrixTransform3D = new MatrixTransform3D( matrix );
  for ( int i=IndexOfFirstGeometryModel3DInModel3DGroup;
        i < model3DGroup.Children.Count; i++ )
    ((GeometryModel3D)model3DGroup.Children[i]).Transform = matrixTransform3D;
}

```

```

private void on_slider_value_changed(object sender, EventArgs e)
{ switch( ((Slider)sender).Name )
  { case "latitudes_slider":
    latitudes = Convert.ToInt32( latitudes_slider.Value );
    latitudes_slider_textBox.Text = Convert.ToInt32( latitudes_slider.Value ).ToString();
    GenerateImageMaterials();
    GenerateSphere( longitudes, latitudes );
    GenerateAllCylinders();
    break;
  case "longitudes_slider":
    longitudes = Convert.ToInt32( longitudes_slider.Value );
    longitudes_slider_textBox.Text = Convert.ToInt32( longitudes_slider.Value ).ToString();
    if ( longitudes < emptySlices ) emptySlices = 0;
    if ( (bool)mesh.IsChecked | (bool)ball.IsChecked) GenerateImageMaterials();
    GenerateSphere( longitudes, latitudes );
    GenerateAllCylinders();
    break;
  case "light_move_slider":
    double arcus = 2.0*Math.PI*light_move_slider.Value / 360;
    Vector3D v = new Vector3D();
    v.X = -3.0 * Math.Sin( arcus );
    v.Y = 0;
    v.Z = -3.0 * Math.Cos( arcus );
    directionalLight.Direction = v;
    light_move_slider_textBox.Text = Convert.ToInt32( light_move_slider.Value ).ToString();
    break;
  case "quake_magnitude_slider":
    quake_magnitude_slider_textBox.Text =
      Convert.ToInt32( quake_magnitude_slider.Value ).ToString();
    break;
  }
}

```

```

private void on_button_clicked( object sender, EventArgs e )
{ switch( ((Button)sender).Name )
  { case "slices":
    emptySlices++;
    if ( emptySlices > longitudes ) emptySlices = 0;
    GenerateImageMaterials();
    GenerateAllCylinders();
    perspectiveCamera.Position      = new Point3D ( 3,0, 1 );
    perspectiveCamera.LookDirection = new Vector3D( -3,0, -1 );
    break;
  case "north":
    if ( model3DGroup.Children.Count > IndexOfFirstGeometryModel3DInModel3DGroup )
      model3DGroup.Children.RemoveAt( IndexOfFirstGeometryModel3DInModel3DGroup );
    break;
  case "south":
    if ( model3DGroup.Children.Count > IndexOfFirstGeometryModel3DInModel3DGroup )
      model3DGroup.Children.RemoveAt( model3DGroup.Children.Count-1 );
    break;
  case "equator":
    if ( model3DGroup.Children.Count > IndexOfFirstGeometryModel3DInModel3DGroup )
    { int mid = (model3DGroup.Children.Count -
      IndexOfFirstGeometryModel3DInModel3DGroup)/2;
      mid += IndexOfFirstGeometryModel3DInModel3DGroup;
      model3DGroup.Children.RemoveAt( mid );
    }
    break;
  case "stacks":
    for ( int i=model3DGroup.Children.Count-1;
          i >= IndexOfFirstGeometryModel3DInModel3DGroup; i-=2 )
      model3DGroup.Children.RemoveAt( i );
    break;
  case "boom":
    Point3D cp = perspectiveCamera.Position;
    if ( cp.Z > 3 ) break; //already boomed
    double amplitude = 3; //earthquake magnitude: 30
    cp.Z += 4; //step back from explosion
    perspectiveCamera.Position = cp;
    goto earthquake;
  case "earthquake":
    amplitude = quake_magnitude_slider.Value * 0.01;
earthquake: Point3D p = new Point3D();
    double ah = amplitude / 2.0;
    Random r = new Random();
    for ( int i=IndexOfFirstGeometryModel3DInModel3DGroup+2;
          i < model3DGroup.Children.Count-2; i++ )
    { GeometryModel3D geometryModel3D = (GeometryModel3D)model3DGroup.Children[i];
      MeshGeometry3D meshGeometry3D = (MeshGeometry3D)geometryModel3D.Geometry;
      for ( int j=0; j < meshGeometry3D.Positions.Count; j++ )
      { p = meshGeometry3D.Positions[j];
        p.Z += - ah + amplitude*r.NextDouble();
        p.Y += - ah + amplitude*r.NextDouble();
        p.X += - ah + amplitude*r.NextDouble();
        meshGeometry3D.Positions.RemoveAt( j );
        meshGeometry3D.Positions.Insert( j, p );
      }
    }
    break;
  case "reset":
    perspectiveCamera.Position      = new Point3D ( 0,0, 3 );
    perspectiveCamera.LookDirection = new Vector3D( 0,0,-1 );
    emptySlices = 0;
    GenerateImageMaterials();
    GenerateAllCylinders();
    break;
  } //end of switch
} // end of on_button_clicked(...)
}
}

```