

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

Chapter C2: Face

Copyright © by V. Miszalok, last update: 2010-01-07



Preliminaries

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

- 1) Main Menu after start of Visual C# 2010 Express: Tools → Options → check lower left checkbox: Show all Settings → Projects and Solutions → Visual Studio projects location: → C:\temp
- 2) Main Menu after start of Visual C# 2010 Express: File → New Project... → WPF Application → Name: face1 → OK.
- 3) Main menu of Visual C# 2010 Express → Tools → Options... → An Options-window appears. **Double-click** the branch Text Editor. **Double-click** C#. **Double-click** Formatting. Click General. Uncheck all three check boxes. Click Indentation. Uncheck all four check boxes. Click New Lines. Uncheck all thirteen check boxes. Click Spacing. Uncheck all twenty three check boxes. Click IntelliSense. Uncheck all six check boxes. Quit the Options- window with the OK-button.

A picture on corrugated sheet iron

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

MainWindow.xaml:

```
<Window x:Class="face1.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Name="window"
  Title="face1" Width="400" Height="650">
  <Window.Resources><!--StaticResources save XAML-code.-->
    <!--Brush, Material, Camera can be defined here and referenced later in one line.-->
    <ImageBrush          x:Key="brush" ImageSource="http://www.miszalok.de/Images/Foto.jpg"/>
    <DiffuseMaterial     x:Key="material" Brush="{StaticResource brush }"/>
    <OrthographicCamera x:Key="camera" Position="0 0 10" LookDirection="0 0 -1" UpDirection="0 1 0"/>
  <!--Set two properties for all TextBlocks-->
  <Style TargetType="{x:Type TextBlock}">
    <Setter Property="FontSize" Value="10"/>
    <Setter Property="HorizontalAlignment" Value="Center"/>
  </Style>
  <!--Define one Slider-event-handler for all Sliders-->
  <Style TargetType="{x:Type Slider}">
    <EventSetter Event="ValueChanged" Handler="on_slider_value_changed"/>
  </Style>
</Window.Resources>
```

```

<StackPanel Orientation="Vertical" Margin="3">
  <TextBlock Text="Frequency"/>
  <Slider x:Name="frequency_slider"      Minimum="0"    Maximum="16"    Value="8"/>
  <TextBlock Text="Amplitude"/>
  <Slider x:Name="amplitude_slider"      Minimum="0"    Maximum="8"     Value="1"/>
  <TextBlock Text="Phase"/>
  <Slider x:Name="phase_slider"          Minimum="0"    Maximum="6.28" Value="0"/>
  <TextBlock Text="Rotation"/>
  <Slider x:Name="rot_slider"            Minimum="-90"  Maximum="90"   Value="0"/>
  <TextBlock Text="Swap from Ambient to Directional Light"/>
  <Slider x:Name="light_voltage_slider"  Minimum="0"    Maximum="255"  Value="0"/>
  <TextBlock Text="Directional Light Direction"/>
  <Slider x:Name="light_direction_slider" Minimum="-2"    Maximum="2"    Value="0"/>
  <!--Viewport3D is a drawing canvas which resizes its Content automatically-->
  <Viewport3D x:Name="viewport" Camera="{StaticResource camera}">
    <!--Any 3D-content must be packed in a ModelVisual3D-object-->
    <ModelVisual3D>
      <ModelVisual3D.Content>
        <!--Only one Content is allowed. Thus we have to bundle ours inside a group-object.-->
        <Model3DGroup>
          <!--GeometryModel3D obtains here nothing but a Name and its Material = ImageBrush.-->
          <!--Its Geometry will be defined later in facel.cs.-->
          <GeometryModel3D x:Name="geometryModel3D" Material="{StaticResource material}"/>
          <AmbientLight x:Name="ambientLight" Color="#ffffff"/>
          <DirectionalLight x:Name="directionalLight" Color="#000000" Direction="0 0 -1"/>
        </Model3DGroup>
      </ModelVisual3D.Content>
    </ModelVisual3D>
  </Viewport3D>
</StackPanel><!--end of <StackPanel Orientation="Vertical" Margin="3">-->
</Window>

```

MainWindow.xaml.cs:

```

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

namespace facel
{
  public partial class MainWindow : Window
  {
    const int nn = 50; //face width 1.0 will be divided into nn vertical stripes
    Point3D [] p = new Point3D [ 2*nn + 2 ]; //a stripe has 4 vertices
    Vector3D[] n = new Vector3D[ 2*nn + 2 ];
    Point    [] t = new Point    [ 2*nn + 2 ];
    int[] index = new int[ 6*nn ]; //2 triangles times 3 vertices per stripe
    double x, dx = 1.0 / nn;      //dx = stripe width
    int i, j;
    double frequency = 8, amplitude = 1, phase = 0;
    MeshGeometry3D mesh = new MeshGeometry3D();
    Matrix3D matrix = new Matrix3D();
    double old_rot_slider_value = 0;

    public MainWindow() //constructor
    {
      InitializeComponent();
      for ( i=0, x=0; i < p.Length; i+=2, x+=dx )
      {
        p[i].X =      p[i+1].X = x - 0.5; //face starts at x=-0.5 and ends at x = 0.5
        t[i].X =      t[i+1].X = x;      //texture starts at x= 0.0 and ends at x = 1.0
        p[i].Y = 0.5; p[i+1].Y = -0.5; //face top and bottom
        t[i].Y = 0;   t[i+1].Y = 1;    //texture top and bottom
        p[i].Z =      p[i+1].Z = amplitude*Math.Sin( phase + x*frequency*Math.PI );
      }
      for ( i=0, j=0; i < 6*nn; i+=6, j+=2 )
      {
        index[i ] = j; //1. vertex of 1. triangle
        index[i+1] = j+1; //2. vertex of 1. triangle
        index[i+2] = j+2; //3. vertex of 1. triangle
        index[i+3] = j+1; //1. vertex of 2. triangle
        index[i+4] = j+3; //2. vertex of 2. triangle
        index[i+5] = j+2; //3. vertex of 2. triangle
      }
    }
  }
}

```

```

    mesh.Positions          = new Point3DCollection ( p );
    mesh.TextureCoordinates = new PointCollection   ( t );
    mesh.TriangleIndices   = new Int32Collection   ( index );
    geometryModel3D.Geometry = mesh;
    matrix.Scale ( new Vector3D ( 1.6, 1.6, 1 ) );
    matrix.Rotate( new Quaternion( new Vector3D(1,0,0), 1 ) );
    geometryModel3D.Transform = new MatrixTransform3D( matrix );
}

private void on_slider_value_changed(object sender, EventArgs e)
{ switch ( ((Slider)sender).Name )
  { case "amplitude_slider":
    case "frequency_slider":
    case "phase_slider":
      amplitude = amplitude_slider.Value;
      frequency = frequency_slider.Value;
      phase     = phase_slider   .Value;
      for ( i=0, x=0; i < p.Length; i+=2, x+=dx )
        p[i].Z = p[i+1].Z = amplitude*Math.Sin( phase + x*frequency*Math.PI );
      mesh.Positions = new Point3DCollection( p );
      geometryModel3D.Geometry = mesh;
      break;
    case "rot_slider":
      double angle_increment = rot_slider.Value - old_rot_slider_value;
      matrix.Rotate( new Quaternion( new Vector3D(0,0,1), angle_increment ) );
      geometryModel3D.Transform = new MatrixTransform3D( matrix );
      old_rot_slider_value = rot_slider.Value;
      break;
    case "light_voltage_slider":
      Byte directionalVolt = (Byte)light_voltage_slider.Value;
      Byte ambientVolt     = (Byte)( 255 - directionalVolt );
      directionalLight.Color = Color.FromRgb( directionalVolt, directionalVolt,
                                              directionalVolt );
      ambientLight   .Color = Color.FromRgb( ambientVolt   , ambientVolt   ,
                                              ambientVolt   );
      break;
    case "light_direction_slider":
      directionalLight.Direction = new Vector3D( light_direction_slider.Value, 0, -2 );
      break;
  }
}
protected override void OnRenderSizeChanged(SizeChangedEventArgs sizeInfo)
{ viewport.Width = window.ActualWidth;
  viewport.Height = window.ActualHeight - 12*frequency_slider.ActualHeight;
}
}
}
}

```

geometryModel3D.Geometry:

The image is symmetrically centered around $(0,0)$.

It is divided into nn vertical stripes each containing 2 triangles.

$p[0] = (-0.5, +0.5)$ = upper left corner position of the mesh,

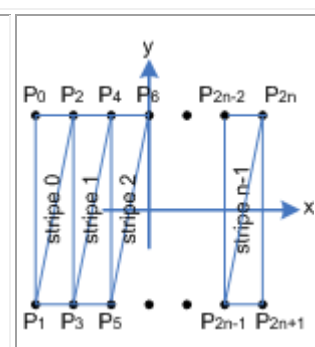
$p[2nn+1] = (+0.5, -0.5)$ = lower right corner position of the mesh.

The mesh undulates in depth with $p[i].Z = p[i+1].Z =$
 $amplitude * \text{Sinus}(phase + x * frequency * \Pi)$.

The corresponding flat texture coordinates $t[i].X$ and $t[i].Y$ have to be ≥ 0 :

$t[0] = (0.0, 0.0)$ = upper left corner of the image,

$t[2nn+1] = (1.0, 1.0)$ = lower right corner of the image.



Experiments:

1.	Shift <code>Frequency</code> to 10%, <code>Amplitude</code> to 100% and <code>Phase</code> from 0 to 100%	Morphing
2.	Try out all sliders.	
3.	Drag the lower border of the <code>face1</code> -window down.	Image size remains unchanged
4.	Drag the right border of the <code>face1</code> -window to the left.	Zoom down
5.	Drag the right border of the <code>face1</code> -window to the right.	Zoom up

Experiments in `face1.xaml`: (Restore the original values after any experiment.)

1.	Double the <code>Maximum</code> -values of all sliders.	More crazy effects
2.	Try out other images. For images from the hard disk use a syntax like this: "C:\\temp\\image.jpg".	
3.	Try out other zoom factors in <code><ScaleTransform3D ScaleX="1.6" ScaleY="1.6" /></code> .	Smaller, bigger
4.	Try out other <code>OrthographicCamera</code> Positions.	Clipped image
5.	Reverse the <code>OrthographicCamera UpDirection="0 1 0"</code> to <code>"0 -1 0"</code> .	Upside down

Experiments in `face1.cs`: (Restore the original values after any experiment.)

1.	Reduce <code>const int nn = 50; //no of rectangular stripes</code> to 5 and to 1.	Angular and flat profile
2.	Change the line <code>p[i].X = p[i+1].X = x - 0.5;</code> to <code>p[i].X = p[i+1].X = x;</code>	Right part lost
3.	Change the line <code>p[i].Y = 0.5; p[i+1].Y = -0.5;</code> to <code>p[i].Y = 0.5; p[i+1].Y = 0;</code>	Half height
4.	Change the line <code>t[i].Y = 0; t[i+1].Y = 1;</code> to <code>t[i].Y = 1; t[i+1].Y = 0;</code>	Upside down
5.	Remove the line <code>viewport.Width = window.ActualWidth;</code>	A viewport without width is invisible.