Course 3D_WPF: 3D-Computer Graphics with C# + WPF
Chapter C1: Moving Triangles

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: triangle1 → 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.

Version 01: A triangle

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

MainWindow.xaml

```
<Window x:Class="triangle1.MainWindow"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 Title="triangle1">
 <Window.Resources>
  <MeshGeometry3D x:Key="coordinates"
   Positions         ="-1 -1 0, 1 -1 0, 0   1 0"
   Normals           =" 0  0 1, 0  0 1, 0   0 1"
   TextureCoordinates=" 0  1  , 1  1  , 0.5 0  "/>
  <ImageBrush      x:Key="brush" ImageSource="http://www.miszalok.de/Images/Foto.jpg"/>
  <DiffuseMaterial x:Key="front" Brush="{StaticResource brush}"/>
  <DiffuseMaterial x:Key="back"  Brush="Gray"/>
 </Window.Resources>

 <Viewport3D>
  <ModelVisual3D>
   <ModelVisual3D.Content>
    <Model3DGroup x:Name="group">
     <GeometryModel3D x:Name="triangle_prototype"
      Geometry=    "{StaticResource coordinates}"
      Material=    "{StaticResource front      }"
      BackMaterial="{StaticResource back       }"/>
     <AmbientLight     Color="#404040"/>
     <DirectionalLight Color="#ffffff" Direction="0 0 -1"/>
    </Model3DGroup>
   </ModelVisual3D.Content>
  </ModelVisual3D>
```

```
  <Viewport3D.Camera>
   <PerspectiveCamera
    Position     ="0 0  5"
    LookDirection="0 0 -1"
    UpDirection  ="0 1  0"/>
  </Viewport3D.Camera>
 </Viewport3D>
</Window>
```

MainWindow.xaml.cs:

```
using System;
using System.Windows;
using System.Windows.Controls;
namespace triangle1
{ public partial class MainWindow: Window
  { public MainWindow()
    { InitializeComponent();
    }
  }
}
```

Experiments with MainWindow.xaml: (Restore the original values after any experiment.)

1.	Change the y-coordinate of the second vertex from `Positions="-1 -1 0, 1 -1 0, 0 1 0"` to `"-1 -1 0, 1 -1 0, 0 3 0"`.	acute triangle
2.	Shift away the `PerspectiveCamera Position="0 0 5"` to `"0 0 10"`.	half size
3.	Reverse the `PerspectiveCamera Position="0 0 5"` to `"0 0 -5"` and reverse the `LookDirection` from `"0 0 -1"` to `"0 0 1"`.	back side
4.	Change the y-axis `UpDirection="0 1 0"` to `"0 -1 0"`.	upside down
5.	Switch off the `DirectionalLight"` to `Color="#000000"`.	constant dim light
6.	Switch off the `AmbientLight"` to `Color="#000000"`.	Brightness depends merely on orientation; back side is black.

Version 02: A triangle rotates in 3D

Replace the complete code of **`MainWindow.xaml.cs`** by:

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Threading;
using System.Windows.Media.Media3D;
namespace triangle1
{ public partial class MainWindow: Window
  { DispatcherTimer timer = new DispatcherTimer();
    AxisAngleRotation3D axis_angle = new AxisAngleRotation3D();
    public MainWindow()
    { InitializeComponent();
      axis_angle.Axis = new Vector3D( 0,1,0 );
      triangle_prototype.Transform = new RotateTransform3D( axis_angle );
      timer.Interval = TimeSpan.FromMilliseconds( 1 );
      timer.Tick += TimerOnTick;
      timer.Start();
    }
    private void TimerOnTick( Object sender, EventArgs args )
    { axis_angle.Angle++;
    }
  }
}
```

Experiments with triangle1.xaml: (Restore the original values after any experiment.)

1.	Change the x-TextureCoordinate of the third vertex from `TextureCoordinates=" 0 1, 1 1, 0.5 0"` to `"0 1, 1 1, 0.1 0"`.	right eye lost
2.	Change the image `ImageSource="http://www.miszalok.de/Images/Foto.jpg"` to `"http://www.discoverhongkong.com/eng/showtime/lighting/images/st_mega_sol.jpg"`.	HongKong
3.	Change the line `BackMaterial="{StaticResource back }"/>` to `BackMaterial="{StaticResource front}"/>`.	Image on both sides

Experiments with MainWindow.xaml.cs: (Restore the original values after any experiment.)

1.	Change the rotation axis `axis_angle.Axis = new Vector3D( 0,1,0 );` to `"1 0 0"` and to `"0 0 1"`.	rot around the X- and Z-axis
2.	Slow down the animation from `TimeSpan.FromMilliseconds( 1 );` to `100` and to `400` milliseconds.	slower
3.	Speed up the animation from `axis_angle.Angle++;` to `axis_angle.Angle += 2;`.	double speed

Version 03: Three triangles in 3D

Replace the **`<Model3DGroup x:Name="group">`**-block of **`MainWindow.xaml`** by:

```
<Model3DGroup x:Name="group">
 <GeometryModel3D x:Name="t_mid"
  Geometry=    "{StaticResource coordinates}"
  Material=    "{StaticResource front      }"
  BackMaterial="{StaticResource back       }"/>
 <GeometryModel3D x:Name="t_left"
  Geometry=    "{StaticResource coordinates}"
  Material=    "{StaticResource front      }"
  BackMaterial="{StaticResource back       }"/>
 <GeometryModel3D x:Name="t_right"
  Geometry=    "{StaticResource coordinates}"
  Material=    "{StaticResource front      }"
  BackMaterial="{StaticResource back       }"/>
 <AmbientLight     Color="#404040"/>
 <DirectionalLight Color="#ffffff" Direction="0 0 -1"/>
</Model3DGroup>
```

Replace the complete **`MainWindow.xaml.cs`** by:

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Threading;
using System.Windows.Media.Media3D;
namespace triangle1
{ public partial class MainWindow: Window
  { DispatcherTimer timer = new DispatcherTimer();
    Matrix3D matrix_m = new Matrix3D(),
             matrix_l = new Matrix3D(),
             matrix_r = new Matrix3D();
    //This quaternion defines a rotation of 1 degree around the Y-axis (0,1,0)
    //It will be used in void TimerOnTick(...) to animate the triangles
    Quaternion q = new Quaternion( new Vector3D(0,1,0), 1 );
```

```
    public MainWindow()
    { InitializeComponent();
      matrix_l.Scale    ( new Vector3D( 0.5,0.5,0.5 ) ); //smaller
      matrix_l.Translate( new Vector3D(-1  ,0  ,0   ) ); //shift left
      matrix_r.Scale    ( new Vector3D( 0.5,0.5,0.5 ) ); //smaller
      matrix_r.Translate( new Vector3D( 1  ,0  ,0   ) ); //shift right
      timer.Interval = TimeSpan.FromMilliseconds( 1 );
      timer.Tick += TimerOnTick;
      timer.Start();
    }
    void TimerOnTick( Object sender, EventArgs args )
    { matrix_m.Rotate       ( q ); //1 degree rotation
      matrix_l.RotatePrepend( q ); //Insert this rotation in front of Scale + Translate
      matrix_r.RotatePrepend( q ); //Insert this rotation in front of Scale + Translate
      t_mid  .Transform = new MatrixTransform3D( matrix_m );
      t_left .Transform = new MatrixTransform3D( matrix_l );
      t_right.Transform = new MatrixTransform3D( matrix_r );
    }
  }
}
```

Experiments with MainWindow.xaml.cs: (Restore the original values after any experiment.)

1.	Shrink the left triangle from `matrix_l.Scale(new Vector3D(0.5,0.5,0.5));` to `(0.1,0.1,0.1)`.	asymmetry
2.	Zoom up the right triangle from `matrix_r.Scale(new Vector3D(0.5,0.5,0.5));` to `(2,2,2)`.	asymmetry
3.	Shift away the left triangle from `matrix_l.Translate(new Vector3D(-1,0,0));` to `"(-2,0,0)"`.	asymmetry
4.	Change the rotation axis of `Quaternion q=new Quaternion(new Vector3D(0,1,0),1);` to `(1,0,0)` and to `(0,0,1)`.	rot around the X- and Z-axis
5.	Speed up the animation in `Quaternion q=new Quaternion(new Vector3D(0,1,0),1);` and change the last parameter=rotation angle from `1` to `3` degrees.	triple speed

Version 04: Many triangles in 3D

Replace the big `<Model3DGroup x:Name="group">`-block in **`MainWindow.xaml`** by the much simpler `group`:

```
    <Model3DGroup x:Name="group">
     <GeometryModel3D x:Name="triangle_prototype"
      Geometry=    "{StaticResource coordinates}"
      Material=    "{StaticResource front      }"
      BackMaterial="{StaticResource back       }"/>
     <AmbientLight     Color="#404040"/>
     <DirectionalLight Color="#ffffff" Direction="0 0 -1"/>
    </Model3DGroup>
```

Replace the complete code in **`MainWindow.xaml.cs`** by:

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Threading;
using System.Windows.Media.Media3D;
using System.Windows.Markup;
using System.IO;
using System.Xml;
```

```
namespace triangle1
{ public partial class MainWindow: Window
  { const Int32 nTriangles = 50;
    Matrix3D[] matrix          = new Matrix3D       [nTriangles];
    GeometryModel3D[] triangle = new GeometryModel3D[nTriangles];
    float[] dx                 = new float          [nTriangles];
    float[] dy                 = new float          [nTriangles];
    float[] dz                 = new float          [nTriangles];
    //This quaternion defines a rotation of 1 degree around axis (1,1,1)
    //It will be used in void TimerOnTick(...) to animate all triangles
    Quaternion quaternion  = new Quaternion( new Vector3D(1,1,1), 1 );
    DispatcherTimer timer  = new DispatcherTimer();
    Random r = new Random();
    public MainWindow()
    { InitializeComponent();
      //Remember triangle_prototype as XAML-string
      string triangle_prototype_string = XamlWriter.Save( triangle_prototype );
      group.Children.Remove( triangle_prototype ); //Kick the prototype out
      for ( int i = 0; i < nTriangles; i++ )
      { dx[i]  = 2*(float)r.NextDouble() - 1;  //random permanent translation dx
        dy[i]  = 2*(float)r.NextDouble() - 1;  //random permanent translation dy
        dz[i]  = 2*(float)r.NextDouble() - 1;  //random permanent translation dz
        matrix[i].Rotate   ( new Quaternion( new Vector3D(1,1,1),
                                             (float)r.NextDouble() * 45 ) );
        matrix[i].Scale    ( new Vector3D( 0.5  ,0.5  ,0.5   ) ); //smaller
        matrix[i].Translate( new Vector3D( dx[i],dy[i],dz[i] ) ); //random shifts
        //Clone the triangle_prototype from its XAML-string
        StringReader  sr = new StringReader( triangle_prototype_string );
        XmlTextReader xr = new XmlTextReader( sr );
        triangle[i] = (GeometryModel3D)XamlReader.Load( xr );
        group.Children.Add( triangle[i] );
      }
      timer.Interval = TimeSpan.FromMilliseconds( 1 );
      timer.Tick += TimerOnTick;
      timer.Start();
    }

    void TimerOnTick( Object sender, EventArgs args )
    { for ( int i = 0; i < nTriangles; i++ )
      { //Insert this rotation in front of previous transforms
        matrix[i].RotatePrepend( quaternion );
        triangle[i].Transform = new MatrixTransform3D( matrix[i] );
      }
    }
  }
}
```

Experiments: (Restore the original values after any experiment.)

1.	Change the no of triangles from `const Int32 nTriangles = 50;` to `5` and to `500`.	less or more triangles
2.	Expand the triangle cloud by changing `dx[i]`, `dy[i]` and `dz[i]` from `2*(float)r.NextDouble()-1;` to `4*(float)r.NextDouble()-2;`.	more space between the triangles
3.	Change the initial rotation angle from `matrix[i].Rotate(new Quaternion(new Vector3D(1,1,1),(float)r.NextDouble()*45));` to `(float)r.NextDouble()*180` and to `360`.	more chaos
4.	Change the rotation axis from `Quaternion q=new Quaternion(new Vector3D(1,1,1),1);` to `(1,0,0)`, to `(0,1,0)` and to `(0,0,1)`.	rot just around the X-, Y- and Z-axis
5.	Zoom down the triangles from `matrix[i].Scale(new Vector3D(0.5,0.5,0.5));` to `(0.1,0.1,0.1)`.	small triangles