Course 2DCis: 3D-Computer Graphics with C#
Chapter C1: The Complete Code of the Moving Triangles Project

Copy all this code into an empty `Form1.cs` of a new `Windows Application C#`-project `triangle1` and (in case of VS 2005) clear `Form1.Designer.cs` and `Program.cs`.

If the `Solution Explorer -` window is invisible, open it via the main menu: `View -> Solution Explorer`.
In the `Solution Explorer -` window click the + sign in front of `triangle1`. A tree appears with a branch: "`References`".
Right-click on `References` and **left**-click on `Add Reference...`.
An `Add Reference` Dialog Box opens. Scroll down until you see the component name: `Microsoft.DirectX Version 1.0.2902.0`.
Mark this reference by a left-click and (keeping the Strg-key pressed) mark the reference which follows somewhere below `Microsoft.DirectX.Direct3D Version 1.0.2902.0`.
Quit the `Add Reference` Dialog Box with `OK`.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  static Device device = null;
  static float fAngle;
  VertexBuffer vertexBuffer;
  CustomVertex.PositionColored[] v = new CustomVertex.PositionColored[3];
  Timer myTimer = new Timer();
  const Int32 nTriangles = 100;
  static float[] dx = new float[nTriangles];
  static float[] dy = new float[nTriangles];
  static float[] dz = new float[nTriangles];
  static float[] ax = new float[nTriangles];
  static float[] ay = new float[nTriangles];
  static float[] az = new float[nTriangles];
  static  Random r = new Random();
  public Form1()
  { Text = "D3DTriangleAnimation";
    //fill coordinates and colors into an array "v"
    v[0].X=-1f; v[0].Y=-1f; v[0].Z=0f;
    v[1].X= 1f; v[1].Y=-1f; v[1].Z=0f;
    v[2].X= 0f; v[2].Y= 1f; v[2].Z=0f;
    v[0].Color = System.Drawing.Color.DarkGoldenrod.ToArgb();
    v[1].Color = System.Drawing.Color.MediumOrchid.ToArgb();
    v[2].Color = System.Drawing.Color.Cornsilk.ToArgb();
    myTimer.Tick += new EventHandler( OnTimer );
    myTimer.Interval = 1;
    for ( int i = 0; i < nTriangles; i++ )
    { dx[i] = (float)r.NextDouble(); //random permanent translation dx
      dy[i] = (float)r.NextDouble(); //random permanent translation dy
      dz[i] = (float)r.NextDouble(); //random permanent translation dz
      ax[i] = (float)r.NextDouble(); //random initial pitch rotation angle
      ay[i] = (float)r.NextDouble(); //random initial yaw   rotation angle
      az[i] = (float)r.NextDouble(); //random initial roll  rotation angle
    }
    ClientSize = new Size( 400, 300 ); //Calls OnResize( ... )
  }
```

```
  protected override void OnResize( System.EventArgs e )
  //Whenever the window changes we have to initialize Direct3D from scratch
  { myTimer.Stop();// stop the timer during initialization
    try
    { //get information from the operating system about its current graphics properties
      PresentParameters presentParams = new PresentParameters();
      //we have to set two extra flags
      presentParams.Windowed = true;                //no full screen display
      presentParams.SwapEffect = SwapEffect.Discard; //no swap buffer
      //create a new D3D-device that serves as canvas
      if ( device != null ) device.Dispose(); //free the old canvas if any
      device = new Device( 0, DeviceType.Hardware, this,
                           CreateFlags.SoftwareVertexProcessing, presentParams );
      //set up the transformation of world coordinates into camera or view space
      device.Transform.View = Matrix.LookAtLH(
        new Vector3( 0f, 0f,-4f ),   //eye point 4.0 in front of the canvas
        new Vector3( 0f, 0f, 0f ),   //camera looks at point 0,0,0
        new Vector3( 0f, 1f, 0f ) ); //worlds up direction is the y-axis
      //set up the projection transformation using 4 parameters:
      //1.: field of view = 45 degrees; 2.: aspect ratio = heigth / width = 1 = square window;
      //3.: near clipping distance = 0; 4.: far clipping distance = 10;
      device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI/4, 1f, 0f, 10f );
      //Turn off culling, so the user sees the front and back of the triangle
      device.RenderState.CullMode = Cull.None;
      //Turn off lighting, since the triangle provides its own colors
      device.RenderState.Lighting = false;
      //set up the property that fills the triangle with colors
      device.VertexFormat = CustomVertex.PositionColored.Format;
      if ( vertexBuffer != null ) vertexBuffer.Dispose();//Free the old vertexBuffer if any.
      //Create a new vertex buffer on the graphic card and connect it to the device.
      vertexBuffer = new VertexBuffer( typeof(CustomVertex.PositionColored), 3,
                                       device, Usage.WriteOnly,
                                       CustomVertex.PositionColored.Format,
                                       Pool.Default );
      vertexBuffer.SetData( v, 0, LockFlags.None );//Copy vertices from main to graphic memory.
      device.SetStreamSource( 0, vertexBuffer, 0 );//Use the vertexBuffer on the graphic card.
      myTimer.Start();//start the timer again
    }
    catch (DirectXException) { MessageBox.Show( "Could not initialize Direct3D." ); return; }
  }

  protected static void OnTimer( Object myObject, EventArgs myEventArgs )
  { if (device == null) return;
    //throw the old image away
    device.Clear( ClearFlags.Target, Color.Blue, 1f, 0 );
    //rotate with an angular velocity = 0.1
    fAngle += 0.1f;
    device.Transform.World = Matrix.RotationY( fAngle );
    //draw on the canvas
    device.BeginScene();
      for ( int i = 0; i < nTriangles; i++ )
     { device.Transform.World = Matrix.Scaling( 0.5f, 0.5f, 0.5f )
                              * Matrix.Translation( dx[i], dy[i], dz[i] )
                              * Matrix.RotationYawPitchRoll( ay[i] += 0.01f * (float)r.NextDouble(),
                                                             ax[i] += 0.01f * (float)r.NextDouble(),
                                                             az[i] += 0.01f * (float)r.NextDouble() );
       device.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
      }
    device.EndScene();
    device.Present(); //show the canvas
  }
}
```