# Course 3D_XNA: 3D-Computer Graphics with XNA
# Chapter C1: Moving Triangles





This project is the XNA-clone of
3D-Computer Graphics with Managed DirectX, Chapter C1: Moving Triangles and of
3D-Computer Graphics with C++ and OpenGL, Chapter C1: Moving Triangles.

## Project triangle1

This chapter requires no DirectX SDK but:
1. Visual C# 2008 Express or Visual Studio 2008 Professional
2. XNA Game Studio 3.0
3. DirectX End-User Runtime November 2008

**1.** Main Menu after starting VS 2008: `File` → `New Project...` →
`Project types: XNA GameStudio 3.0` → `Templates: Windows Game (3.0)` → `Name: triangle1` → `Location: C:\temp` → `Create directory for solution:` switch it off → `OK`.
`Solution Explorer - triangle1`: Delete the file `Program.cs` and the default code of `Game1.cs`.

**2.** If You find no `Solution Explorer`-window, open it via the main menu: `View → Solution Explorer`. Inside the `Solution Explorer`-window click the plus-sign in front of `triangle1`. A tree opens. Look for the branch "`References`". Check if there are four references `Microsoft.XNA.Framework`, `Microsoft.XNA.Framework.Game`, `mscorlib` and `System`.

**3.** You should switch off the vexatious automatic format- and indent- mechanism of the code editor before you copy the following code to `triangle1.cs` (otherwise all the code will be reformatted into chaos):
3.1. Main menu of Visual Studio 2008 Professional: click menu "`Tools`".
3.2. A Drop-Down-menu appears. Click "`Options...`".
3.3. An `Options` dialog box appears.
3.4. Double click the branch "`Text Editor`" → double click "`C#`" → double click "`Formatting`".
3.5. A sub-tree appears with the 5 branches "`General, Indentation, New Lines, Spacing, Wrapping`".
3.6. Uncheck any of the 45 check boxes that these 5 `Formatting`-branches may offer.
3.7. Leave the dialog box with button "`OK`".

## Game1, Initialize, Update, Draw

Write the following code into the empty code window of `Game1.cs`:

```
using System;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;

static class Program
{ static void Main() { Game1 game = new Game1(); game.Run(); }
public class Game1 : Microsoft.Xna.Framework.Game
{ private GraphicsDeviceManager g;
  private VertexBuffer vbuffer;
  private BasicEffect  effect;
  private EffectPass   pass;
  private float rotAngle = 0f;
```

```
  public Game1()
  { Window.Title = "3DTriangleAnimation";
    Window.AllowUserResizing = true;
    g = new GraphicsDeviceManager( this );
    g.IsFullScreen = false;
    g.PreferredBackBufferWidth  = 300;
    g.PreferredBackBufferHeight = 300;
  }
  protected override void Initialize()
  { VertexPositionColor[] v = new VertexPositionColor[3];
    v[0].Position.X=-1f; v[0].Position.Y=-1f; v[0].Position.Z=0f;
    v[1].Position.X= 0f; v[1].Position.Y= 0f; v[1].Position.Z=0f;
    v[2].Position.X= 1f; v[2].Position.Y=-1f; v[2].Position.Z=0f;
    v[0].Color = Color.Blue;
    v[1].Color = Color.Cornsilk;
    v[2].Color = Color.MediumOrchid;
    vbuffer = new VertexBuffer( g.GraphicsDevice, typeof(VertexPositionColor),
                                3, BufferUsage.WriteOnly );
    vbuffer.SetData< VertexPositionColor >( v );
    effect = new BasicEffect( g.GraphicsDevice, null );
    //effect.View = Matrix.CreateLookAt( new Vector3(0, 0, 3),
                                         Vector3.Zero, Vector3.Up );
    //effect.Projection = Matrix.CreatePerspectiveFieldOfView( MathHelper.Pi/4,
                                                               1f, 1f, 10f );
    effect.VertexColorEnabled = true;
    pass = effect.CurrentTechnique.Passes[0];
  }
  protected override void Update( GameTime gameTime )
  { g.GraphicsDevice.Vertices[0].SetSource(vbuffer,0, VertexPositionColor.SizeInBytes);
    g.GraphicsDevice.VertexDeclaration =
     new VertexDeclaration( g.GraphicsDevice, VertexPositionColor.VertexElements );
    //g.GraphicsDevice.RenderState.CullMode = CullMode.None;
  }
  protected override void Draw( GameTime gameTime )
  { //g.GraphicsDevice.Clear( Color.DarkBlue );
    //effect.World = Matrix.CreateRotationY( rotAngle += 0.01f );
    effect.Begin();
      pass.Begin();
        try { g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
            } catch {}
      pass.End();
    effect.End();
  }
} // end of class Game1
} // end of class Program
```

Click Debug → `Start Without Debugging Ctrl F5`. Drag the window borders.

**Enhancements**: (`Start Without Debugging Ctrl F5` after any change.)

| | | |
|---|---|---|
| 1. | Change `g.PreferredBackBufferWidth` and `g.PreferredBackBufferHeight` from `300` to `600`. | The window is larger at start. |
| 2. | Change `v[1].Position.Y= 0f;` to `v[1].Position.Y= 1f;`. | The triangle is equilateral. |
| 3. | Change `v[0].Color = Color.Blue;` to `v[0].Color = Color.DarkGoldenrod;`. | The lower left color changes. |
| 4. | Remove the comment slashes `//` in front of `//g.GraphicsDevice.Clear( Color.DarkBlue );` | blue background |
| 5. | Remove the comment slashes `//` in front of `//g.GraphicsDevice.RenderState.CullMode = CullMode.None;`<br>Remove the comment slashes `//` in front of `//effect.View = ...`<br>Remove the comment slashes `//` in front of `//effect.Projection = ...` | Triangle seems smaller. |
| 6. | Remove the comment slashes `//` in front of `//effect.World *= ...` | Rotation around the Y-axis |

Keep all enhancements.
**Experiments**: (Recover the change after any experiment.)

| | | |
|---|---|---|
| 1. | Set comment slashes `//` in front of `g.GraphicsDevice.RenderState.CullMode = CullMode.None;`. | The backside of the triangle disappears. |
| 2. | Change the Z-value of the camera position in `new Vector3(0,0,3)` in `effect.View=Matrix.CreateLookAt(new Vector3(0, 0,3),Vector3.Zero,Vector3.Up)` from 3 to `4, 2, -2, -3`. | The camera moves on the Z-axis. |
| 3. | Change the direction of the Y-axis in `effect.View=Matrix.CreateLookAt(new Vector3(0,0,3),Vector3.Zero,Vector3.Up )` from `Vector3.Up` to `Vector3.Down`. | The triangle appears upside down. |
| 4. | Open the viewing angle in `effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/4,1f,1f,10f);` from `MathHelper.Pi/4` = 45° to `MathHelper.Pi/2` = 90°. | The triangle appears smaller. |
| 5. | Change the angle increment in `effect.World=Matrix.CreateRotationY(rotAngle+=0.01f);` from `0.01f` to `0.1f` and to `0.003f`. | Faster and slower rotation |
| 6. | Change the rotation axis in `effect.World=Matrix.CreateRotationY(rotAngle+=0.01f);` from `CreateRotationY` to `CreateRotationX`. | Rotation around the X-axis |
| 7. | Change the rotation axis in `effect.World=Matrix.CreateRotationY(rotAngle+=0.01f);` from `CreateRotationY` to `CreateRotationZ`. | Rotation around the Z-axis |

## Three triangles

Declare 4 new global variables in the head of `public class Game1 : Microsoft.Xna.Framework.Game` just below the line `float rotationAngle = 0f;`:

```
  private Matrix rot        = Matrix.CreateRotationY( 0f );
  private Matrix zoom       = Matrix.CreateScale( 0.5f );
  private Matrix leftshift  = Matrix.CreateTranslation( new Vector3( -1f, 0f, 0f ) );
  private Matrix rightshift = Matrix.CreateTranslation( new Vector3(  1f, 0f, 0f ) );
```

Change the `protected override void Draw( GameTime gameTime )`-function as follows:

```
  protected override void Draw( GameTime gameTime )
  { g.GraphicsDevice.Clear( Color.DarkBlue );
    rot = Matrix.CreateRotationY( rotAngle += 0.01f );
    //rot = Matrix.CreateFromYawPitchRoll( rotAngle += 0.01f, rotAngle, rotAngle );
    effect.Begin();
      pass.Begin();
        effect.World = rot;
        try { g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
            } catch {}
      pass.End();
      pass.Begin();
        effect.World = zoom * rot * leftshift;
        try { g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
            } catch {}
      pass.End();
      pass.Begin();
        effect.World = zoom * rot * rightshift;
        try { g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
            } catch {}
      pass.End();
    effect.End();
  }
```

Click `Debug` → `Start Without Debugging Ctrl F5`. Drag the window borders.

**Experiments**: (Recover the change after any experiment.)

| | | |
|---|---|---|
| 1. | Change the `zoom`- factor of the small triangles in `Matrix zoom = Matrix.CreateScale(0.5f);` from `0.5f` to `0.2f` and to `1.2f`. | Smaller / larger triangles. |
| 2. | Change the `leftshift` of the left small triangle in `Matrix leftshift = Matrix.CreateTranslation(new Vector3(-1f,0f,0f));` from `-1f` to `-1.5f` and to `-0.5f`. | The left triangle is horizontally shifted along the X-axis. |
| 3. | Change the `leftshift` of the left small triangle in `Matrix leftshift = Matrix.CreateTranslation(new Vector3(-1f,0f,0f));` from `new Vector3(-1f,0f,0f)` to `new Vector3(-1f,1f,0f)` and to `new Vector3(-1f,-1f,0f)`. | The left triangle is vertically shifted. |
| 4. | Change the `leftshift` of the left small triangle in `Matrix leftshift = Matrix.CreateTranslation(new Vector3(-1f,0f,0f));` from `new Vector3(-1f,0f,0f )` to `new Vector3(-1f,0f,-2f)`. | The left triangle is shifted back and seems smaller. |
| 5. | Set comment slashes `//` in front of `//Matrix.CreateRotationY( rotationAngle );`. Remove the comment slashes `//` in front of `//rot=Matrix.CreateFromYawPitchRoll(rotAngle+=0.01f,rotAngle,rotAngle);`. Keep this change. | The triangles rotate simultaneously around their X-, Y- and Z-axes. |

## Hundred triangles

Add the following declarations to the head of `public class Game1 : Microsoft.Xna.Framework.Game` below the line: `Matrix rightshift=Matrix.CreateTranslation(new Vector3(1f,0f,0f));`:

```
  private const Int32 nTriangles = 100;
  private float[] dx = new float[nTriangles];
  private float[] dy = new float[nTriangles];
  private float[] dz = new float[nTriangles];
  private float[] ax = new float[nTriangles];
  private float[] ay = new float[nTriangles];
  private float[] az = new float[nTriangles];
  private Random r = new Random();
```

Add the following intialisations to `protected override void Initialize()` below the line: `pass = effect.CurrentTechnique.Passes[0];`.

```
    for ( int i = 0; i < nTriangles; i++ )
    { dx[i] = (float)r.NextDouble() - 0.5f;  //random permanent translation dx
      dy[i] = (float)r.NextDouble() - 0.5f;  //random permanent translation dy
      dz[i] = (float)r.NextDouble() - 0.5f;  //random permanent translation dz
    }

    g.GraphicsDevice.RenderState.DepthBufferEnable = false;
    g.GraphicsDevice.RenderState.DepthBufferWriteEnable = false;
```

Change the `protected override void Draw( GameTime gameTime )` - function as follows:

```
  protected override void Draw( GameTime gameTime )
  { g.GraphicsDevice.Clear( Color.DarkBlue );
    rot = Matrix.CreateFromYawPitchRoll( rotAngle += 0.01f, rotAngle, rotAngle );
    effect.Begin();
      for ( int i = 0; i < nTriangles; i++ )
      { pass.Begin();
          leftshift = Matrix.CreateTranslation( dx[i], dy[i], dz[i] );
          effect.World = zoom * rot * leftshift;
          try { g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
              } catch {}
        pass.End();
      }
    effect.End();
  }
```

Click `Debug` → `Start Without Debugging Ctrl F5`.
Hundred small triangles will dance synchronously as fishes do.

# Chaos

Change `protected override void Initialize()` below the line:
`pass = effect.CurrentTechnique.Passes[0];` as follows:

```
    for ( int i = 0; i < nTriangles; i++ )
    { dx[i] = (float)r.NextDouble() - 0.5f;  //random permanent translation dx
      dy[i] = (float)r.NextDouble() - 0.5f;  //random permanent translation dy
      dz[i] = (float)r.NextDouble() - 0.5f;  //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
    }
    g.GraphicsDevice.RenderState.DepthBufferEnable      = false;
    g.GraphicsDevice.RenderState.DepthBufferWriteEnable = false;
```

Change the `protected override void Draw( GameTime gameTime )` - function as follows:

```
  protected override void Draw( GameTime gameTime )
  { g.GraphicsDevice.Clear( Color.DarkBlue );
    effect.Begin();
      for ( int i = 0; i < nTriangles; i++ )
      { pass.Begin();
          rot = Matrix.CreateFromYawPitchRoll( ay[i] += 0.01f,
                                               ax[i] += 0.01f,
                                               az[i] += 0.01f );
          leftshift = Matrix.CreateTranslation( dx[i], dy[i], dz[i] );
          effect.World = zoom * rot * leftshift;
          g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleList, 0, 1 );
        pass.End();
      }
    effect.End();
  }
```

Click `Debug` → `Start Without Debugging Ctrl F5`.
Hundred small triangles will dance as flying paper sheets do.

**Experiments**: (Recover the change after any experiment.)

| | | |
|---|---|---|
| 1. | Allow the default Z-test and put comment slashes `//` in front of both `g.GraphicsDevice.RenderState.DepthBufferEnable = false;` and `g.GraphicsDevice.RenderState.DepthBufferWriteEnable = false;`. Keep this change. | New triangles don't cover the old ones. |
| 2. | Vary the constant `const Int32 nTriangles = 100;` between `10` and `1000`. | Less or more triangles. |
| 3. | Vary the random initial pitch+yaw+roll rotation angles `ax[i], ay[i], az[i]` inside the constructor between `0f` and `(float)(2 * Math.PI * r.NextDouble())`. | The initial disorder changes. |
| 4. | Vary the retarding factors `0.01f` inside `Matrix.CreateFromYawPitchRoll` of the angular increments `ay[i], ax[i]` and `az[i]` between `0.001f` and `0.1f`. | Less or more chaos. |
| 5. | Insert a new line just below `pass.Begin();`: `zoom = Matrix.CreateScale( new Vector3( (float)r.NextDouble(), (float)r.NextDouble(), (float)r.NextDouble() ) );`. | Bustling excitement. |

# Help

**1**. I recommend the following german booklet: KONEROW Jens: XNA Framework. schnell + kompakt, **entwickler.press**, ISBN: 978-3-939084-53-2, € 7,90 (130 pages) and the code of Konerow's samples from **XNA Framework - Schnell & Kompakt Beispiele**.

**2**. You find extensive information about XNA here: **http://msdn2.microsoft.com/en-us/library/bb200104.aspx**.
Click trough the tree on the left side:
1: XNA Game Studio x.0 → Getting Started with XNA Game Studio → Your First Game: Microsoft XNA Game Studio in 2D
2: XNA Game Studio x.0 → Programming Guide
3: XNA Game Studio x.0 → XNA Framework Class Library Reference