Course 3D_XNA
Chapter C3: Comments to the Drunken Tiger Project

Recommended for early beginners:
XNA Overview: **http://msdn2.microsoft.com/en-us/library/bb200104.aspx**
Video Training: **http://msdn2.microsoft.com/en-us/xna/bb245766.aspx**

namespaces

```
using System; //Home of the base class of all classes "System.Object.
```

```
using Microsoft.Xna.Framework; //Provides commonly needed game classes such as timers and game loops.
```
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.aspx**

```
using Microsoft.Xna.Framework.Content;
```
//Contains functions to load a mesh or model from the Content Pipeline and to manage the lifespan of the loaded objects.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.content.aspx**

```
using Microsoft.Xna.Framework.Graphics; //Contains methods to access the graphics board directly.
```
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphics.aspx**

Entry to start `public class Game1 : Microsoft.Xna.Framework.Game`

//Our `Game1` is derived from `Microsoft.Xna.Framework.Game` which is the base class of XNA.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.game_members.aspx**

```
static class Program { [STAThread] static void Main()
                       { Game1 game = new Game1(); game.Run(); }
```
//Create a single thread instance of `Game1` and ask the operating system to start it.

```
private GraphicsDeviceManager g; //A GraphicsDevice performs rendering, creates resources and creates
```
shaders. The `GraphicsDeviceManager` handles the configuration and management of the `GrahipcsDevice`.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphicsdevicemanager_members.aspx**

```
private Model model; //A Model contains all parts of a 3D drawing, at least one ModelMesh=set of vertices,
```
one `BasicEffect`=graphics bord settings and one `Bone`=dependence level.
`Models` are stored in the Content Pipeline of the game.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphics.modelmesh_members.aspx**

```
private BasicEffect effect; //A BasicEffect is a part of Model and is loaded with Model from the content
```
pipeline of the game (file name extension: `.fx`).
It describes the current settings of the graphics board such as Vertex Shader, Pixel Shader and Technique and
it requires a set of world, view, and projection matrices, a vertex buffer and a vertex declaration.

```
private float trans             = 0,
              transIncrement    = 0.02f,
              zoom              = 1,
              zoomIncrement     = 0.005f,
              rotation          = 0,
              rotationIncrement = 0.01f; //no shift, no zoom and no rotation at start.
```

Constructor `public Game1()` inside `public class Game1`

```
g = new GraphicsDeviceManager( this ); //Create an instance of GraphicsDeviceManager. See above.
```

Event handler `protected override void Initialize()` inside `public class Game1`

//Obligatory event handler of any XNA Game

```
g.PreferredBackBufferWidth = 600; //Allocate output buffer space on the graphics board
g.PreferredBackBufferHeight = 600; //Allocate output buffer space on the graphics board
g.ApplyChanges(); //Force the graphics board to rearrange its memory structure
```

```
Window.Title = "Tiger Rotation"; //The title of the window.
Window.AllowUserResizing = true; //The user is allowed to change the window size at run time.
```

```
base.Initialize();
```

//Complete initialization of DirectX. Enumerate through any graphics components that have been added to Components and call their Initialize methods. This call is obligatory after the `GraphicsDevice` is created, but before `LoadContent`.

Event handler `protected override void LoadContent()` inside `public class Game1`

```
model = content.Load< Model >("Content\\tiger"); //The ContentManager loads both
```

the `tiger.x` mesh file and `tiger_texture.bmp` image file from directory `C:\temp\TigerRot1\Content`.
See: **Content Pipeline**

```
effect = (BasicEffect)model.Meshes[0].Effects[0];
```

//Extract from model "tiger" the first part of the first mesh and write it into the vertex buffer of the graphics board.
Since model "tiger" just has one single part in one single mesh this line loads the complete tiger.

```
effect.View = Matrix.CreateLookAt( new Vector3( 0f, 0f, 4f ), Vector3.Zero, Vector3.Up );
```

//Overrides the empty values of `effect` by the following settings: camera position 4.0 in front of the display.
The camera axis points from `(0,0,4)` to the center of the window `(0,0,0)`. The y-axis goes up.

```
effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/4, 1f, 1f, 1000000f);
```

//Overrides the empty values of `effect` by the following settings: viewing angle = 45 degrees; aspect ratio 1:1;
front clipping plane very close at z=1; back clipping plane very far away at z=1000000f.

Render method `protected override void Draw( GameTime gameTime )`.

// This function is derived from a virtual function `protected virtual void Draw( GameTime gameTime )`
of the parent class `Microsoft.Xna.Framework.Game` which calls this function automatically as often as possible.
Here is the right place to set the tiger's World-matrix.

```
//g.GraphicsDevice.RenderState.FillMode = FillMode.WireFrame;
```

//Experimental code which leaves the triangles empty and shows the tiger's naked wire frame.

```
g.GraphicsDevice.Clear( Color.DarkBlue ); //Erase any former content.
```

```
if ( trans < -0.8f | trans > 0.8f ) transIncrement *= -1f;
if ( zoom  <  0.1f | zoom  > 1.2f ) zoomIncrement  *= -1f;
```

// Reverse the shift when the tiger hits the left or the right border.
// Reverse the zoom when the tiger too small or too big.

```
trans    += transIncrement;
zoom     += zoomIncrement;
rotation += rotationIncrement; //These increments keep the tiger moving.
```

```
effect.World  = Matrix.CreateScale      ( zoom        );
effect.World *= Matrix.CreateRotationX  ( rotation    );
effect.World *= Matrix.CreateRotationY  ( rotation    );
effect.World *= Matrix.CreateRotationZ  ( rotation    );
effect.World *= Matrix.CreateTranslation( trans, 0, 0 );
```

//This statements drive the tiger mad. They replace the former world transform matrix of `effect` by a combined one
which is the result of a matrix multiplication of 5 single matrices. Each of them defines a specific movement.
The 3 rotation angles are identical. The last matrix shifts the tiger along the x-axis.

```
model.Meshes[0].Draw(); //Render the tiger and then flip back-buffer to front-buffer to show the scene.
```
