# Course 3D_XNA: 3D-Computer Graphics with XNA
# Chapter C6: Comments to the SkyBox Project

## 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 classes 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 classes, structures and enumerations to access the graphics board directly.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphics.aspx**

---

`using Microsoft.Xna.Framework.Input;` //Contains classes, structures and enumerations to receive input from keyboard, mouse, and Xbox 360 Controller.
See: **http://msdn2.microsoft.com/en-us/library/microsoft.xna.framework.input.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.

---

`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**

---

`ContentManager content;`
//`ContentManager` loads binary files from the content pipeline of the game.
It loads and prepares your graphic to be drawn, and will reload your graphic if the graphics device is reset (such as in the case of the game window being resized).

---

`Model skyModel, shipModel;`
//A `Model` contains all parts of a 3D drawing, at least one `ModelMesh`=set of vertices, one `BasicEffect`=graphics card 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**
Our current Content Pipeline consists of three resource-files: `ship.fbx, ship_texture.tga` and `skybox.x`. The term "Pipeline" comes from the fact, that such files normally form a chain where one file references and calls the next one.
Sample: `ship.fbx` and `ship_texture.tga` form such a chain,
because `ship_texture.tga` is not loaded by our code but by a reference inside `ship.fbx`.

---

`BasicEffect skyEffect;` //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.

---

`Matrix viewM, projM, skyworldM, shipworldM;` //Four matrices for transformations of vertices.
The `Matrix`-class provides methods for performing standard matrix operations such as addition, subtraction, and multiplication, in addition to helper methods for creating scale, translation, and rotation matrices.

---

`static float skyscale = 10000f;` //Zoom factor of the cubic sky hull.
Enlarging this factor enlarges the space of the sky. Narrowing this factor will shrink the sky. We must declare it `static` when we want it to initialize here. Otherwise we must initialize it inside the constructor `Game1()`.

---

`float slow = skyscale / 200f; // step width of movements`
//It's reasonable that the velocity of the ship depends on the size of the urrounding space.
When you lower the divisor to 20f, the ship will move 10 times faster.

---

`float rotationX, rotationY, rotationZ;` //These three rotation angles are set by the controller's left and right ThumbSticks. See below inside `Update(GameTime gameTime)`.

---

```
Vector3 shipPos = Vector3.Zero,
```
//The ship starts in the middle of the skybox
```
        cameraPos = new Vector3( 0f, 500f, 3000f ),
```
//The camera starts 500 above and 3000 behind the ship.
```
        camLookAt = new Vector3( 0f, 500f, -3000f );
```
//The camera looks at first horizontally parallel to the ship and in direction of the ship.

---

`public Game1()` //Constructor of `Game1`
`g = new GraphicsDeviceManager( this ); }` // Create an instance of `GraphicsDeviceManager` which we need below inside `Initialize()` to set some properties of our `GraphicsDevice`.
`content = new ContentManager( Services );` //Constructor of `ContentManager`. See above.

---

Event handler `protected override void Initialize()` inside `public class Game1`
//Obligatory event handler of any XNA Game

---

`g.IsFullScreen = false;` //`Game1` appears in a resizable window.
`Window.Title = "Dice";` //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 `LoadGraphicsContent`.

---

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

---

```
skyModel  = content.Load< Model >( "skybox" );
```
//Load the model "skybox" from the content pipeline of the game.
```
shipModel = content.Load< Model >( "ship"   );
```
//Load the model "ship" from the content pipeline of the game.
```
//skyModel just has 1 effect whereas shipModel has 3 effects.
skyEffect = (BasicEffect)skyModel .Meshes[0].Effects[0];
```
//The one and only effect of "skybox".

---

Event handler `protected override void UnloadContent()` inside `public class Game1`
// This function is symmetrical to `LoadContent()`

---

`content.Unload();` //Free the memory space occupied by the "dice" model.

Update method `protected override void Update( GameTime gameTime )` = Game State Manager inside `public class Game1`

// This function is derived from a virtual function `protected virtual void Update( GameTime gameTime )` of the parent class `Microsoft.Xna.Framework.Game` which calls this function automatically as often as possible together with the `protected override void Draw( GameTime gameTime )`-function below.
Here is the right place to check the player's actions and to respond to his movements, collisions etc.

---

```
GamePadState s = GamePad.GetState( PlayerIndex.One );
```
//Get the status of player's 1 controller. (`PlayerIndex.One` just represents value `0`.)

---

`if ( !s.IsConnected ) { Exit(); }` //Stop the game if there is no XBox 360 controller.

---

```
ButtonState bp = ButtonState.Pressed; //bp is just an acronym for ButtonState.Pressed.
if ( s.Buttons.Back == bp ) //Back-Button = Reset
{ shipPos = Vector3.Zero; //Reset the ship to its initial positions in the middle of the sky box.
  cameraPos = new Vector3( 0f, 500f,  3000f );
  //Reset the camera to its initial position above and behind the ship.
  camLookAt = new Vector3( 0f, 500f, -3000f );
  //Reset the camera to its initial direction coaxial to the ship.
  rotationX = rotationY = rotationZ = 0f; //Reset all rotations to 0f.
}
```

---

```
if ( s.DPad.Left ==bp ) { cameraPos.X-=slow; camLookAt.X-=slow; shipPos.X-=slow; }
//Move camera and ship to the left.
if ( s.DPad.Right==bp ) { cameraPos.X+=slow; camLookAt.X+=slow; shipPos.X+=slow; }
//Move camera and ship to the right.
if ( s.DPad.Up   ==bp ) { cameraPos.Y+=slow; camLookAt.Y+=slow; shipPos.Y+=slow; }
//Move camera and ship upward.
if ( s.DPad.Down ==bp ) { cameraPos.Y-=slow; camLookAt.Y-=slow; shipPos.Y-=slow; }
//Move camera and ship downward.
```

---

```
Vector2 ts = s.ThumbSticks.Left;  rotationZ-=0.1f*ts.X; rotationX-=0.1f*ts.Y;
        ts = s.ThumbSticks.Right; rotationZ-=0.1f*ts.X; rotationY-=0.1f*ts.Y;
```
//Decrement the Z- and X-rotations by 10% of the `ThumbSticks.Left` X- and Y-coordinates.
//Decrement the Z- and Y-rotations by 10% of the `ThumbSticks.Right` X- and Y-coordinates.

---

```
shipworldM = Matrix.CreateTranslation( shipPos ); //Matrix form of the ship movement
skyworldM  = Matrix.CreateScale( skyscale , skyscale , skyscale );
             //Matrix form of the sky box zoom factor
viewM      = Matrix.CreateTranslation( -shipPos.X, -shipPos.Y, -shipPos.Z ) *
             //Before rotation of the camera axis we have to shift the camera to (0,0,0) = middle of skybox.
             Matrix.CreateRotationX( rotationX ) * //Multiply the X-rotation matrix.
             Matrix.CreateRotationY( rotationY ) * //Multiply the Y-rotation matrix.
             Matrix.CreateRotationZ( rotationZ ) * //Multiply the Z-rotation matrix.
             Matrix.CreateLookAt( cameraPos, camLookAt, Vector3.Up ) * shipworldM;
             //Multiply the old camera axis and multiply the ship movement matrix
               which shifts the camera back from (0,0,0) to its old position above and behind the ship.
projM = Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/3, 1f, 1f, 10f*skyscale);
//Matrix form of the viewing frustrum (60 degrees, aspect ratio = 1f, near plane at 1f, far plane at 10f*SkyZoom).
```

---

```
g.GraphicsDevice.RenderState.CullMode = CullMode.None;
//In order to save GPU time, the backsides of the polygons are normally black.
//Since our camera is inside the box, normal culling would result in nothing to shade.
//CullingMode.None advices the GPU to render both faces of the polygons.
//g.GraphicsDevice.RenderState.DepthBufferEnable = false;
//g.GraphicsDevice.RenderState.FillMode = FillMode.WireFrame;
// Try out what happens when there is no Z-buffer test.
// Try out how the model looks as wire frame.
```

---

`base.Update(gameTime);` //Call the base entity `Update()`-method which will check the players actions.

---

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 together with the `protected override void Update( GameTime gameTime )`-function above.
Here is the right place to apply the transformation matrices
and to show the one and only effect of the skybox and the 3 effects of the ship.

---

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

---

```
skyEffect.World;      = skyworldM; //Apply the sky zoom matrix.
skyEffect.View;       = viewM;     //Apply the camera matrix.
skyEffect.Projection = projM;      //Apply the frustrum matrix.
skyModel .Meshes[0].Draw(); //Render the sky box.
```

---

```
foreach ( BasicEffect effect in shipModel.Meshes[0].Effects )
//Iterate through all 3 effects of model "ship".
{ effect.EnableDefaultLighting();     //Switch on some default ambient and directional light.
  effect.World;         = shipworldM; //Apply the ship transform matrix.
  effect.View;          = viewM;      //Apply the camera matrix.
  skyEffect.Projection = projM;       //Apply the frustrum matrix.
  skyModel .Meshes[0].Draw();         //Render the ship effect.
}
```

---