

Course 3D_XNA: 3D-Computer Graphics with XNA

Chapter C5: Dice with Texture

Copyright © by V. Miszalok, last update: 05-12-2008

- ↓ [XBox 360 Controller](#)
- ↓ [Project dice1](#)
- ↓ [The Complete Code of `Game1.cs`](#)
- ↓ [The Complete Data of `dice.x`](#)
- ↓ [Experiments](#)

Chapter 5 demonstrates:

- How to write an X-file describing the 6 faces of a dice, describing materials, faces and normals, by `Vertex Buffer`, `Index Buffer`, `MeshMaterialList`, `MeshTextureCoords` and `MeshNormals`.
- How to arrange and orientate the six faces properly.
- How to load the dice model.
- How to scale, rotate and translate the model via the XBox 360 controller.
- How to draw the model with its six effects.

Chapter 5 doesn't demonstrate:

- How to program light sources.
- How to prevent the camera from penetrating the dice.

XBox 360 Controller



Plug Your XBox 360 Controller into an USB-port and check it: Start → Settings → Gamecontroller → Controller (XBOX 360 For Windows) → Properties → Test.

Project dice1

1. Main Menu after starting VS 2008: File → New Project... → Project types: XNA GameStudio 3.0 → Templates: Windows Game 3.0 → Name: `dicel` → Location: `C:\temp` → Create directory for solution: switch it off → OK
Solution Explorer - `dicel`: Delete the file `Program.cs` and the content 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 + in front of `dicel`. A tree opens. Look for the branch "References". Click the + in front of References. Check if there are (among others) these four references: `Microsoft.XNA.Framework` and `Microsoft.XNA.Framework.Game` and `microsoftlib` and `System`.
3.
Right click this link: [front.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [back.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [top.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [bottom.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [right.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [left.bmp](#) and store it in the project directory `C:\temp\dicel\Content`.
Right click this link: [dice.x](#) and store it in the project directory `C:\temp\dicel\Content`.
4. Now You have to add these six images and the Mesh file to project dicel:
Right click the branch `dicel` → Content → Add → Existing Item... →
Files of type: All Files (*.*)
Select the 7 imported files, quit by clicking the Add-button and check whether they arrived underneath the branch `dicel` → Content.
5. The 6 images are automatically compiled from `*.bmp` to `*.xnb` when `dice.x` is compiled to `dice.xnb`. For this reason they should not compile on their own. Change their default Build Action-properties "Compile" to "None".

The Complete Code of Game1.cs

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

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

static class Program
{ [STAThread] static void Main() { Game1 game = new Game1(); game.Run(); }

public class Game1 : Game
{ private GraphicsDeviceManager g;
  private ContentManager content;
  private Model model;
  private Matrix world, view, proj;
  private float positionX = 0.0f, positionY = 0.0f, positionZ = 0.0f;
  private float scaleX = 1.0f, scaleY = 1.0f, scaleZ = 1.0f;
  private float rotationX = 0.5f, rotationY = 0.5f, rotationZ = 0.0f;

  public Game1()
  { g = new GraphicsDeviceManager( this );
    content = new ContentManager( Services );
  }
  protected override void Initialize()
  { g.PreferredBackBufferWidth = 400;
    g.PreferredBackBufferHeight = 400;
    g.ApplyChanges();
    Window.Title = "Dice";
    g.IsFullScreen = false;
    Window.AllowUserResizing = true;
    base.Initialize();
  }
  protected override void LoadContent()
  { model = content.Load< Model >( "Content\\dice" );
  }
  protected override void UnloadContent()
  { content.Unload();
  }
  protected override void Update( gameTime )
  { GamePadState s = GamePad.GetState( PlayerIndex.One );
    if ( !s.IsConnected ) Exit();
    Vector2 ts = s.ThumbSticks.Left;
    rotationY += 0.05f * ts.X;
    rotationX -= 0.05f * ts.Y;
    ts = s.ThumbSticks.Right;
    rotationY += 0.05f * ts.X;
    positionZ -= 0.1f * ts.Y;
    if ( s.DPad.Left == ButtonState.Pressed ) positionX -= 0.05f;
    if ( s.DPad.Right == ButtonState.Pressed ) positionX += 0.05f;
    if ( s.DPad.Up == ButtonState.Pressed ) positionY += 0.05f;
    if ( s.DPad.Down == ButtonState.Pressed ) positionY -= 0.05f;
    if ( s.Buttons.Y == ButtonState.Pressed ) scaleY *= 1.01f;
    if ( s.Buttons.A == ButtonState.Pressed ) scaleY *= 0.99f;
    if ( s.Buttons.X == ButtonState.Pressed ) scaleX *= 0.99f;
    if ( s.Buttons.B == ButtonState.Pressed ) scaleX *= 1.01f;
    if ( s.Buttons.Back == ButtonState.Pressed )
    { positionX = positionY = positionZ = 0.0f;
      scaleX = scaleY = scaleZ = 1.0f;
      rotationX = rotationY = rotationZ = 0.0f;
    }
    world = Matrix.CreateScale( scaleX, scaleY, scaleZ ) *
      Matrix.CreateRotationX( rotationX ) *
      Matrix.CreateRotationY( rotationY ) *
      Matrix.CreateRotationZ( rotationZ ) *
      Matrix.CreateTranslation( positionX, positionY, positionZ );
    view = Matrix.CreateLookAt( new Vector3(0f, 0f, 4f), Vector3.Zero, Vector3.Up );
    proj = Matrix.CreatePerspectiveFieldOfView( MathHelper.Pi/4, 1f, 0.1f, 200f );
    g.GraphicsDevice.RenderState.CullMode = CullMode.None;
    base.Update( gameTime );
  }
}
```

```

protected override void Draw( GameTime gameTime )
{
    g.GraphicsDevice.Clear( Color.DarkBlue );
    foreach ( BasicEffect effect in model.Meshes[0].Effects )
    {
        effect.EnableDefaultLighting();
        effect.World      = world;
        effect.View       = view;
        effect.Projection = proj;
        model.Meshes[0].Draw();
    }
}
} // end of class Game1
} // end of class Program

```

Debug → Start Without Debugging.

Try out both ThumbSticks, the DPad and the buttons Y, A, X, B and Back.

The Complete Data of dice.x

It's surprisingly complicated to write an x-File for a textured box.

The main problems are:

- 1) A textured box needs 24 vertices.
- 2) The arrangement of the vertices inside the VertexBuffer and the IndexBuffer determine the orientation of any single face. It's tricky to fit the borders of the 6 faces in a consistent order.
- 3) The MeshTextureCoords-List must carefully match the order of the VertexBuffer and the IndexBuffer.
- 4) The MeshNormals-List must contain oblique vectors normalized to length 1.0.
- 5) Widths and heights of the textured images must be powers of 2: 32,64,128,256 etc.
- 6) Only *.bmp and *.jpg image file formats are allowed.

This is the listing of dice.x:

```

xof 0302txt 0064
Material frontMaterial {
1.0; 1.0; 1.0; 0.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "front.bmp"; }
}
Material backMaterial {
1.0; 1.0; 1.0; 0.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "back.bmp"; }
}
Material topMaterial {
1.0; 1.0; 1.0; 0.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "top.bmp"; }
}
Material bottomMaterial {
1.0; 1.0; 1.0; 0.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "bottom.bmp"; }
}
Material rightMaterial {
1.0; 1.0; 1.0; 0.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "right.bmp"; }
}
Material leftMaterial {
1.0; 1.0; 1.0; 1.0;; // R = 1.0, G = 1.0, B = 1.0
0.0;
0.0; 0.0; 0.0;;
0.0; 0.0; 0.0;;
TextureFilename { "left.bmp"; }
}

```

```

Mesh dice {
//VertexBuffer
//A dice has 8 vertices. The problem is that each vertex needs 3 texture coordinates,
//because any vertex margins 3 textures needing different texture coordinates.
//For this reason any vertex has to be defined triply = 24 vertices.
//We order the vertices by faces and surround any face clockwise starting at its left upper corner.
//Center of the coordinate system = center of the dice = Vector3.Zero.
//Left-handed coordinate system = Y-axis upwards = Vector3.Up.
24;
-1.0; 1.0; -1.0;, //front
1.0; 1.0; -1.0;,
1.0; -1.0; -1.0;,
-1.0; -1.0; -1.0;,
1.0; 1.0; 1.0;, //back
-1.0; 1.0; 1.0;,
-1.0; -1.0; 1.0;,
1.0; -1.0; 1.0;,
-1.0; 1.0; 1.0;, //top
1.0; 1.0; 1.0;,
1.0; 1.0; -1.0;,
-1.0; 1.0; -1.0;,
-1.0; -1.0; -1.0;, //bottom
1.0; -1.0; -1.0;,
1.0; -1.0; 1.0;,
-1.0; -1.0; 1.0;,
1.0; 1.0; -1.0;, //right
1.0; 1.0; 1.0;,
1.0; -1.0; 1.0;,
1.0; -1.0; -1.0;,
-1.0; 1.0; 1.0;, //left
-1.0; 1.0; -1.0;,
-1.0; -1.0; -1.0;,
-1.0; -1.0; 1.0;;
//IndexBuffer
6; // 6 faces
4; 0, 1, 2, 3;,
4; 4, 5, 6, 7;,
4; 8, 9,10,11;,
4;12,13,14,15;,
4;16,17,18,19;,
4;20,21,22,23;;

MeshMaterialList {
6; // No. of materials = no. of faces
6; // A material for each face
0, 1, 2, 3, 4, 5;;
{frontMaterial}
{backMaterial}
{topMaterial}
{bottomMaterial}
{rightMaterial}
{leftMaterial}
}

MeshTextureCoords {
24;
0.0; 0.0; //front
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;
0.0; 0.0; //back
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;
0.0; 0.0; //top
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;
0.0; 0.0; //bottom
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;
0.0; 0.0; //right
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;
0.0; 0.0; //left
1.0; 0.0;
1.0; 1.0;
0.0; 1.0;;
}

```

```

MeshNormals {
24;
//As with the vector of directional light,
//normals should be normalized to length = 1.0.
//If the the length is > 1.0 the scalar product of light*normal give an augmented cosinus(alpha),
//which produces exaggerated local brightness.
//Explanation of value 0.577: 1.0 / Sqrt(3.0) = 0.577.
//Pythagorean theorem: Length of a 3D-vector = Sqrt(dx*dx + dy*dy + dz*dz).
//With dx=dy=dz=0.577 Sqrt(dx*dx + dy*dy + dz*dz) is 1.0.
//Any vertex needs a normal. Because the zero-point lies in the mid of the dice,
//we have just to replace all coordinates 1.0 by 0.577 and all -1.0 by -0.577.
-0.577; 0.577; -0.577; , //front
 0.577; 0.577; -0.577; ,
 0.577; -0.577; -0.577; ,
-0.577; -0.577; -0.577; ,
 0.577; 0.577; 0.577; , //back
-0.577; 0.577; 0.577; ,
-0.577; -0.577; 0.577; ,
 0.577; -0.577; 0.577; ,
-0.577; 0.577; 0.577; , //top
 0.577; 0.577; 0.577; ,
 0.577; 0.577; -0.577; ,
-0.577; 0.577; -0.577; ,
-0.577; -0.577; -0.577; , //bottom
 0.577; -0.577; -0.577; ,
 0.577; -0.577; 0.577; ,
-0.577; -0.577; 0.577; ,
 0.577; 0.577; -0.577; , //right
 0.577; 0.577; 0.577; ,
 0.577; -0.577; 0.577; ,
 0.577; -0.577; -0.577; ,
-0.577; 0.577; 0.577; , //left
-0.577; 0.577; -0.577; ,
-0.577; -0.577; -0.577; ,
-0.577; -0.577; 0.577; ;
6;
4; 0, 1, 2, 3; ,
4; 4, 5, 6, 7; ,
4; 8, 9,10,11; ,
4;12,13,14,15; ,
4;16,17,18,19; ,
4;20,21,22,23; ;
} //end of MeshNormals {
} //end of Mesh dice {

```

Experiments

1. Move inside the cube by canting the right ThumbStick down.
2. Comment out the line "device.RenderState.CullMode = CullMode.None;" inside the protected override void Update(gameTime gameTime)-function and move again into the interior of the cube.
3. Comment out the line "graphics.GraphicsDevice.Clear(Color.DarkBlue);" inside the protected override void Draw(gameTime gameTime) event handler function. Observe how the Z-buffer prevents from writing into the back buffer.
4. Comment out the line "effect.EnableDefaultLighting();" inside the protected override void Draw(gameTime gameTime)-function.
5. Replace the line "foreach (BasicEffect effect in model.Meshes[0].Effects)" inside the protected override void Draw(gameTime gameTime)-function by "BasicEffect effect = (BasicEffect)model.Meshes[0].Effects[0];".
Then replace the "0" of Effects[0] by "1", by "2" etc.
6. Try out other texture images.
7. Switch off the directional light by deleting the complete MeshNormals { .. } section from dice.x.
8. Change the first material from white to red color:

```

Material frontMaterial {
1.0; 1.0; 1.0; 0.0; // R = 1.0, G = 1.0, B = 1.0
-
Material frontMaterial {
1.0; 0.0; 0.0; 0.0; // R = 1.0, G = 0.0, B = 0.0

```

9. Reduce the six materials to one by changing:

```

MeshMaterialList {
6; // No. of materials = no. of faces
6; // A material for each face
0, 1, 2, 3, 4, 5; ;
-
MeshMaterialList {
6; // No. of materials = no. of faces
6; // One material for all faces
0, 0, 0, 0, 0, 0; ;

```