Course 3D_XNA: 3D-Computer Graphics with XNA
Chapter C2: Light and Texture

Project light1

It requires no DirectX SDK but:
1. Visual C# 2008 Express or Visual Studio 2008 Professional
3. XNA Game Studio 3.0
4. DirectX 9.0c End User Runtime

Main Menu after starting VS 2008: `File` → `New Project...` → `Templates: Windows Game (3.0)` → `Name: light1` → `Location: C:\temp` → `Create directory for solution:` switch it off → `OK`.
`Solution Explorer - light1`: Delete the file `Program.cs` and the content of `Game1.cs`.

Game1, Initialize, Update, Draw

Write the following code into the empty code window of `Form1.cs`:

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

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;
  private const int xsteps = 100;

  public Game1()
  { Window.Title = "3DTriangleAnimation";
    Window.AllowUserResizing = true;
    g = new GraphicsDeviceManager( this );
    g.IsFullScreen = false;
    g.PreferredBackBufferWidth  = 600;
    g.PreferredBackBufferHeight = 600;
  }
```

```
  protected override void Initialize()
  { VertexPositionNormalTexture[] v = new VertexPositionNormalTexture[xsteps];
    float  amplitude = 0.2f;
    double frequency = 4.0 * MathHelper.TwoPi;
    for ( int i = 0; i < xsteps; i++ )
    { float x = (float)i / xsteps;
      v[i].Position.X          = -0.5f + x;
      v[i].TextureCoordinate.X =         x;
      if ( i%2 == 0 ) { v[i].Position.Y =  0.5f; v[i].TextureCoordinate.Y = 0f; }
      else            { v[i].Position.Y = -0.5f; v[i].TextureCoordinate.Y = 1f; }
      v[i].Position.Z = amplitude * (float)System.Math.Sin( x*frequency );
    }
    for ( int i = 0; i < xsteps-2; i++ )
    { Vector3 a = v[i+1].Position - v[i].Position;
      Vector3 b = v[i+2].Position - v[i].Position;
      Vector3 c = Vector3.Cross( a, b );
      c.Normalize();
      if ( c.Z > 0f ) v[i].Normal = c;
      else            v[i].Normal = Vector3.Negate( c );
    }
    vbuffer = new VertexBuffer( g.GraphicsDevice, typeof(VertexPositionNormalTexture),
                                xsteps, BufferUsage.WriteOnly );
    vbuffer.SetData< VertexPositionNormalTexture >( v );
    effect = new BasicEffect( g.GraphicsDevice, null );
    effect.View = Matrix.CreateLookAt( new Vector3(0, 0, 4), Vector3.Zero, Vector3.Up );
    effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/8, 1f, 1f, 10f);
    effect.LightingEnabled = true;
    //effect.Texture = new ContentManager(Services).Load< Texture2D >("Content\\Foto");
    //effect.TextureEnabled = true;
    pass = effect.CurrentTechnique.Passes[0];
    BasicDirectionalLight light = effect.DirectionalLight0;
    light.Direction = new Vector3( 0f, 0f, -1f );
    light.Direction.Normalize();
    light.DiffuseColor  = Color.White.ToVector3();
    light.SpecularColor = Color.CornflowerBlue.ToVector3();
    light.Enabled = true;
  }
  protected override void Update( GameTime gameTime )
  { g.GraphicsDevice.Vertices[0].SetSource( vbuffer, 0, VertexPositionNormalTexture.SizeInBytes );
    g.GraphicsDevice.VertexDeclaration =
      new VertexDeclaration(g.GraphicsDevice, VertexPositionNormalTexture.VertexElements);
    g.GraphicsDevice.RenderState.CullMode = CullMode.None;
    //g.GraphicsDevice.RenderState.FillMode = FillMode.WireFrame;
    g.GraphicsDevice.RenderState.DepthBufferEnable      = false;
    g.GraphicsDevice.RenderState.DepthBufferWriteEnable = false;
  }
  protected override void Draw( GameTime gameTime )
  { g.GraphicsDevice.Clear( Color.DarkBlue );
    //effect.World = Matrix.CreateRotationZ( rotAngle += 0.01f );
    //effect.World = Matrix.CreateRotationY( rotAngle += 0.01f );
    //effect.World = Matrix.CreateRotationX( rotAngle += 0.01f );

    effect.Begin();
      pass.Begin();
        g.GraphicsDevice.DrawPrimitives( PrimitiveType.TriangleStrip, 0, xsteps-2 );
      pass.End();
    effect.End();
    base.Draw(gameTime);
  }
} // end of class Game1
} // end of class Program
```

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

Experiments

1.	Remove the comment slashes `//` in front of `//g.GraphicsDevice.RenderState.FillMode = FillMode.WireFrame;`.	No filling, just border lines.
2.	Keeping `FillMode.WireFrame` change `const int xsteps = 100;` from `100` to `20` and to `300`.	Few or many border lines.
3.	Change `float amplitude = 0.2f;` from `0.2f` to `0.05f`, `0.0f`, `0.5f`, `1f`.	Flatter or steeper waves.
4.	Change `double frequency = 4.0 * MathHelper.TwoPi;` from `4.0` to `2.0` and `8.0f`.	Less or more waves.
5.	Set comment slashes `//` in front of `c.Normalize();` and `light.Direction.Normalize();`.	No visible change. Seem not to be important.
6.	Change `effect.View = Matrix.CreateLookAt(new Vector3(0,0,3), Vector3.Zero, Vector3.Up);` from `new Vector3(0,0,4)` to `new Vector3(0,0,8)` and to `new Vector3(0,0,3)`.	The camera moves backward and forward.
7.	Change the 1. parameter of `effect.Projection=Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/8,1f,1f,10f);` from `MathHelper.Pi/8` to `MathHelper.Pi/4`.	The camera angle widens from 45° to 90°. The wave object seems smaller.
8.	Change the 2. parameter of `effect.Projection=Matrix.CreatePerspectiveFieldOfView(MathHelper.Pi/8,1f,1f,10f);` from `1f` to `0.5f` and to `2f`.	Horizontal/vertical distortion.
9.	Change the 3. parameter of `light.Direction = new Vector3(0f, 0f, -1f);` from `-1f` to `1f`.	Black, because the light direction goes from back to front now.
10.	Change the 3. parameter of `light.Direction = new Vector3(0f, 0f, -1f);` from `-1f` to `-2f`, `-3f`, `-5f`.	Brighter and brighter hillsides.
11.	Change the 1. parameter of `light.Direction = new Vector3(0f, 0f, -1f);` from `0f` to `-1f`.	Light goes from right to left. → Bright right and dark left hillsides.
12.	Change `light.DiffuseColor = Color.White.ToVector3();` from `Color.White.ToVector3()` to `Color.Green.ToVector3()`.	Green
13.	Change `light.SpecularColor = Color.CornflowerBlue.ToVector3();` from `Color.CornflowerBlue.ToVector3();` to `Color.White.ToVector3();`.	Brightness at the summits.
14.	Set comment slashes `//` in front of `light.SpecularColor = Color.CornflowerBlue.ToVector3();`.	The summits darken.

15.	Remove the comment slashes `//` in front of one of the following lines:
	`//effect.World = Matrix.CreateRotationZ(rotAngle+=0.01f);`
	`//effect.World = Matrix.CreateRotationY(rotAngle+=0.01f);`
	`//effect.World = Matrix.CreateRotationX(rotAngle+=0.01f);`
		3 different rotation animations.

Texture

1. Remove all experimental changes and restore the original code from Game1, Initialize, Update, Draw above.

2. Right click this link: **Foto.jpg** and store the image into the project directory `C:\temp\light1`.
Now You have to add the texture image to project `light1`:
Right click `light1 → Content → Add → Existing Item...` → `Files of type: All Files (*.*)`.
Select `Foto.jpg`, quit by clicking the `Add`-button and check whether it arrived underneath
project `light1 → Content`.

3. Remove the comment slashes `//` in front of
`//effect.Texture = new ContentManager(Services).Load< Texture2D >("Content\\Foto");` and
`//effect.TextureEnabled = true;`

4. Click `Debug` → `Start Without Debugging Ctrl F5`.

Experiments:

1.	Change `v[i].TextureCoordinate.X = x;` in function `protected override void Initialize()` from `x` to `2*x` and to `x/2`.	The image doesn't fit anymore.

2.	Exchange the values of `v[i].TextureCoordinate.Y` in function `protected override void Initialize()`
	`if ( i%2 == 0 ) { v[i].Position.Y =  0.5f; v[i].TextureCoordinate.Y = 0f; }`
	`else { v[i].Position.Y = -0.5f; v[i].TextureCoordinate.Y = 1f; }`
	→
	`if ( i%2 == 0 ) { v[i].Position.Y =  0.5f; v[i].TextureCoordinate.Y = 1f; }`
	`else { v[i].Position.Y = -0.5f; v[i].TextureCoordinate.Y = 0f; }`
		Upside down.

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 2.0 → Getting Started with XNA Game Studio →
Your First Game: Microsoft XNA Game Studio in 2D
2: XNA Game Studio 2.0 → Programming Guide
3: XNA Game Studio 2.0 → XNA Framework Class Library