Course 3D_MDX: 3D-Graphics with Managed DirectX 9.0
Chapter C2: Cylinder with Directional Light

Project lights1

This chapter is a summary of a Direct3D-Tutorial from Microsoft: `Tutorial4`. You find the tutorial here: `C:\DXSDK\Samples\Managed\Direct3D\Tutorials`.

Main Menu after starting VS 2005: `File → New Project... → Templates: Windows Application Name: lights1 → Location: C:\temp → Create directory for solution:` switch it off `→ OK`
Delete the files `Program.cs` and `Form1.Designer.cs` and the content of `Form1.cs`, as descibed in the chapters 2DCisC1 to 2DCisC4.

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 `lights1`. A tree opens. Look for the branch "`References`". **Right**-click `References` and **left**-click `Add Reference...`. An `Add Reference` dialog box opens. Scroll down to the component name: `Microsoft.DirectX Version 1.0.2902.0`.
Highlight this reference by a left-click and (holding the `Strg`-key pressed) the reference `Microsoft.DirectX.Direct3D Version 1.0.2902.0` somewhere below. Quit the `Add Reference` dialog box with `OK`.
Check if both references `Microsoft.DirectX` and `Microsoft.DirectX.Direct3D` are now visible inside the Solution Explorer window underneath `lights1 → References`.

If You use Visual Studio 2005 Professional You should switch off the vexatious automatic format- and indent- mechanism of the code editor before You copy the following code to `Form1.cs` (otherwise all the code will be reformatted into chaos):
1. Main menu of Visual Studio 2005 Professional: click menu "`Tools`".
2. A DropDown-menu appears. Click "`Options...`".
3. An `Options` dialog box appears.
4. Click the branch "`Projects and Solutions`". Click "`General`". Redirect all three paths to `C:\temp`.
5. Click the branch "`Text Editor`", then click "`C#`".
6. A sub-tree appears with the branches "`General, Tabs, Advanced, Formatting, IntelliSense`".
7. Click "`Tabs`". Change "`Indenting`" to `None`, "`Tab size`" and "`Indent size`" to `1` and switch on the option "`Insert spaces`".
8. Inside the sub-tree "`C#`" click the plus-sign in front of "`Formatting`" and change all "`Formatting`"-branches as follows: "`General`": switch off all CheckBoxes, "`Indentation`": switch off all CheckBoxes, "`New Lines`": switch off all CheckBoxes, "`Spacing`": switch off all CheckBoxes, "`Wrapping`": switch **on** both CheckBoxes.
9. Leave the dialog box with button "`OK`".

Form1, OnResize, OnTimer

Write the following code to `Form1.cs`:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  static Device device = null;
  static float fAngle;
  VertexBuffer vertexBuffer;
  const int N = 100; //N must be an even no. 6, 8, 10, etc
  CustomVertex.PositionNormal[] vv = new CustomVertex.PositionNormal[N];
  Timer myTimer = new Timer();
```

```
public Form1()
{ Text = "D3DLights";
  //TriangleStrip forming a cylinder
  //radius = 1; axis = Z-axis; top = 1; bottom = -1; => height = 2;
  //in order to see the vertices, replace the TriangleStrip by LineStrip in OnTimer(...)
  float arcus_increment = (float)( 2 * Math.PI / (N-2) );
  Vector3 v = new Vector3();
  for (int i = 0; i < N; i++) //Fill up coordinates and normal vectors
  { float arcus = i * arcus_increment;
    v.X = (float)Math.Cos( arcus );
    v.Y = (float)Math.Sin( arcus );
    if ( i%2 == 0 ) v.Z =  1f;
    else            v.Z = -1f; //zigzag between top and bottom
    vv[ i ].Position = v; //vertex = (cos,sin,+1) or (cos,sin,-1)
    v.Z =  0;             //cylinder normals are parallel to the xy-plane
    vv[ i ].Normal   = v; //normal = (cos,sin,0)
  }
  //set up the timer
  myTimer.Tick += new EventHandler( OnTimer );
  myTimer.Interval = 1;
  ClientSize = new Size( 400, 300 ); //Calls OnResize( ... )
}

protected override void OnResize( System.EventArgs e )
//Whenever the window changes we have to initialize Direct3D from scratch
{ myTimer.Stop();// stop the timer during initialization
  try
  { //get information from the operating system about its current graphics properties
    PresentParameters presentParams = new PresentParameters();
    //we have to set two flags
    presentParams.Windowed = true;                 //no full screen display
    presentParams.SwapEffect = SwapEffect.Discard; //no swap buffer
    presentParams.EnableAutoDepthStencil = true;   //with depth buffer
    presentParams.AutoDepthStencilFormat = DepthFormat.D16; //16 bit depth
    //Create a new D3D-device that serves as canvas.
    if ( device != null ) device.Dispose(); //free the old canvas if any
    device = new Device( 0, DeviceType.Hardware, this,
                         CreateFlags.SoftwareVertexProcessing, presentParams );
    //Create a white material.
    Material mtrl = new Material();
    mtrl.Diffuse = mtrl.Ambient = Color.White;
    device.Material = mtrl;
    //Create a single, white, directional, diffuse light source and a gray ambient light.
    //Many lights may be active at a time. (Each one slows down the render process.)
    device.Lights[0].Type = LightType.Directional;
    device.Lights[0].Diffuse = System.Drawing.Color.DarkTurquoise;
    device.Lights[0].Direction = new Vector3( 1, 1, 5 );
    device.Lights[0].Enabled = true;//turn it on
    //Finally, turn on some ambient light that scatters and lights the object evenly
    device.RenderState.Ambient = System.Drawing.Color.FromArgb( 0x202020 );
    //set up the transformation of world coordinates into camera or view space
    device.Transform.View = Matrix.LookAtLH(
      new Vector3( 0f, 0f,-4f ),   //eye point 4.0 in front of the canvas
      new Vector3( 0f, 0f, 0f ),   //camera looks at point 0,0,0
      new Vector3( 0f, 1f, 0f ) ); //world's up direction is the y-axis
    //set up the projection transformation using 4 parameters:
    //1.: field of view = 45 degrees; 2.: aspect ratio=heigth/width = 1 = square window;
    //3.: near clipping distance = 1; 4.: far clipping distance = 10;
    device.Transform.Projection = Matrix.PerspectiveFovLH( (float)Math.PI/4,1f,1f,10f );
    device.RenderState.CullMode = Cull.None;
    device.RenderState.Lighting = true;
    device.VertexFormat = CustomVertex.PositionNormal.Format;
    //create a new vertex buffer and connect it to the device
    if ( vertexBuffer != null ) vertexBuffer.Dispose();//free the old vertexBuffer if any
    vertexBuffer = new VertexBuffer( typeof(CustomVertex.PositionNormal), N,
                                     device, 0, CustomVertex.PositionNormal.Format,
                                     Pool.Default );
    //copy the coordinates and colors of "vv" into the vertex buffer
    vertexBuffer.SetData( vv, 0, LockFlags.None );
    device.SetStreamSource( 0, vertexBuffer, 0 );
    myTimer.Start();//start the timer again
  }
  catch (DirectXException) { MessageBox.Show("Could not initialize Direct3D."); return; }
}
```

```
  protected static void OnTimer( Object myObject, EventArgs myEventArgs )
  { if (device == null) return;
    //throw the old image away
    device.Clear( ClearFlags.Target | ClearFlags.ZBuffer, Color.Blue, 1f, 0 );
    //rotate with an angular velocity = 5.7°/timer event
    fAngle += 0.1f;
    device.Transform.World = Matrix.RotationAxis( new Vector3(1, 1, 1), fAngle );
    //draw on the canvas
    device.BeginScene();
      device.DrawPrimitives( PrimitiveType.TriangleStrip, 0, N-2 );
      //Experiment: Replace the TriangleStrip by a LineStrip as follows:
      //device.DrawPrimitives( PrimitiveType.LineStrip, 0, N-2 );
    device.EndScene();
    device.Present(); //show the canvas
  }
}
```

Click `Debug` → `Start Without Debugging Ctrl F5`. Try to drag all window borders.

Exercises

1. In the 5th next to the last line of `protected static void OnTimer( Object myObject, EventArgs myEventArgs )` replace `TriangleStrip` by `LineStrip` and try out how the vertex framework changes when You vary the constant `N` (in the 5th line of the class declaration of `Form1`) from 6 to 8, ,12, ,48 etc. to 100.
2. Slow the animation down by increasing `myTimer.Interval = 100;` inside the constructor.
3. Accelerate the animation by increasing `fAngle += 0.5f;` in `OnTimer`.
4. Rotate the cylinder around other axes as `(1,1,1)` in `OnTimer`.
5. Light the cylinder with another diffuse color.
6. Light the cylinder with another ambient color.
7. Vary the position of the light source: from top only, from bottom only, from left only etc.
8. Step back with the eye point from `(0f,0f,-4f)` to `(0f,0f,-100f)` and get closer with `(0f,0f,-2f)`.
9. Read the comments **C3DCisC2_Comment.htm** and try to understand the sense of the code lines.
10. You can find more explanations and comments about this chapter here: **http://msdn.microsoft.com/library/default.asp**.

Caution: Mozilla Firefox doesn't correctly display the tree on the left side. Use the Internet Explorer here !
Click trough the tree on the left side:
Win32 and COM Development → Graphics and Multimedia → DirectX → SDK Documentation →
DirectX SDK Managed → DirectX SDK → Introducing DirectX 9.0 → Direct3D Graphics →
Getting Started with Direct3D → Direct3D Tutorials → Tutorial 4: Using Materials and Lights.