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

Projekt lights1

Diese Übungsaufgabe ist eine kurze, übersichtliche Fassung eines Direct3D-Tutorials von Microsoft: `Tutorial4`. Sie finden es unter `C:\DXSDK\Samples\Managed\Direct3D\Tutorials\Tutorial4`.

Main Menu nach dem Start von VS 2005: `File -> New Project... -> Templates: Windows Application`
`Name: lights1 -> Location: C:\temp -> Create directory for solution:` ausschalten `-> OK`
Löschen Sie die Files `Program.cs` und `Form1.Designer.cs` und den Inhalt von `Form1.cs`, wie es in den Kapiteln 2DCisC1 bis 2DCisC4 beschrieben wurde.

Falls das `Solution Explorer -` Fenster nicht schon offen ist, öffnen Sie es über das Hauptmenü: `View -> Solution Explorer`.
Im `Solution Explorer -` Fenster klicken Sie auf das Pluszeichen vor `lights1`. Es öffnet sich ein Baum. Ein Ast heißt "`References`". Klicken Sie mit der **rechten** Maustaste auf `References` und dann mit der **linken** Maustaste auf `Add Reference...`. Es öffnet sich eine `Add Reference` Dialog Box. Scrollen Sie abwärts, bis Sie den Component Name: `Microsoft.DirectX Version 1.0.2902.0` sehen.
Markieren Sie durch Linksklick diese Referenz und (bei gedrückter der Strg-Taste) die weiter unten stehende Referenz `Microsoft.DirectX.Direct3D Version 1.0.2902.0`. Verlassen Sie die `Add Reference` Dialog Box mit `OK`.
Kontrollieren Sie, ob jetzt im Solution Explorer Fenster unter `lights1 -> References` (unter anderen) die beiden Referenzen `Microsoft.DirectX` und `Microsoft.DirectX.Direct3D` stehen.

Wenn Sie nicht Visual C# Express sondern Visual Studio 2005 Professional benutzen sollten Sie die Nerven tötende Formatier- und Einrückautomatik des Code-Editors ausschalten, bevor Sie den unten vorgegebenen Code durch kopieren nach `Form1.cs` transferieren:
1. Hauptmenu von Visual Studio 2005 Professional: Klick auf Menupunkt "`Tools`".
2. Es erscheint ein DropDown-Menu. Klick auf "`Options...`".
3. Es erscheint eine `Options` Dialog Box.
4. Klick auf den Ast "`Projects and Solutions`". Klick auf "`General`". Alle drei Pfade auf `C:\temp` stellen.
5. Klick auf den Ast "`Text Editor`", dann auf "`C#`".
6. Es erscheint ein Unterbaum mit den Ästen "`General, Tabs, Advanced, Formatting, IntelliSense`".
7. Klick auf "`Tabs`". Stellen Sie "`Indenting`" auf `None`, "`Tab size`" und "`Indent size`" auf `1` und schalten Sie die Option "`Insert spaces`" ein.
8. Klicken sie im Unterbaum "`C#`" auf das Pluszeichen vor "`Formatting`" und ändern Sie alle "`Formatting`"-Äste:
"`General`": alle CheckBoxes ausschalten, "`Indentation`": alle CheckBoxes ausschalten, "`New Lines`": alle CheckBoxes ausschalten, "`Spacing`": alle CheckBoxes ausschalten, "`Wrapping`": beide CheckBoxes **ein**schalten.
9. Verlassen Sie den Dialog mit Button "`OK`".

Form1, OnResize, OnTimer

Schreiben in das leere Codefenster `Form1.cs` folgenden Code:

```
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 a 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. (Notice: 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
  }
}
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Erproben Sie das Programm durch ziehen an allen Fensterrändern.

Weitere Aufgaben

1. Ersetzen Sie in der 5.-letzte Zeile von `protected static void OnTimer( Object myObject, EventArgs myEventArgs )` den Bezeichner `TriangleStrip` durch `LineStrip` und erproben Sie, wie das Vertexgerüst dichter wird, wenn Sie die Konstante `N` (in der 5. Zeile der Klassendeklaration von `Form1`) von 6 auf 8, auf 12, auf 48 etc. bis auf 100 hoch setzen.
2. Bremsen Sie die Animation durch hoch setzen von `myTimer.Interval = 100;` im Konstruktor.
3. Beschleunigen Sie die Animation durch hoch setzen von `fAngle += 0.5f;` in `OnTimer`.
4. Drehen Sie die Röhre um eine andere Achse als `(1,1,1)` in `OnTimer`.
5. Geben Sie dem diffusen Licht eine andere Farbe.
6. Geben Sie dem ambienten Licht eine andere Farbe.
7. Ändern Sie die Position der Lichtquelle: nur von oben, nur von unten, nur von links etc.
8. Treten Sie mit dem Augenpunkt weiter zurück von `( 0f, 0f,-4f )` nach `( 0f, 0f,-100f )` und näher heran mit `( 0f, 0f,-2f )`.
9. Lesen Sie die Kommentare und versuchen Sie, den Sinn der Befehle zu verstehen.
10. Sie finden Erklärungen und Kommentare zu dieser Übung unter
http://msdn.microsoft.com/library/default.asp.
Verzweigen Sie in dem Baum auf der linken Seite auf:
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.