Course 3DCis: 3D-Computer Graphics with C# and DirectX 9.0
Chapter C4: Standard Meshes = Primitives

Projekt mesh_primitive1

Main Menu nach dem Start von VS 2005: `File -> New Project... -> Templates: Windows Application`
`Name: mesh_primitive1 -> 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 `mesh_primitive1`. 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 beiden weiter unten stehende Referenzen
`Microsoft.DirectX.Direct3D  Version 1.0.2902.0` und
`Microsoft.DirectX.Direct3DX Version 1.0.2902.0` oder `1.0.2903.0` oder `1.0.2904.0`.
Verlassen Sie die `Add Reference` Dialog Box mit `OK`.
Kontrollieren Sie, ob jetzt im Solution Explorer Fenster unter `mesh_primitive1 -> References` (unter anderen) die drei Referenzen
`Microsoft.DirectX` und
`Microsoft.DirectX.Direct3D` und
`Microsoft.DirectX.Direct3DX` stehen.

Form1, OnResize, OnTimer

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

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

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  static Device device = null;
  static float xAngle, yAngle, zAngle ;
  static Mesh meshPolygon, meshBox, meshSphere, meshTorus, meshCylinder, meshTeapot, meshText;
  static Matrix mPol = Matrix.Translation( new Vector3(    0f,    0f, -1.5f ) ); //polygon
  static Matrix mBox = Matrix.Translation( new Vector3(  1.5f,    0f,    0f ) ); //box
  static Matrix mSph = Matrix.Translation( new Vector3( -1.5f,    0f,    0f ) ); //sphere
  static Matrix mTor = Matrix.Translation( new Vector3(    0f,  1.5f,    0f ) ); //torus
  static Matrix mCyl = Matrix.Translation( new Vector3(    0f, -1.5f,    0f ) ); //cylinder
  static Matrix mTea = Matrix.Translation( new Vector3(    0f,    0f,  1.5f ) ); //teapot
  static Matrix mTex;                                                            //text
  Timer myTimer = new Timer();

  public Form1()
  { Text = "Primitive Meshes";
    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 );
      if ( meshPolygon  != null ) meshPolygon .Dispose(); //free the old mesh if any
      if ( meshBox      != null ) meshBox     .Dispose(); //free the old mesh if any
      if ( meshSphere   != null ) meshSphere  .Dispose(); //free the old mesh if any
      if ( meshTorus    != null ) meshTorus   .Dispose(); //free the old mesh if any
      if ( meshCylinder != null ) meshCylinder.Dispose(); //free the old mesh if any
      if ( meshTeapot   != null ) meshTeapot  .Dispose(); //free the old mesh if any
      if ( meshText     != null ) meshText    .Dispose(); //free the old mesh if any
      meshPolygon  = Mesh.Polygon ( device, 0.3f, 8 ); //line length + no of vertices
      meshBox      = Mesh.Box     ( device, 0.5f, 0.5f, 0.5f ); //xSize, ySize, zSize
      meshSphere   = Mesh.Sphere  ( device, 0.5f, 20, 20 ); //radius, no slices, no stacks
      meshTorus    = Mesh.Torus   ( device, 0.2f, 0.4f, 20, 20 );//in+out radii, slices+stacks
      meshCylinder = Mesh.Cylinder( device, 0.5f, 0.2f, 0.8f, 20, 20 ); // radii, length, slices+stacks
      meshTeapot   = Mesh.Teapot  ( device );
      String text = "Size: " + ClientSize.Width .ToString() + "/" +
                               ClientSize.Height.ToString();
      GlyphMetricsFloat[] gly = new GlyphMetricsFloat[text.Length];
      meshText = Mesh.TextFromFont( device,
                                    new System.Drawing.Font( FontFamily.GenericSerif, 12 ),
                                    text, 0.01f, 0.25f, out gly ); //string, smooth, thick, char info
      float x=0f, y=0f;
      for ( int i=0; i < text.Length; i++ )              //for all chars
      { x += gly[i].CellIncX;                            //x=text.Width = sum of char.Width
        if ( gly[i].BlackBoxY > y ) y = gly[i].BlackBoxY; //y=text.Height= max of char.Height
      }
      mTex = Matrix.Translation( new Vector3( -x/2f, -y/2f, 2f ) );
      //set up material with diffuse and ambient white color
      Material myMaterial = new Material();
      myMaterial.Diffuse = myMaterial.Ambient = Color.White;
      device.Material = myMaterial;
      //turn on some blue directional light coaxial to the camera
      device.Lights[0].Type = LightType.Directional;
      device.Lights[0].Diffuse = Color.DarkBlue;
      device.Lights[0].Direction = new Vector3( 0, 0, 5 );
      device.Lights[0].Enabled = true;
      //set up the transformation of world coordinates into camera or view space
      device.Transform.View = Matrix.LookAtLH(
        new Vector3( 0f, 0f, -5f ),   //eye point 5.0 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
      device.Transform.Projection = Matrix.PerspectiveFovLH( (float)Math.PI/4, 1f, 1f, 100f );
      device.RenderState.CullMode = Cull.None; device.RenderState.Lighting = true;
      xAngle = yAngle = zAngle = 0; //start angles
      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.Gray, 1f, 0 );
    //rotate with 3 angular velocities
    Matrix m = Matrix.RotationYawPitchRoll( yAngle += 0.02f, xAngle += 0.02f, zAngle += 0.02f );
    //draw on the canvas
    device.BeginScene();
      device.Transform.World  = m * mPol; meshPolygon .DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = m * mBox; meshBox     .DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = m * mSph; meshSphere  .DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = m * mTor; meshTorus   .DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = m * mCyl; meshCylinder.DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = m * mTea; meshTeapot  .DrawSubset( 0 ); //rotate + translate
      device.Transform.World  = mTex * m; meshText    .DrawSubset( 0 ); //translate + rotate
    device.EndScene(); device.Present(); //show the canvas
  }
}
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Wenn Sie am Fensterrand ziehen, verändert sich der Text.

Weitere Aufgaben

1. Verschieben Sie die Primitiven auf andere Positionen.
2. Legen Sie die 7 Primitiven alle hintereinander auf die z-Achse.
3. Belegen Sie in OnTimer die Rotationsachsen mit anderen Winkelincrementen (etwa 0);
4. Machen Sie sich klar, wie die Matrixmultiplikationen `m * mPol` etc. Rotation und Translation miteinander verknüpfen.
5. Tauschen Sie am Ende des Programms die Reihenfolge von Translation und Rotation durch ändern von `device.Transform.World = mTex * m;` in `device.Transform.World = m * mTex;` und beobachten Sie wie der Text sich nun um sein linke untere Ecke dreht.
6. Lesen Sie die Kommentare und versuchen Sie, den Sinn der Befehle zu verstehen.
7. 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 9.0 SDK Update (August 2005) Managed -> DirectX SDK -> Introducing DirectX 9.0 -> Direct3D Graphics -> Getting Started with Direct3D -> Direct3D Tutorials -> Tutorial 6: Using Meshes`.
8. Über 3D-Text finden Sie weitere Information unter
http://pluralsight.com/wiki/default.aspx/Craig.DirectX/FontBasicsTutorial.html
9. Variieren Sie den 3.letzten Parameter von `meshText = Mesh.TextFromFont( ... )` von `0.01f` zu `1f` und zu `0.00001f` und vergrößern Sie zur Laufzeit das Fenster maximal. Im ersten Fall hat die Schrift wenig Vertices --> wird eckig, im zweiten wird sie fein tesseliert und geschmeidig rund.
10. Variieren Sie den 2.letzten Parameter von `meshText = Mesh.TextFromFont( ... )` von `0.25f` zu `1f` und zu `0f` und vergrößern Sie zur Laufzeit das Fenster maximal. Im ersten Fall sind die Buchstaben tief, im zweiten nur 2D.
11. Probieren Sie andere Texte.