Course 3D_MDX: 3D-Graphics with Managed DirectX 9.0
Chapter C4: Standard Meshes = Primitives

Project mesh_primitive1

Main Menu after starting VS 2005: `File → New Project... → Templates: Windows Application Name: mesh_primitive1 → 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 described in the chapters 2DCisC1 to 2DCisC4.

If You can't find a `Solution Explorer`-window, open it via the main menu: `View → Solution Explorer`. Inside the `Solution Explorer`-window click the plus-sign in front of `mesh_primitive1`. 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) two more references:
`Microsoft.DirectX.Direct3D Version 1.0.2902.0` and
`Microsoft.DirectX.Direct3DX Version 1.0.2902.0` or `1.0.2903.0` or `1.0.2904.0`.
Quit the `Add Reference` dialog box with `OK`.
Check if three references:
`Microsoft.DirectX` and
`Microsoft.DirectX.Direct3D` and
`Microsoft.DirectX.Direct3DX` are now visible inside the Solution Explorer window underneath `mesh_primitive1 → References`.

Form1, OnResize, OnTimer

Write the following code to `Form1.cs`:

```
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,PerCharInfo
    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
    //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 = 0; 4.: far clipping distance = 10;
    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
  }
}
```

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

Exercises

1. Move the primitives to other positions.

2. Put all 7 primitives one behind the other on the z-axis.

3. In `OnTimer` give each rotation axis its own angular increment (try out 0.0).

4. Get clear in Your mind how the matrix-multiplications `m * mPol` etc. connect rotation and translation.

5. At the end of the program change the sequence of translation and rotation by changing
`device.Transform.World = mTex * m;` into
`device.Transform.World = m * mTex;` and realize that the text now rotates around its lower left corner.

6. Read the **Code Comments** and try to understand the sense of the command lines.

7. 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 !
Within the Internet Explorer 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 6: Using Meshes.

8. You can find further information about 3D-text here:
http://pluralsight.com/wiki/default.aspx/Craig.DirectX/FontBasicsTutorial.html

9. Change the 4th parameter of `meshText = Mesh.TextFromFont( ... )` from `0.01f` to `1f` and to `0.00001f` and maximize the window at run time. In the case of `1f` the letters have few vertices -→ angular appearence, in case of `0.00001f` all letters appear smoothly tesselated and curved.

10. Change the second last parameter of `meshText = Mesh.TextFromFont( ... )` from `0.25f` to `1f` and to `0f` and maximize the window at run time. This parameter varies the depth (= z-extension) of the 3D-text from deep to 2D-flat.

11. Try out other text strings.