Course 3D_MDX: 3D-Graphics with Managed DirectX 9.0
Chapter C4, Grid Version: Building a 3D-Grid with Cylinders

Project grid1

This code is the `grid1` variant of the previous `mesh_primitive1` project. It animates a 3D-grid of cylinders.

Main Menu after starting VS 2005: `File → New Project... → Templates: Windows Application Name: grid1 → 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 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 `grid1`. 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 `grid1 → References`.

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 = 0f;
  const int GridSize = 4; //can be any no. between 1 and 10
  static Mesh meshCylinder;
  Timer myTimer = new Timer();

  public Form1()
  { Text = "Grid1";
    myTimer.Tick += new EventHandler( OnTimer );
    myTimer.Interval = 1;
    ClientSize = new Size( 400, 300 ); //Calls OnResize( ... )
  }
```

```
  protected override void OnResize( System.EventArgs e )
  { myTimer.Stop();// stop the timer during initialization
    try
    { PresentParameters presentParams = new PresentParameters();
      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
      if ( device != null ) device.Dispose(); //free the old canvas if any
      device = new Device( 0, DeviceType.Hardware, this,
                           CreateFlags.SoftwareVertexProcessing, presentParams );
      Material mtrl = new Material();
      mtrl.Diffuse = mtrl.Ambient = Color.White;
      device.Material = mtrl;
      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
      device.RenderState.Ambient = System.Drawing.Color.FromArgb( 0x202020 );
      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
      device.Transform.Projection = Matrix.PerspectiveFovLH( (float)Math.PI/4, 1f, 1f, 10f );
      device.RenderState.CullMode = Cull.None;
      device.RenderState.Lighting = true;
      if ( meshCylinder != null ) meshCylinder.Dispose(); //free the old mesh if any
      meshCylinder = Mesh.Cylinder( device, 0.05f, 0.05f, 2f, 10, 2 ); //front+back radii, etc.
      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;
    device.Clear( ClearFlags.Target | ClearFlags.ZBuffer, Color.Blue, 1f, 0 );
    Matrix anim = Matrix.RotationY( fAngle +=0.01f );
    float GridStep = 2f / GridSize;
    device.BeginScene();
      for ( float y = -1f; y <= 1f; y += GridStep )  //z-rods
        for ( float x = -1f; x <= 1f; x += GridStep )
        { device.Transform.World  = Matrix.Translation( new Vector3( x, y, 0f ) ) * anim;
          meshCylinder.DrawSubset( 0 );
        }
      Matrix turn = Matrix.RotationY( (float)Math.PI/2f ); //horiz. rods
      for ( float z = -1f; z <= 1f; z += GridStep )
        for ( float y = -1f; y <= 1f; y += GridStep )
        { device.Transform.World  = turn * Matrix.Translation( new Vector3( 0f, y, z ) ) * anim;
          meshCylinder.DrawSubset( 0 );
        }
      turn = Matrix.RotationX( (float)Math.PI/2f ); //vertical rods
      for ( float z = -1f; z <= 1f; z += GridStep )
        for ( float x = -1f; x <= 1f; x += GridStep )
        { device.Transform.World  = turn * Matrix.Translation( new Vector3( x, 0f, z ) ) * anim;
          meshCylinder.DrawSubset( 0 );
        }
    device.EndScene();
    device.Present();
  }
}
```

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

Exercise: Try out different grids by changing `const int GridSize = 4;` in line 11 from `4` to any arbitrary integer value between `1` and `10`.