Course 3D_MDX: 3D-Graphics with Managed DirectX 9.0
Chapter C7: The Graphics Card Project

Project graphics_card1

This chapter is a short summary of chapter 2 of Tom MILLERs book:Managed DirectX 9 Graphics and Game Programming. SAMS 2003, ISBN: 0-672-32596-9. US$ 34.99.
The program displays the available information of Your computer about its grphics board an its DirectX-driver. This information is necessary for the computer games that normally try to use all the facilities of Your graphics hardware.

Main Menu after starting VS 2005: `File → New Project... → Templates: Windows Application Name: graphics_card1 → Location: C:\temp → Create directory for solution:` switch 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_viewer1`. 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.DirectX3D Version 1.0.2902.0`. Highlight this reference by a left-click and quit the `Add Reference` Dialog Box with `OK`.
Check if the reference `Microsoft.DirectX.Direct3D` is now visible inside the Solution Explorer window underneath `graphics_card1 → References`.

Form1

Write the following code into the empty `Form1.cs`:

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

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  public Form1()
  { string s1 = "", s2= "";
    StringBuilder s = new StringBuilder();
    TextBox TB = new TextBox();
    TB.Size = ClientSize = new Size( 400, 800 );
    TB.Font = new System.Drawing.Font( "Courier New", 8 );
    TB.Multiline  = true;
    TB.ScrollBars = ScrollBars.Both;
    Controls.Add( TB );
    if ( Manager.Adapters.Count > 1 )
      s.Append( "You have two graphics cards or one dual monitor card.\r\n" +
                "The infos are about your primary card.\r\n\r\n" );
    AdapterInformation ai = Manager.Adapters[0];
    s.Append( "Graphic Board: " + ai.Information.Description   + "\r\n" );
    s.Append( "Driver       : " + ai.Information.DriverName    + "\r\n" );
    s.Append( "Driver Vers. : " + ai.Information.DriverVersion + "\r\n\r\n" );
    s.Append( "current display mode:\r\n" );
    s.Append( ai.CurrentDisplayMode.Width .ToString() + "\tx " );
    s.Append( ai.CurrentDisplayMode.Height.ToString() + "\t" );
    s.Append( ai.CurrentDisplayMode.Format.ToString() + "\r\n\r\n" );
    s.Append( s1 + "possible display modes:\r\n" );
```

```
    foreach ( DisplayMode dm in ai.SupportedDisplayModes)
    { s2 = string.Format( "{0}\tx {1}\t{2}\r\n", dm.Width, dm.Height, dm.Format );
      if ( s2 != s1 ) { s.Append( s2 ); s1 = s2; }
    }
    s.Append( "\r\n\r\nCapabilities of this graphics card:\r\n\r\n" );
    Caps caps = Manager.GetDeviceCaps( 0, DeviceType.Hardware );
    s1 = caps.DeviceCaps.ToString();
    s1 = s1.Replace("Caps: ", "Caps:\r\n" );// Insert carriage return and line feed
    s1 = s1.Replace( "\n", "\r\n");          //Insert carriage return
    s.Append( s1 );
    TB.Text = s.ToString();
  }
}
```

Click `Debug` → `Start Without Debugging Ctrl F5`.

More Information

1. Multimon capability: New graphics cards all have dual head support, which allow two monitors to be attached to a single graphics card. Direct3D treats both hookups (regardless if DVI or VGA) as two adapters.

2. "`Manager`" is a static class in the Direct3D assemblies to enumerate adapters, driver info and to retrieve its capabilities.

3. `AdapterInformation` is a structure of structures:

```
public struct AdapterInformation
{ int                 Adapter;               //No. of an adapter in the system
  AdapterDetails      Information;           //structure cont. much adapter/driver info
  AdapterDetails      GetWhqlInformation();  //additional Windows Hardware Quality Labs info
  DisplayMode         CurrentDisplayMode;    //struct. cont. Width, Height, Format, Refresh Rate
  DisplayModeEnumerator SupportedDisplayModes; //collection of DiplayMode structures
}
```

4. The `AdapterInformation.DisplayMode.Format` structure follows this pattern: A letter represents the type of data and the number denotes the number of bits. letters: R=red, G=green, B=blue, X=unused, A=alpha, L=luminance, P=palette. Examples: `X8R8G8B8` denotes a 32 bit color format, `R5G6B5` denotes a 16 bit color format.

5. While the list of formats is quite large, there are only a few that are valid and can be used as a Direct3D display or back buffer format. Valid Direct3D formats are: `X8R8G8B8`, `R5G6B5`, `X1R5G5B5`, `A2R10G10B10`. DirectX3D is not possible when the adapter does not support one of these.

6. You could use the Manager class to find out, whether or not your graphics card supports a particular format. Example: bool b = `Manager.CheckDeviceFormatConversion(0, DeviceType.Hardware, Format.X8R8G8B8, Format.A8R8G8B8, true);` (The 3. and 4. parameter refer to the front buffer and the back buffer. The 5. denotes WindowedMode=`true` or FullScreenMode=`false`)

7. The function `Manager.GetDeviceCaps` generates a `Cap` structure containing every possible capability a graphics card and its driver can have. These hundreds of capabilities are broken down mainly into Boolean values (i.e. feature supported or not) and integer values (i.e. max. no. of features).

8. see: **http://csharp-home.com/index/tiki-read_article.php?articleId=105&page=2**