Microsoft C# Direct3D Tutorial 1: Creating a Device

This is preliminary documentation and is subject to change.
The *CreateDevice* tutorial project initializes Microsoft® Direct3D®, renders a simple blue screen, and eventually shuts down.

Creating an Application Window

The first thing any Microsoft® Windows® application must do when run is to create an application window. To do this, the first call in the *Main()* function of the following sample code is to the constructor of an application-defined *CreateDevice* class that sets the size of the display window, the caption of the window, and the window icon. *CreateDevice* is created from the System.Windows.Forms.Form class used in the Microsoft .NET Framework to represent a window in an application.

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

namespace DeviceTutorial
{
    public class CreateDevice : Form
    {
        // Global variables for this project
        Device device = null; // Rendering device

        public CreateDevice()
        {
        // Set the initial size of form
        this.ClientSize = new System.Drawing.Size(400,300);
        // And its caption
        this.Text = "Direct3D Tutorial 01: CreateDevice";
        // Load icon from the resources of the .exe
        this.Icon = new Icon(this.GetType(), "directx.ico");
        }
    }
}
```

Initializing the Direct3D Object

After you create the application window, you are ready to initialize the Direct3D object that you will use to render the scene. This process includes creating the object, setting the presentation parameters, and finally creating the Direct3D device.

```
public bool InitializeGraphics()
{
    try
    {
    // Create a PresentParameters object
    PresentParameters presentParams = new PresentParameters();

    // Don't run full screen
    presentParams.Windowed = true;

    // Discard the frames
    presentParams.SwapEffect = SwapEffect.Discard;

    // Instantiate a device
    device = new Device(0,
                        DeviceType.Hardware,
                        this,
                        CreateFlags.SoftwareVertexProcessing,
                        presentParams);
    return true;
    }
    catch { return false; }
}
```

The preceding code sample relies upon a PresentParameters object that is used to set the windows display characteristics. For example, by setting the Windowed property to **true**, the size of the window displayed is less than full screen. This default small-window format has no menu or child windows, but it includes Minimize, Maximize, and Close buttons common to windowed applications. In this case, the ability to quickly swap back buffer memory into system memory is disabled with the SwapEffect.Discard flag. If the Windowed property is instead **false**, then the created window is placed above all non-topmost windows and should stay above them, even when the window is deactivated.
The final step in the initialization procedure is to create the Direct3D device. In this example the flags input to the Device(Int32,DeviceType,Control,CreateFlags,PresentParameters) specify that a hardware device is preferred, and that vertex processing is to be done in software. Note that if you tell the system to use hardware vertex processing by specifying CreateFlags.HardwareVertexProcessing, you will see a significant performance gain on video cards that support hardware vertex processing.

Rendering the Direct3D Object

The application is kept running in a loop using the Application.DoEvents method, which here takes as its argument a *CreateDevice* object called *frm*. DoEvents runs a standard Windows application message loop on the current thread.

```
static void Main()
{
    using (CreateDevice frm = new CreateDevice())
    {
        if (!frm.InitializeGraphics()) // Initialize Direct3D
        {
           MessageBox.Show("Could not initialize Direct3D.  This tutorial will exit.");
           return();
        }
        frm.Show();

        // While the form is still valid, render and process messages
        while(frm.Created)
        {
           frm.Render();
           Application.DoEvents();
        }
    }
}
```

While the created *CreateDevice* Form object is valid, an application-defined *Render* method is called to render the Direct3D object. First the viewport (the open window) is set to be a uniform blue color with the Device.Clear method. Scene rendering is begun using the Device.BeginScene method. Rendering is completed and the scene is ended with successive calls to the EndScene and Present methods.

```
private void Render()
{
    if (device == null)
    return;

    //Clear the backbuffer to a blue color (ARGB = 000000ff)
    device.Clear(ClearFlags.Target, System.Drawing.Color.Blue, 1.0f, 0);

    //Begin the scene
    device.BeginScene();

    // Rendering of scene objects can happen here

    //End the scene
    device.EndScene();
    device.Present();
}
```