sign up log in tour help

x Dismiss

Join the Stack Overflow Community

Stack Overflow is a community of 6.7 million programmers, just like you, helping each other. Join them; it only takes a minute:

Sign up

How to control stereo-frames separately with C#? (NVIDIA 3D shutter glasses)

I'm trying to make a very simple application which would display different images on each eye. I have Asus VG236H monitor and NVIDIA 3D Vision kit, the stereo 3D shutter glasses. The I'm using C#, .NET Framework 2.0, DirectX 9 (Managed Direct X) and Visual Studio 2008. I have been searching high and low for examples and tutorials, have actually found a couple and based those I have created the program but for some reason I can't get it working.

When looking for examples how to display different images for each eye, many people keep referring to the NVIDIA presentation at GDC 09 (the famous GDC09-3DVision-The_In_and_Out.pdf document) and the pages 37-40. My code is mainly constructed based on that example:

1. I'm loading two textures (Red.png and Blue.png) on surface (_imageLeft and _imageRight), in function LoadSurfaces()
2. Set3D() function puts those two images side-by-side to one bigger image which has the size of 2x Screen width and Screen height + 1 (_imageBuf).
3. Set3D() function continues by appending the stereo signature on the last row.
4. OnPaint()-function takes the back buffer (_backBuf) and copies the content of the combined image (_imageBuf) to it.

When I run the program, shutter glasses start working, but I only see the two images side-by-side on the screen. Could someone help out and tell me what am I doing wrong? I believe that solving this problem might also help others as there does not yet seem to be a simple example how to do this with C#.

Below are the tactical parts of my code. Complete project can be downloaded here: http://koti.mbnet.fi/jjantti2/NVStereoTest.rar

```
    public void InitializeDevice()
    {
        PresentParameters presentParams = new PresentParameters();

        presentParams.Windowed = false;
        presentParams.BackBufferFormat = Format.A8R8G8B8;
        presentParams.BackBufferWidth = _size.Width;
        presentParams.BackBufferHeight = _size.Height;
        presentParams.BackBufferCount = 1;
        presentParams.SwapEffect = SwapEffect.Discard;
        presentParams.PresentationInterval = PresentInterval.One;
        _device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);
    }

    public void LoadSurfaces()
    {
        _imageBuf = _device.CreateOffscreenPlainSurface(_size.Width * 2, _size.Height + 1,
Format.A8R8G8B8, Pool.Default);

        _imageLeft = Surface.FromBitmap(_device, (Bitmap)Bitmap.FromFile("Blue.png"),
Pool.Default);
        _imageRight = Surface.FromBitmap(_device, (Bitmap)Bitmap.FromFile("Red.png"),
Pool.Default);
    }

    private void Set3D()
    {
        Rectangle destRect = new Rectangle(0, 0, _size.Width, _size.Height);
        _device.StretchRectangle(_imageLeft, _size, _imageBuf, destRect,
TextureFilter.None);
        destRect.X = _size.Width;
        _device.StretchRectangle(_imageRight, _size, _imageBuf, destRect,
TextureFilter.None);

        GraphicsStream gStream = _imageBuf.LockRectangle(LockFlags.None);

        byte[] data = new byte[] {0x44, 0x33, 0x56, 0x4e,   //NVSTEREO_IMAGE_SIGNATURE
= 0x4433564e
                                  0x00, 0x00, 0x0F, 0x00,   //Screen width * 2 = 1920*2 =
3840 = 0x00000F00;
                                  0x00, 0x00, 0x04, 0x38,   //Screen height = 1080
= 0x00000438;
                                  0x00, 0x00, 0x00, 0x20,   //dwBPP = 32
= 0x00000020;
                                  0x00, 0x00, 0x00, 0x02};  //dwFlags = SIH_SCALE_TO_FIT
= 0x00000002;

        gStream.Seek(_size.Width * 2 * _size.Height * 4, System.IO.SeekOrigin.Begin);
//last row
        gStream.Write(data, 0, data.Length);
        gStream.Close();

        _imageBuf.UnlockRectangle();
    }

    protected override void OnPaint(System.Windows.Forms.PaintEventArgs e)
    {
        _device.BeginScene();

        // Get the Backbuffer then Stretch the Surface on it.
        _backBuf = _device.GetBackBuffer(0, 0, BackBufferType.Mono);
        _device.StretchRectangle(_imageBuf, new Rectangle(0, 0, _size.Width * 2,
_size.Height + 1), _backBuf, new Rectangle(0, 0, _size.Width, _size.Height),
TextureFilter.None);
        _backBuf.ReleaseGraphics();

        _device.EndScene();

        _device.Present();

        this.Invalidate();
    }
```

c# directx nvidia stereo-3d

asked May 9 '11 at 10:11
Andows
33 1 3

1 Answer

A friend of mine found the problem. The bytes in the stereo signature were in reversed order. Here is the correct order:

```
byte[] data = new byte[] {0x4e, 0x56, 0x33, 0x44,   //NVSTEREO_IMAGE_SIGNATURE         =
0x4433564e;
0x00, 0x0F, 0x00, 0x00,   //Screen width * 2 = 1920*2 = 3840 = 0x00000F00;
0x38, 0x04, 0x00, 0x00,   //Screen height = 1080             = 0x00000438;
0x20, 0x00, 0x00, 0x00,   //dwBPP = 32                       = 0x00000020;
0x02, 0x00, 0x00, 0x00};  //dwFlags = SIH_SCALE_TO_FIT       = 0x00000002;
```

The code works perfectly after this change. This code might even serve as a good tutorial for someone else attempting the same thing. :)

answered May 9 '11 at 12:29
Andows
36 1

protected by Community ♦ Jun 27 '11 at 18:28

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?