

Re: Z-Buffer problems rendering a Mesh

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.managed/2004-03/00>

From: Patrice Scribe (*a_at_b.c*)

Date: 03/04/04

Date: Thu, 4 Mar 2004 09:30:18 +0100

Show rather the code as it is when it doesn't work. When you enable the Zbuffer do you add the corresponding flag in the Device.Clear call (else you'll clear only the render target, not the ZBuffer) ?

Patrice

--

```
"Taren" <ediego@miami.edu> a Ã©crit dans le message de
news:0D3D73E5-9A9A-463B-8AEA-75B96F5149F9@microsoft.com...
> I am creating a windows app to store and display .x files. I have the
application to the point where I am loading the .x files into a mesh and
then rendering them to the display. I am encountering a problem that I do
not know how to resolve. It looks like a z-buffer problem. When i set the
zBufferEnable render state to false I get the model looking like the
following 3 images. That's how it should look since there is no z buffer.
When i set the ZBufferEnable to true, nothing renders to the display. I
don't get any errors or exceptions, I just see the empty background. Anyone
have any ideas what I am doing wrong.
>
> http://homer.bus.miami.edu/~ediego/render1.jpg
> http://homer.bus.miami.edu/~ediego/render2.jpg
> http://homer.bus.miami.edu/~ediego/render3.jpg
>
> I am using the spaceship 13.x file included with the DX9 samples. Here is
some of the source code that I am using.
>
> TO CREATE THE DEVICE:
>
*****
*****
> public System.Boolean createDevice(System.Windows.Forms.Control
ourRenderTarget, System.Int32 iWidth, System.Int32 iHeight, Format dxFormat,
System.Boolean bWindowed)
> {
> System.Boolean bSuccess = false;
>
> /* Now we need to chec to see if our adapters will support the display
mode we want.
> /* We only need to check this if we are trying to create a fullscreen
application.
> if (!bWindowed)
> {
> bSuccess = this.isResolutionSupported(0, iWidth, iHeight, dxFormat);
> if (bSuccess == false)
```

```
> {
> /** We were unable to find a valid display mode.
> this.addMessage("Unable to find a compatible Display Mode: " + iWidth +
> "x" + iHeight + " " + dxFormat.ToString()),
rdMessageLog.rdLogLevel.rdLogLevelError);
> return false;
> }
> }
>
> /** We have found our display mode.
> /** Lets set our presentation parameters
> this.m_iWidth = iWidth;
> this.m_iHeight = iHeight;
> this.m_dxPixelFormat = dxFormat;
> this.m_bWindowed = bWindowed;
>
> this.m_dxPresentParams = new PresentParameters();
> this.m_ourRenderTarget = ourRenderTarget;
> this.m_dxPresentParams.Windowed = this.m_bWindowed;
> this.m_dxPresentParams.BackBufferCount = 1;
> this.m_dxPresentParams.SwapEffect = SwapEffect.Discard;
> this.m_dxPresentParams.PresentFlag = PresentFlag.None;
> this.m_dxPresentParams.EnableAutoDepthStencil = true;
> this.m_dxPresentParams.AutoDepthStencilFormat = Direct3D.DepthFormat.D16;
> this.m_dxPresentParams.PresentFlag = PresentFlag.None;
>
>
> if (m_bWindowed)
> {
> /** If we are displaying in a windowed environemnt then get the curent
display mode.
> DisplayMode dxCurrentMode = Manager.Adapters[0].CurrentDisplayMode;
>
> this.m_dxPresentParams.BackBufferWidth =
this.m_ourRenderTarget.ClientRectangle.Right -
ourRenderTarget.ClientRectangle.Left;
> this.m_dxPresentParams.BackBufferHeight =
this.m_ourRenderTarget.ClientRectangle.Bottom -
ourRenderTarget.ClientRectangle.Top;
> this.m_dxPresentParams.BackBufferFormat = dxCurrentMode.Format;
> this.m_dxPresentParams.FullScreenRefreshRateInHz = 0;
> this.m_dxPresentParams.PresentationInterval = PresentInterval.Immediate;
> this.m_dxPresentParams.DeviceWindow = ourRenderTarget;
> }
> else
> {
> m_dxPresentParams.BackBufferWidth = this.m_iWidth;
> m_dxPresentParams.BackBufferHeight = this.m_iHeight;
> m_dxPresentParams.BackBufferFormat = this.m_dxPixelFormat;
> }
>
> /** Lets create our device.
> this.m_dxDevice = new Microsoft.DirectX.Direct3D.Device(0,
DeviceType.Hardware, ourRenderTarget, CreateFlags.SoftwareVertexProcessing,
this.m_dxPresentParams);
>
> return true;
> }
>
>
>
> TO LOAD THE FILE INTO A MESH:
```



```
> // Create the texture
> this.m_textures[i] = this.loadTextureFromFile(dxDevice,
m_extMaterials[i].TextureFilename);
> System.Diagnostics.Debug.WriteLine("Texture = " +
m_extMaterials[i].TextureFilename);
> }
>
> }
> catch (System.NullReferenceException e)
> {
>
>
> this.addMessage("Null Refernece Exception occured: " + e.Message,
rdMessageLog.rdLogLevel.rdLogLevelError);
> return false;
>
> }
> catch (System.Exception e)
> {
> this.addMessage("Unknown exception occured: " + e.Message,
rdMessageLog.rdLogLevel.rdLogLevelError);
> return false;
> }
>
> return true;
> }
>
>
> TO RENDER THE MESH:
>
*****
*****
> /* Set some global render states
> this.m_dxGrafixEnv.Device.RenderState.ZBufferEnable = false;
> this.m_dxGrafixEnv.Device.RenderState.CullMode = Direct3D.Cull.None;
> this.m_dxGrafixEnv.Device.RenderState.Ambient =
System.Drawing.Color.White;
> this.m_dxGrafixEnv.Device.RenderState.FillMode = Direct3D.FillMode.Solid;
>
>
>
> // Now we're ready to recieve and process Windows messages.
> while (!this.m_dxGrafixEnv.isDeviceNull())
> {
> /* We need to do a few more things here.
> /* We need to add a few functions to render the scene.
> try
> {
> this.m_dxGrafixEnv.Device.BeginScene();
> this.m_dxGrafixEnv.Device.Clear(ClearFlags.Target, Color.White, 1.0f, 0);
>
> System.Single siAspectRatio =
System.Convert.ToSingle(this.pnlModelView.Height) /
System.Convert.ToSingle(this.pnlModelView.Width);
>
> /* Check the objectto render to see if we need to do anything with it.
>
> if (this.m_objectType == ArtifactManager.frmMain.rdRenderTpye.rdD3DMesh &&
> this.m_mesh != null)
> {
> /* Let's get the radius of the mesh so that we can adjust the
```

```
> /** view to encompass the whole object.
> System.Single siObjectRadius = this.m_mesh.Radius;
> System.Diagnostics.Debug.WriteLine("Radius of mesh: " + " = " +
siObjectRadius);
>
> /** Set our projection matrix
> this.m_dxGrafixEnv.setProjectionMatrix( System.Convert.ToSingle(Math.PI /
4),
> siAspectRatio,
> /*siObjectRadius / 64.0f,*/
> 1.0f,
> siObjectRadius * 200.0f);
> /*
> 0.0f,
> 100.0f);
> */
>
> /** Set our view
> this.m_dxGrafixEnv.setViewMatrix( new DirectX.Vector3(0, 0, 0 -
(siObjectRadius + 8) ),
> new DirectX.Vector3(0, 0, 0),
> new DirectX.Vector3(0, 1, 0));
>
> /** Render the 3D Mesh
> m_mesh.renderMesh(this.m_dxGrafixEnv.Device);
>
> }
>
>
> this.m_dxGrafixEnv.Device.EndScene();
> this.m_dxGrafixEnv.Device.Present();
>
> /** Spin the scene around the y-axis.
> this.m_dxGrafixEnv.Device.Transform.World =
Microsoft.DirectX.Matrix.RotationY(System.Environment.TickCount / 800.0f );
>
> }
> catch (System.NullReferenceException e)
> {
> /** A null reference exceptin occured while rendering the scene.
> this.m_dxGrafixEnv.addMessage(e.Message,
rdMessageLog.rdLogLevel.rdLogLevelError);
> return;
> }
>
> System.Windows.Forms.Application.DoEvents();
> }
```