Beyond Printf

Debugging Graphics Through Tools
Presenters

サー Dave Aronson
NVIDIA – Technical Evangelist
daronson@nvidia.com

サー Karen Stevens
Microsoft – Software Design Engineer / Test
XNA Professional Game Platform
kstevens@microsoft.com
Purpose

- To determine criteria for graphics tool selection
- To demonstrate how tools can be used to identify and solve top game scenarios
Agenda

- Tool Selection
- Scenarios
- Live Demos
- Q&A
- References
Preliminary Criteria Points

When selecting a tool, consider:

- Budget
- General machine requirements
- Hardware manufacturers
- Additional required software
- Code modification requirements
- Product support
- Features and general areas of interest
Popular Tool Areas of Interest

- Game Assets
  Textures, Shaders, Vertex Buffers, etc

- API Usage
  DirectX / OpenGL calls, state, debug spew

- Driver
  Driver versions, driver timing

- Hardware
  Timing, hardware usage
Tools Shown Today

- AMD
  - GPU PerfStudio
- Microsoft
  - PIX for Windows
- NVIDIA
  - PerfHUD
  - FX Composer
Tool Categorization

- **Game Asset**
  - PIX for Windows, GPU PerfStudio, FX Composer, PerfHUD

- **API**
  - PIX for Windows, PerfHUD, GPU PerfStudio

- **Driver**
  - PerfHUD, GPU PerfStudio

- **Hardware**
  - PerfHUD, GPU PerfStudio
Example

Criteria:
- Application uses DirectX 9 / HLSL
- NVIDIA GeForce 7800 card is present
- Do not want to change code to use tool
- Preference towards free tools

Possible options from previous list:
- FX Composer
- PIX for Windows
How to Choose

- Determine analysis levels of interest
  - One strategy is to start at the game asset level and work down the list

- Determine how tool fits criteria
  - Prioritize your requirements

- Experiment
  - Most tools are free or have free trial periods, try a variety of scenarios
Scenarios

- Glitches
  - Incorrect behavior

- Bottlenecks
  - Poor performance
Glitches

The game is not behaving as expected:

- Game Crash
- Blank Screen
- Missing Objects
- Flickering
Game Crash

GDCChess.exe has stopped working

Windows can check online for a solution to the problem.

- Check online for a solution and close the program
- Close the program
- Debug the program

View problem details
Game Crash

Scenario:
- Game crashes when moving from windowed to full screen
- Only occurs on specific video cards
- The game does not have a debug build due to performance/game play reasons
Game Crash

Select settings to handle crash analysis

Note: Disabling write caching makes capture slower, but is likely to record more information if the target program crashes.
Game Crash

- Setup diagnostic logging
Game Crash

A diagnostic log (3883 KB) was created while PIX was analyzing 'GDCChess.exe'.

Look for debug output messages regarding incorrect Direct3D usage, or invalid parameters in calls to Direct3D.

Diagnostic log file excerpt (click Save As to save the full log):

```
Frame 000003 ...............PRE: RemoveObject(D3D9 State Block, 0x03CEEB00, 0x0A934D60)
Frame 000003 ...............POST: <> RemoveObject(D3D9 State Block, 0x03CEEB00, 0x0A934D60)
Frame 000003 ...............POST: <0><this=0x03ceeb00> IDirect3DStateBlock9::Release()
Frame 000003 ...............PRE: <this=0x03cefb98> IDirect3DStateBlock9::Release()
Frame 000003 ...............POST: <> RemoveObject(D3D9 State Block, 0x03CEFB98, 0x0A934EC0)
Frame 000003 ...............POST: <> RemoveObject(D3D9 State Block, 0x03CEFB98, 0x0A934EC0)
Frame 000003 ...............POST: <0><this=0x03cebf98> IDirect3DStateBlock9::Release()
Frame 000003 ...............PRE: <this=0x03ce610> IDirect3DStateBlock9::Release()
Frame 000003 ...............PRE: RemoveObject(D3D9 State Block, 0x03CE610, 0xA934E40)
Frame 000003 ...............POST: <> RemoveObject(D3D9 State Block, 0x03CE610, 0xA934E40)
Frame 000003 ...............POST: <> RemoveObject(D3D9 State Block, 0x03CE610, 0xA934E40)
Frame 000003 ...............POST: <0><this=0x03ce610> IDirect3DStateBlock9::Release()
Frame 000003 ...............POST: <0><this=0x02389fe8> IDirect3DDevice9::Release()
Frame 000003 ...............PRE: <this=0x03c53ed8> IDirect3DDevice9::Reset(0x04000F84)
Direct3D9: (ERROR): All user created D3DPOOL_DEFAULT surfaces must be freed before ResetEx can succeed. Re
An unhandled exception occurred.
Closing Run File
```

Do you want to discard or save the log file?
Game Crash

Analysis:

⚠️ Error: Direct3D9: (ERROR) :All user created D3DPOOL_DEFAULT surfaces must be freed before ResetEx can succeed. ResetEx Fails. An unhandled exception occurred.
Game Crash

Open run file for analysis
Game Crash

- Examine objects left after last valid call

### Objects

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Destruction</th>
<th>Status</th>
<th>App Refs</th>
<th>Pool</th>
<th>Usage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02A7A390</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A438</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A4B9B8</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>DepthStencil</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A4B910</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>RenderTarget</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A4E0</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A588</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A630</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A6D8</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A780</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A828</td>
<td>D3D9 Surface</td>
<td>Never</td>
<td>Alive</td>
<td>0</td>
<td>Default</td>
<td>Dynamic</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A7A82C</td>
<td>D3D9 Surface</td>
<td>1</td>
<td>Dead</td>
<td>n/a</td>
<td>Default</td>
<td>RenderTarget</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A4B6E0</td>
<td>D3D9 Surface</td>
<td>1</td>
<td>Dead</td>
<td>n/a</td>
<td>Default</td>
<td>RenderTarget</td>
<td>D3DFMT</td>
</tr>
<tr>
<td>0x02A4B6F0</td>
<td>D3D9 Surface</td>
<td>1</td>
<td>Dead</td>
<td>n/a</td>
<td>Default</td>
<td>RenderTarget</td>
<td>D3DFMT</td>
</tr>
</tbody>
</table>
Game Crash

Located rogue object creation point
Game Crash

- Trace calls for objects requiring release
Game Crash

Conclusion:

- Some D3DPOOL_DEFAULT textures were not released before ResetEx occurred.
- Tools can examine remaining objects/textures to help ID items that require rework.
- Remaining objects are easily cleaned up once identified.
- Allows debugging of both retail and debug builds (assuming no copy write protection).
Blank Screen
Blank Screen

Scenario:

- Many machines render a black screen
- The program works fine on some machines
- Video card is the same on all machines
- Video driver is the same on all machines
Blank Screen

- Overriding states can rule out issues early
Blank Screen

- Overriding texture renders scene viewable
Blank Screen

- Checking for sampler issues

- Samplers exist, values look ok
Blank Screen

⚠️ Check texture sampler 0 - OK
Blank Screen

Sampler texture 1 should not be black
Blank Screen

_render frame and select inaccurate pixel_
Blank Screen

Pixel history shows all calls output black

Event 1254: IDirect3DDevice9::DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 4, 13608, 6, 23324)

Primitive 3 of 23324

Vertex Shader: 0x016EC8F0
  Debug Vertex 0
  Debug Vertex 1
  Debug Vertex 2

Pixel Shader: 0x016EC880
  Debug Pixel (369, 368)

Pixel shader output:

Alpha: 1.000
Red: 0.000
Green: 0.000
Blue: 0.000

Final framebuffer color:

Alpha: 0.000
Red: 0.000
Green: 0.000
Blue: 0.000

Event 1254: IDirect3DDevice9::DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 4, 13608, 6, 23324)
Blank Screen

Shader debugging proves black texture obliterates computed color

//

float4 CausticPS(VS_OUT IN) : COLOR
{
    float2 movement = IN.TexCoord1.xy;
    movement.x = movement.x + cos(Time * 0.2f) * 0.3f;
    movement.y = movement.y + sin(Time * 0.3f) * 0.2f;

    float3 color = IN.Color.rgb * tex2D(CausticTextureSampler, movement.xy * 0.9f);
    color = color * tex2D(MeshTextureSampler, IN.TexCoord0.xy);

    return color;
}

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>(0.000, 0.000, 0.000)</td>
<td>float3</td>
</tr>
<tr>
<td>movement</td>
<td>(0.845, 4.548)</td>
<td>float2</td>
</tr>
</tbody>
</table>
Blank Screen

Analysis:

- Incorrect texture is used
- The texture is involved in all lighting operations, therefore everything is black
- Black is a common fallback for textures which were unable to be loaded at runtime
Blank Screen

Conclusion:

- The texture failed to load
- Texture loading is based on a file path
- Machines with an incorrect path didn’t load the texture
- Correcting path in setup restored lighting to all machines
Missing Objects
Missing Objects

Scenario:

- Code traces prove all draw calls are executed
- A few of the objects drawn are not displaying on the screen
Missing Objects

_rendered scene has missing objects_
Missing Objects

- Check wireframe geometry of scene
Missing Objects

 radians

 Suspicious artifacts present
Missing Objects

* Incorrect vertex shader input
Missing Objects

 يوناخ 3

 Yields unexpected output
Missing Objects

⚠ Incorrect input & fogged out

Pre-Vertex Shader | Post-Vertex Shader | Viewport
--- | --- | ---

<table>
<thead>
<tr>
<th>Prim</th>
<th>VTX</th>
<th>IDX</th>
<th>Position</th>
<th>Diffuse</th>
<th>Fog</th>
<th>TexCoord0</th>
<th>TexCoord1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td>5158</td>
<td>-13.523</td>
<td>10.854</td>
<td>10.953</td>
<td>2.155</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5159</td>
<td>-13.357</td>
<td>-34.861</td>
<td>9.552</td>
<td>9.651</td>
<td>2.090</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5159</td>
<td>-13.357</td>
<td>-34.861</td>
<td>9.552</td>
<td>9.651</td>
<td>2.090</td>
</tr>
</tbody>
</table>

PIX for Windows
WWW.GDCONF.COM
Missing Objects

- Defect demonstration, modifying application: no fog, no cull, zooming out
Missing Objects

Conclusion:

- Incorrect values were sent to vertex shaders in both cases
- Culling reduced odds of detecting the scene was inside the rook, fogging hid few remaining visible faces
Flickering
Flickering

Scenario:

- Texture shifts between two images every time mouse is moved or scene position changes
- There is only one known mesh object used for the chess board
Flickering

- Examine wireframe for obvious z-fighting
Flickering

- Examine mesh view for hidden artifacts

<table>
<thead>
<tr>
<th>Prim</th>
<th>VTX</th>
<th>IDX</th>
<th>Position</th>
<th>Diffuse</th>
<th>Fog</th>
<th>TexCoord0</th>
<th>TexCoord1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td>4</td>
<td>-3.647</td>
<td>0.206 5.252 5.335</td>
<td>D3DCOLOR_RGBA(0x00,0x33,0x33,0x33)</td>
<td>0.387</td>
<td>0.756</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>-3.647</td>
<td>0.745 5.810 6.910</td>
<td>D3DCOLOR_RGBA(0x00,0x33,0x33,0x33)</td>
<td>0.292</td>
<td>0.701</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>-3.647</td>
<td>1.285 6.369 6.486</td>
<td>D3DCOLOR_RGBA(0x00,0x33,0x33,0x33)</td>
<td>0.195</td>
<td>0.645</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7</td>
<td>-3.647</td>
<td>0.206 5.252 5.335</td>
<td>D3DCOLOR_RGBA(0x0a,0x6,0x6,0x6)</td>
<td>0.387</td>
<td>0.756</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>8</td>
<td>-3.647</td>
<td>1.285 6.369 6.486</td>
<td>D3DCOLOR_RGBA(0x0a,0x6,0x6,0x6)</td>
<td>0.195</td>
<td>0.645</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>9</td>
<td>-3.647</td>
<td>-0.872 4.135 4.235</td>
<td>D3DCOLOR_RGBA(0x0a,0x6,0x6,0x6)</td>
<td>0.566</td>
<td>0.866</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>10</td>
<td>-3.647</td>
<td>-0.872 4.135 4.235</td>
<td>D3DCOLOR_RGBA(0x9,0xe,0xe,0xe)</td>
<td>0.566</td>
<td>0.866</td>
</tr>
<tr>
<td>P1</td>
<td>7</td>
<td>11</td>
<td>-3.647</td>
<td>0.205 0.205 0.492</td>
<td>D3DCOLOR_RGBA(0x09,0xe,0xe,0xe)</td>
<td>0.495</td>
<td>0.865</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>12</td>
<td>-3.647</td>
<td>1.285 6.369 6.486</td>
<td>D3DCOLOR_RGBA(0x09,0xe,0xe,0xe)</td>
<td>0.495</td>
<td>0.865</td>
</tr>
</tbody>
</table>

Pre-Vertex Shader

Post-Vertex Shader

Viewport
Flickering

Hidden mesh subset uncovered
Flickering

Conclusion:
- The checkerboard mesh had 2 subsets
- 1 subset was coplanar with the board top
- Removal of subset fixed unanticipated z-fighting
Bottleneck Analysis

Overall behavior is correct, but rendering takes longer than expected:

- Culling & Render Order
- Buffer Sizes
- Ineffective Code
- Inefficient Shaders
- Batch Sizes
Culling & Render Order

Look at the overdraw in the tool
Culling & Render Order

Scroll through the draw calls to see how the frame is composed
Culling & Render Order

- Notice how the draws are just stacking and nothing is culled
- Are objects being rendered multiple times?
Culling & Render Order

- Check the render states
- Render state changes can happen in multiple places
Culling & Render Order

💡 You want to draw where the culling behavior will have the most effect.
Culling & Render Order

- Remember that transparent objects must be drawn after opaque objects. They also need to be drawn via the painters algorithm.
- Render back to front
Culling & Render Order

Guidelines:

Order of culling methods used:

- Software (portal/scene)
- View Frustum
- Z-test
- Bounding box – hw queries
  (did any pixels render or potentially render?)
Buffer Sizes

- Performance is slow
- But everything looks correct
- Thrashing of system resources
Buffer Sizes

There could be lots of swapping occurring
Buffer Sizes

Look at the perfmon counter for memory page faults
is it too high?
Buffer Sizes

- Is the swapping due to textures or other buffers?
- Look at the signals in PerfHUD
Buffer Sizes

- Sort the object table textures in PIX by size
Buffer Sizes

- Use mip-mapped textures
- Use smaller textures
- Use a compact texture format
- Don’t become infatuated with new features
  E.g. Selectively use aniso on textures
Buffer Sizes

- Only use data where necessary
- Pack data buffers with a smaller `vdecl`
- Use LOD techniques to reduce the amount of data needed
- Use a paging algorithm for loading data
- Reuse Render targets when possible
Inefficient Code

Are you sure you are GPU bound?
Look at the timing in PIX, PerfHUD
Inefficient Code

Total time ➔
Input Assembly ➔
Geometry ➔
Shader ➔
Texture ➔
Raster Ops ➔
Frame Buffer ➔
Inefficient Code

- Adjust
  - render size, texture sizes, cull objects
Inefficient Code

- Still slow? CPU bound
- Redundant state setting, set texture calls

<table>
<thead>
<tr>
<th>EID</th>
<th>Event</th>
<th>StartTime</th>
<th>Frame</th>
<th>Duration</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>917</td>
<td>Direct3DDevice9::BeginScene()</td>
<td>19801423920</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>918</td>
<td>D3DXEffect::SetFloat(0x005F6104, 13.9495)</td>
<td>19801436160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>919</td>
<td>D3DXEffect::SetMatrix(0x005F60A0, 0x0012FA10)</td>
<td>19801448448</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>920</td>
<td>D3DXEffect::SetMatrix(0x005F602C, 0x0012F9D0)</td>
<td>19801452544</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>921</td>
<td>D3DXEffect::SetMatrix(0x005F602C, 0x0012F9D0)</td>
<td>19801452544</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>922</td>
<td>D3DXEffect::SetTexture(0x005F5F38)</td>
<td>19801464832</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>923</td>
<td>D3DXEffect::SetVector(0x005F5F24, 0x0012F93C)</td>
<td>19801470976</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>924</td>
<td>D3DXEffect::SetVector(0x005F5F24, 0x0012F93C)</td>
<td>19801470976</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>925</td>
<td>Direct3DDevice9::SetRenderState(D3DRS_FOGENABLE, TRUE)</td>
<td>19801481215</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>926</td>
<td>D3DXEffect::SetVector(0x005F5F00, 0x0012F904)</td>
<td>19801489408</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>927</td>
<td>Direct3DDevice9::SetRenderState(D3DRS_CULLMODE, D3D_CULL_CCW)</td>
<td>19801493504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928</td>
<td>D3DXEffect::SetTexture(0x005F5FE0, 0x090CA7A0)</td>
<td>19801499648</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>User Event: Drawing Chessboard</td>
<td>19801667584</td>
<td>90865664</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inefficient Shaders

Use a tool to analyze your shader

![Shader Performance tool](image)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Show</th>
<th>Drives</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>UntexturedVS</td>
<td>Fragment</td>
<td>163.20</td>
<td>GeForceFX 5800 Ultra (NV30), GeForceFX 5900 Ultra (NV31), GeForceFX 5900 Ultra (NV32)</td>
</tr>
<tr>
<td>TextureVS</td>
<td>Fragment</td>
<td>163.20</td>
<td>GeForceFX 5800 Ultra (NV30), GeForceFX 5900 Ultra (NV31), GeForceFX 5900 Ultra (NV32)</td>
</tr>
<tr>
<td>UntexturedPS</td>
<td>Fragment</td>
<td>163.20</td>
<td>GeForceFX 5800 Ultra (NV30), GeForceFX 5900 Ultra (NV31), GeForceFX 5900 Ultra (NV32)</td>
</tr>
<tr>
<td>TexturedPS</td>
<td>Fragment</td>
<td>163.20</td>
<td>GeForceFX 5800 Ultra (NV30), GeForceFX 5900 Ultra (NV31), GeForceFX 5900 Ultra (NV32)</td>
</tr>
</tbody>
</table>

**NV30 (GeForceFX 5800 Ultra)**
- Regs: n/a
- Cycles: n/a
- MP/s: n/a

**NV31 (GeForceFX 5900 Ultra)**
- Regs: n/a
- Cycles: n/a
- MP/s: n/a

**NV32 (GeForceFX 5900 Ultra)**
- Regs: n/a
- Cycles: n/a
- MP/s: n/a

**NV40 (GeForce 6800 Ultra)**
- Regs: 1, 1, 1, 1, 1
- Cycles: 2105
- MP/s: 2105

**NV40-GT (GeForce 6800 Ultra GT)**
- Regs: 1, 1, 1, 1, 1
- Cycles: 2105
- MP/s: 2105

**NV41 (GeForce 5800)**
- Regs: 1, 1, 1, 1, 1
- Cycles: 4000
- MP/s: 4000

**NV44 (GeForce 5800 GT)**
- Regs: 1, 1, 1, 1, 1
- Cycles: 1730
- MP/s: 1730

**GT60-GT (GeForce 7800 GTX)**
- Regs: 1, 1, 1, 1, 1
- Cycles: 1730
- MP/s: 1730

**Website:** [www.gdconf.com](http://www.gdconf.com)
Inefficient Shaders

- Are you sure it is the shader?
- Swap the shader for a simpler shader, did that make a difference?
- Suboptimal code in inner loop
Batch Sizes

- Small batch sizes are inefficient and hard to detect
- Just because the batches are big doesn’t mean that it is good either
Summary

- Tools can be a valuable aid to quickly determine root causes of a variety of graphics problems.
- Tools can cover a variety of debugging levels, from high-level API issues to low-level hardware issues.
Live Demos

- Microsoft - PIX for Windows
- NVIDIA - PerfHUD
Q&A

Questions, Comments, Concerns?
Resources

Tools shown today can be downloaded at:

- AMD
  - http://developer.amd.com
- Microsoft
  - http://msdn.microsoft.com/directx
- NVIDIA
  - http://developer.NVIDIA.com/

The “PIXGameDebugging” application used in this presentation is available as a d3d9 tutorial in the DirectX Software Development Kit, March 2008 release.
Resources

Recommended Newsgroups, sites, & Forums

- http://developer.NVIDIA.com/forums/
- http://forums.xna.com/
- http://www.gamedev.net/
- http://developer.intel.com
- http://www.opengl.org
- http://www.gremedy.com/
- http://www.acm.org