Beyond Printf

Debugging Graphics Through Tools

Presenters

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

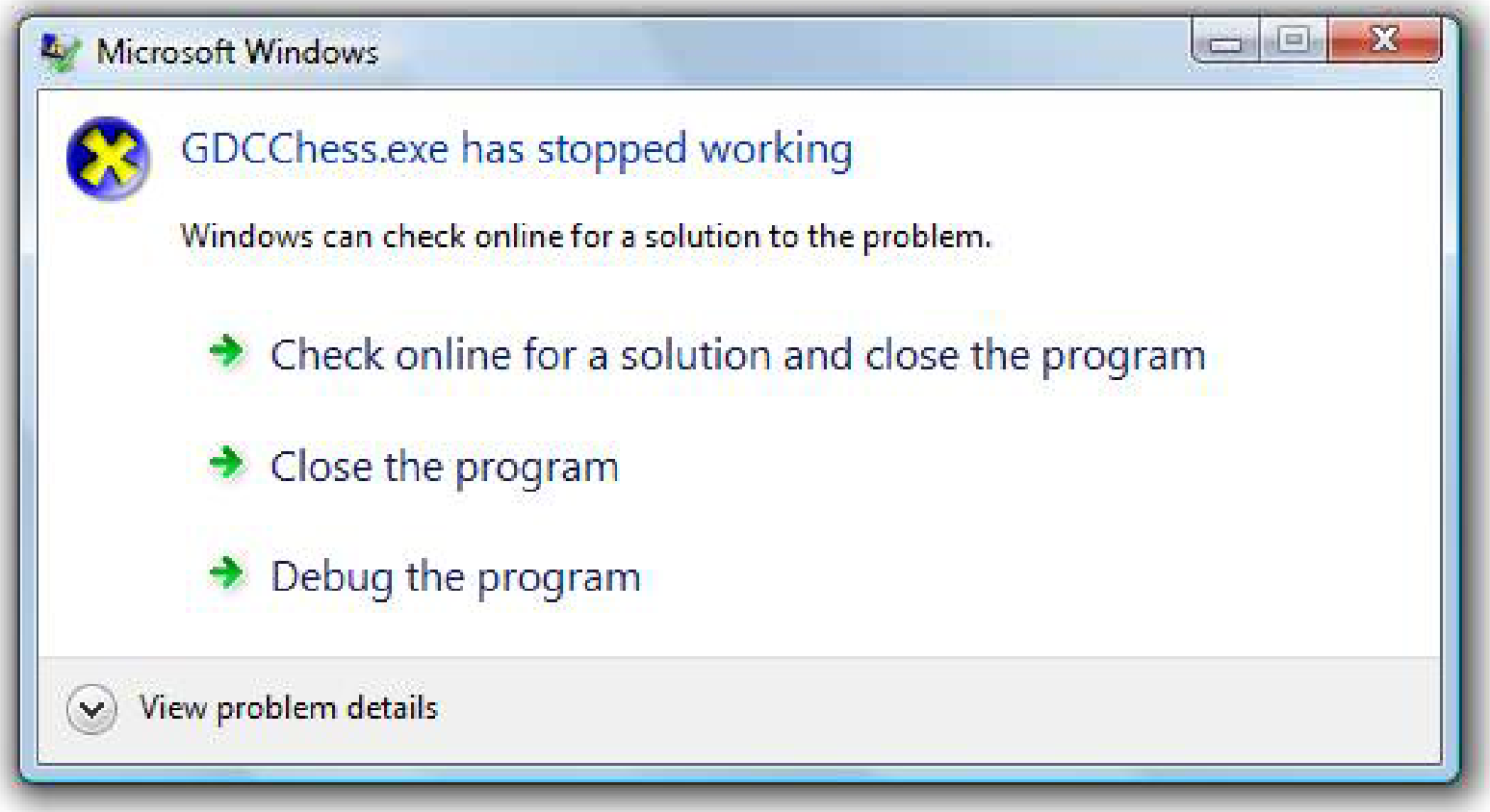
- Glitches
  - Incorrect behavior
- Bottlenecks
  - Poor performance

Glitches

The game is not behaving as expected:

- Game Crash
- Blank Screen
- Missing Objects
- Flickering

Game Crash


Microsoft Windows
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

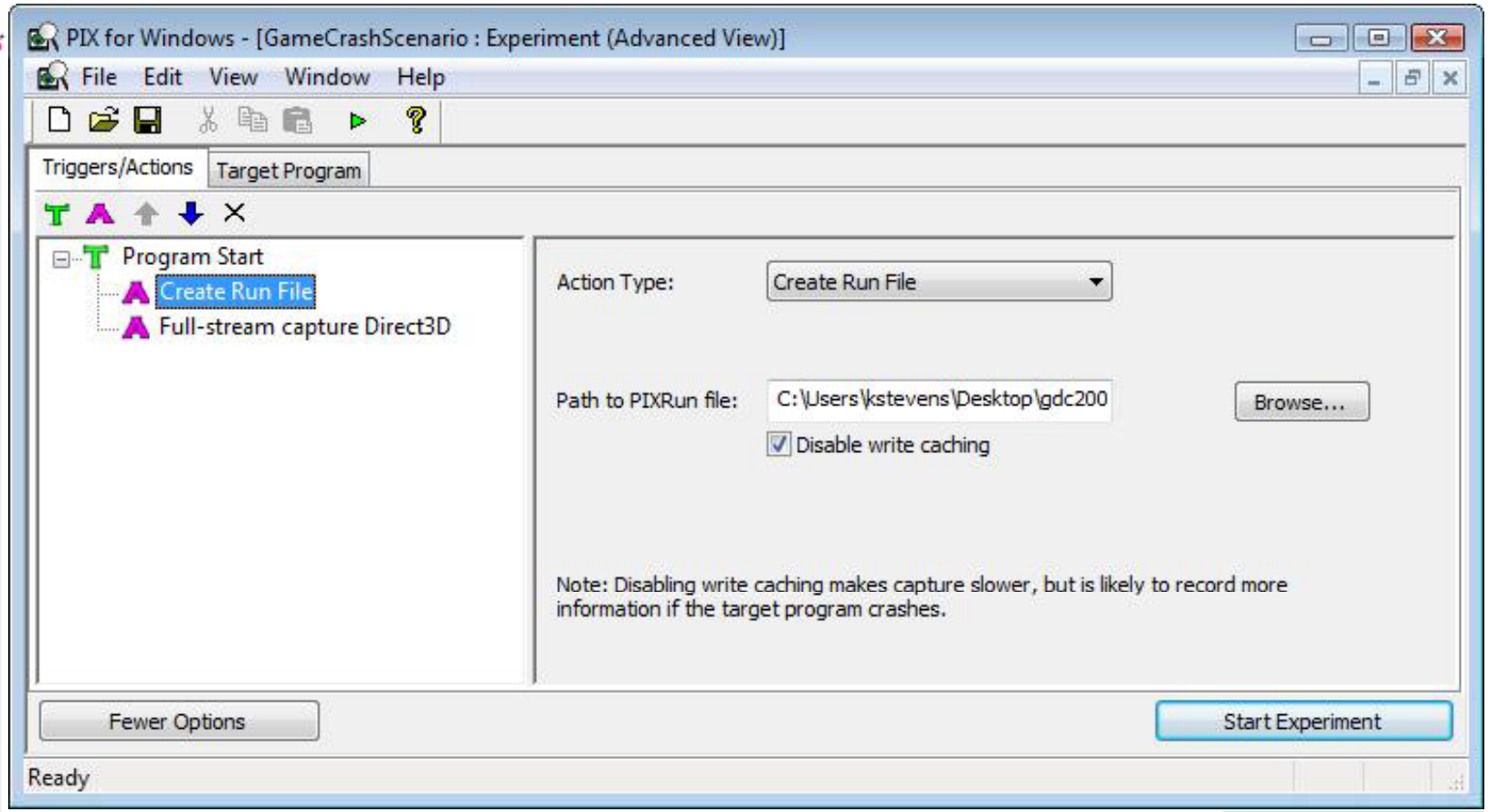
Game Crash

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

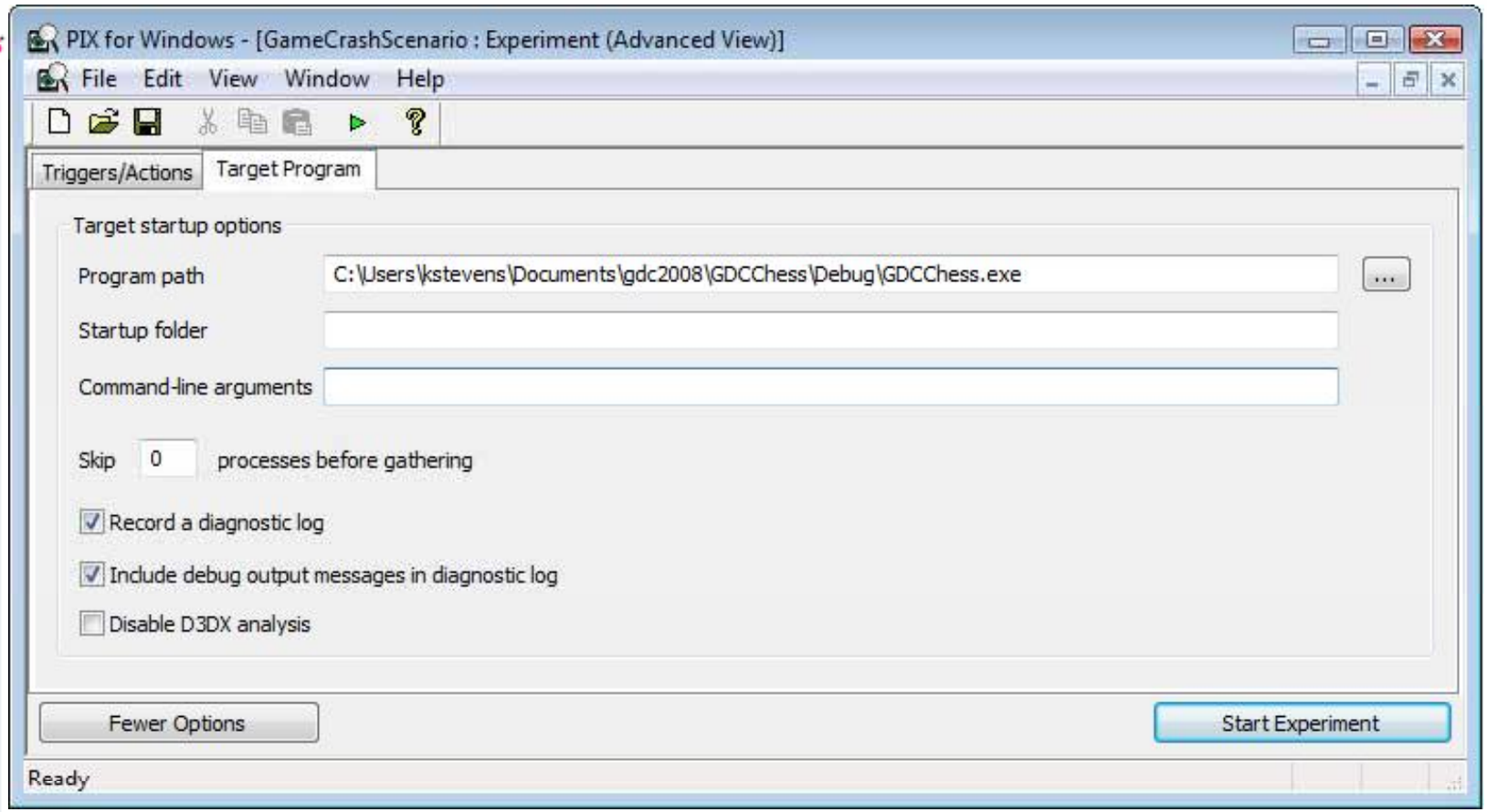
Game Crash

- Select settings to handle crash analysis

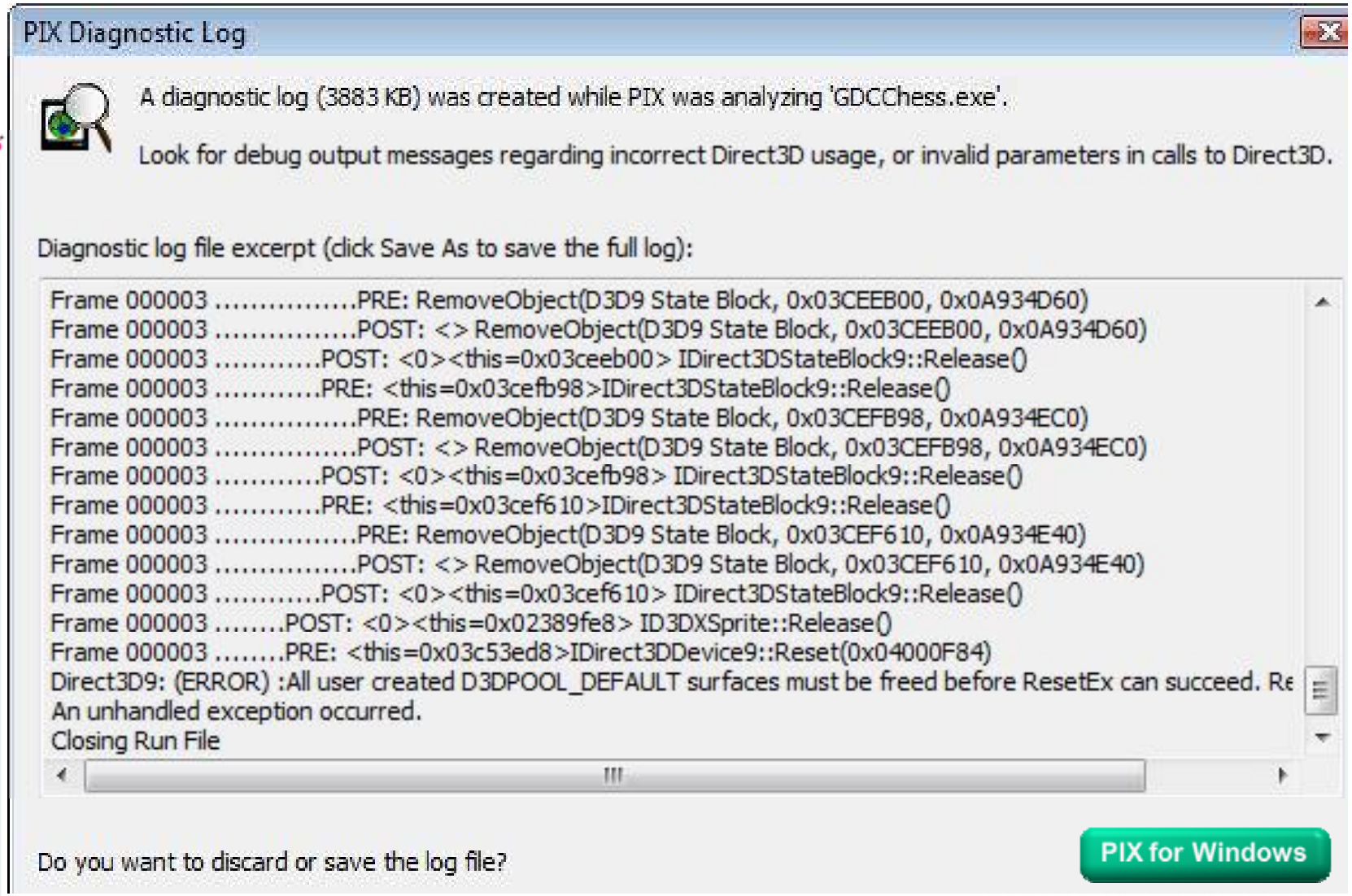


Game Crash

- Setup diagnostic logging



Game Crash


PIX Diagnostic Log
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 .....PRE: RemoveObject(D3D9 State Block, 0x03CEFB98, 0x0A934EC0)
Frame 000003 .....POST: <> RemoveObject(D3D9 State Block, 0x03CEFB98, 0x0A934EC0)
Frame 000003 .....POST: <0><this=0x03cefb98> IDirect3DStateBlock9::Release()
Frame 000003 .....PRE: <this=0x03cef610>IDirect3DStateBlock9::Release()
Frame 000003 .....PRE: RemoveObject(D3D9 State Block, 0x03CEF610, 0x0A934E40)
Frame 000003 .....POST: <> RemoveObject(D3D9 State Block, 0x03CEF610, 0x0A934E40)
Frame 000003 .....POST: <0><this=0x03cef610> IDirect3DStateBlock9::Release()
Frame 000003 .....POST: <0><this=0x02389fe8> ID3DXSprite::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?
PIX for Windows

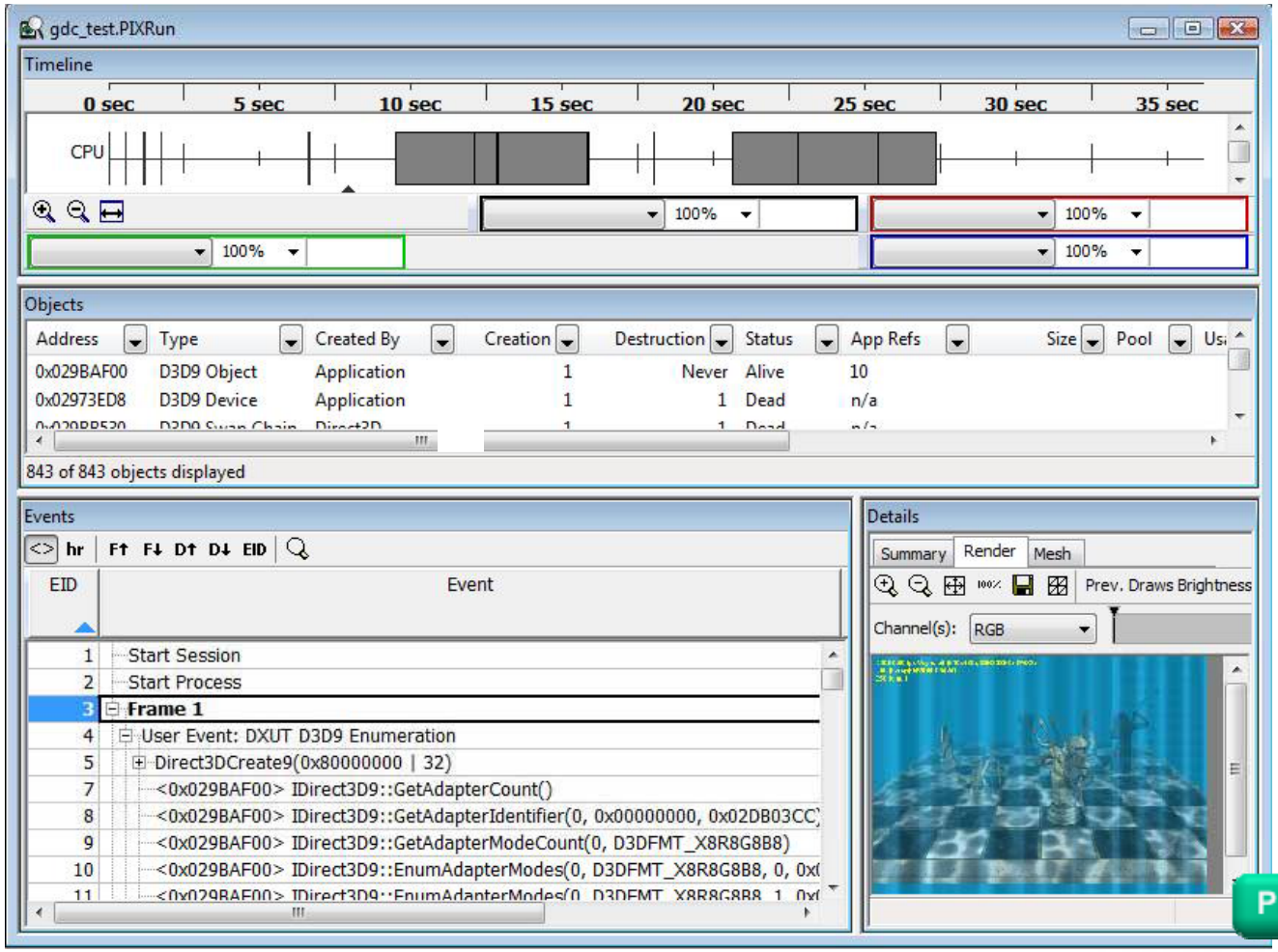
Game Crash

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



PIX for Windows

Game Crash

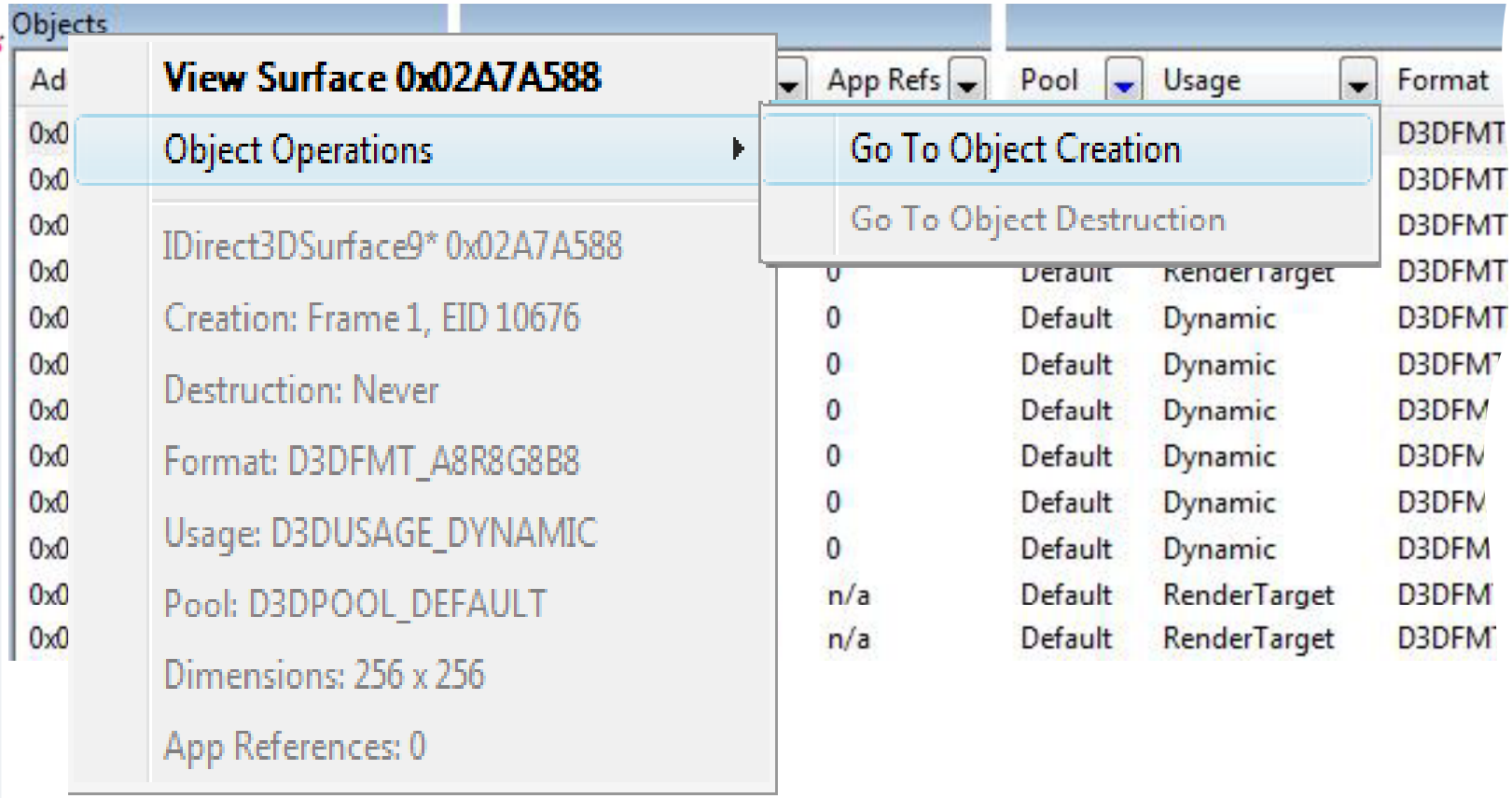
- Examine objects left after last valid call

Objects

Address	Type	Destruction	Status	App Refs	Pool	Usage	Format
0x02A7A390	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFMT
0x02A7A438	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFMT
0x02A4B9B8	D3D9 Surface	Never	Alive	0	Default	DepthStencil	D3DFMT
0x02A4B910	D3D9 Surface	Never	Alive	0	Default	RenderTarget	D3DFMT
0x02A7A4E0	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFMT
0x02A7A588	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFMT
0x02A7A630	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFM
0x02A7A6D8	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFM
0x02A7A780	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFM
0x02A7A828	D3D9 Surface	Never	Alive	0	Default	Dynamic	D3DFM
0x02A4B6B0	D3D9 Surface	1	Dead	n/a	Default	RenderTarget	D3DFM
0x02A4B7E0	D3D9 Surface	1	Dead	n/a	Default	RenderTarget	D3DFM

Game Crash

- Located rouge object creation point



Game Crash

- Trace calls for objects requiring release

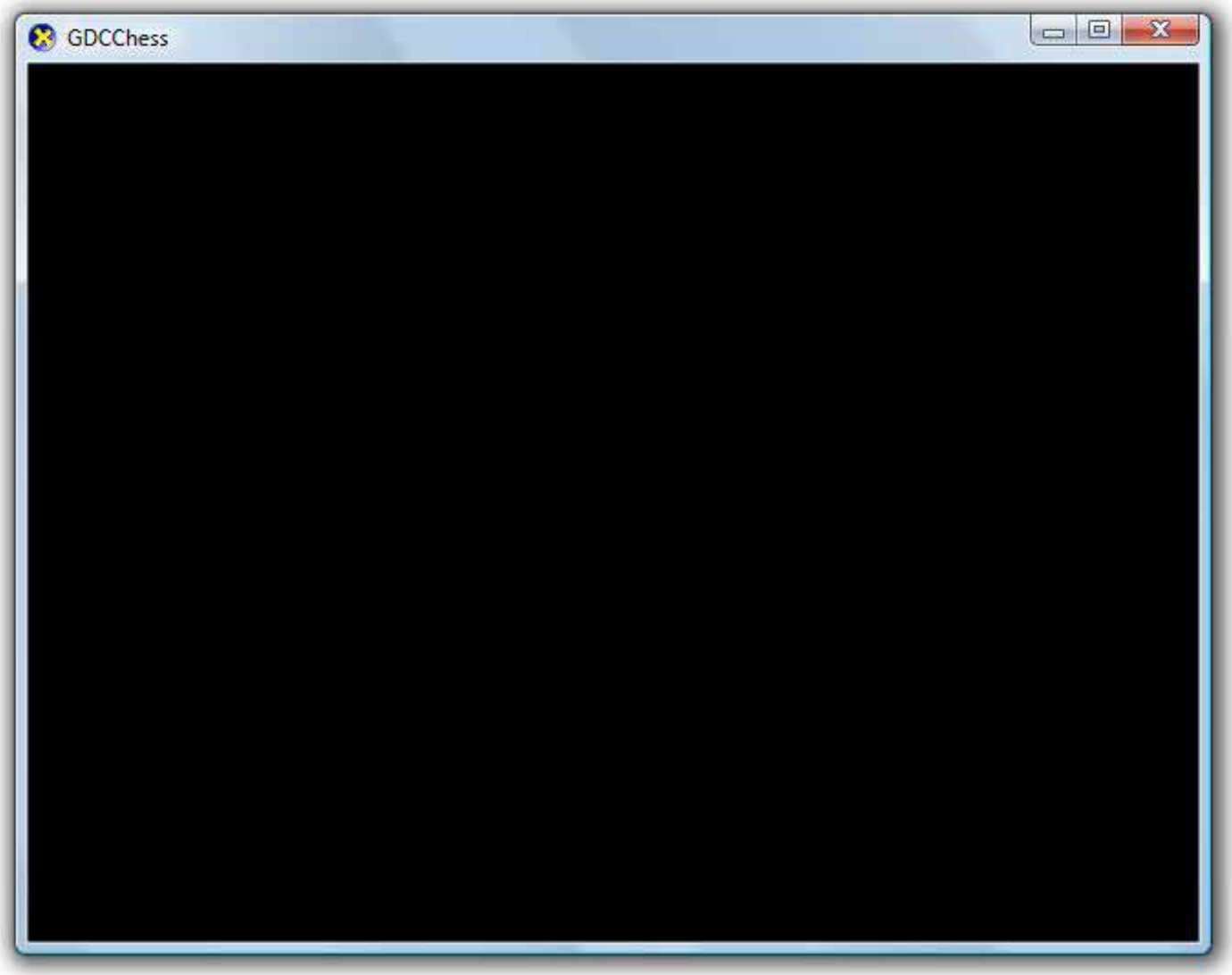
6926	⊞ User Event: Water Texture setup
10671	⊟ <0x02A00B10> IDirect3DDevice9::CreateTexture(2048, 2048, 8,
10672	CreateObject(D3D9 Texture, 0x02A62E38)
10673	CreateObject(D3D9 Surface, 0x02A7A390)
10674	CreateObject(D3D9 Surface, 0x02A7A438)
10675	CreateObject(D3D9 Surface, 0x02A7A4E0)
10676	CreateObject(D3D9 Surface, 0x02A7A588)
10677	CreateObject(D3D9 Surface, 0x02A7A630)
10678	CreateObject(D3D9 Surface, 0x02A7A6D8)
10679	CreateObject(D3D9 Surface, 0x02A7A780)
10680	CreateObject(D3D9 Surface, 0x02A7A828)
10681	<0x02A62E38> IDirect3DTexture9::LockRect(0, 0x0017F6DC, NU

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



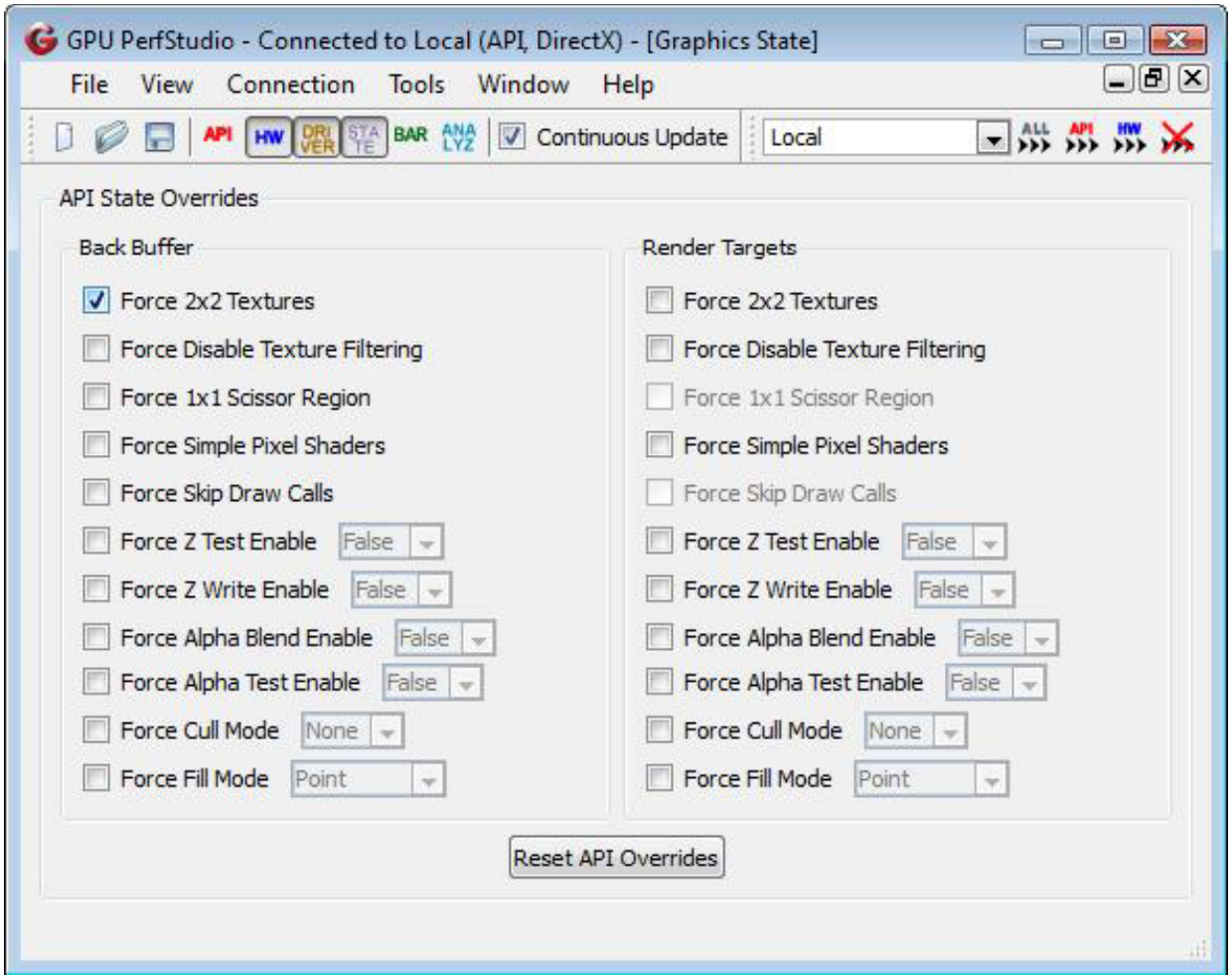
Blank Screen

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

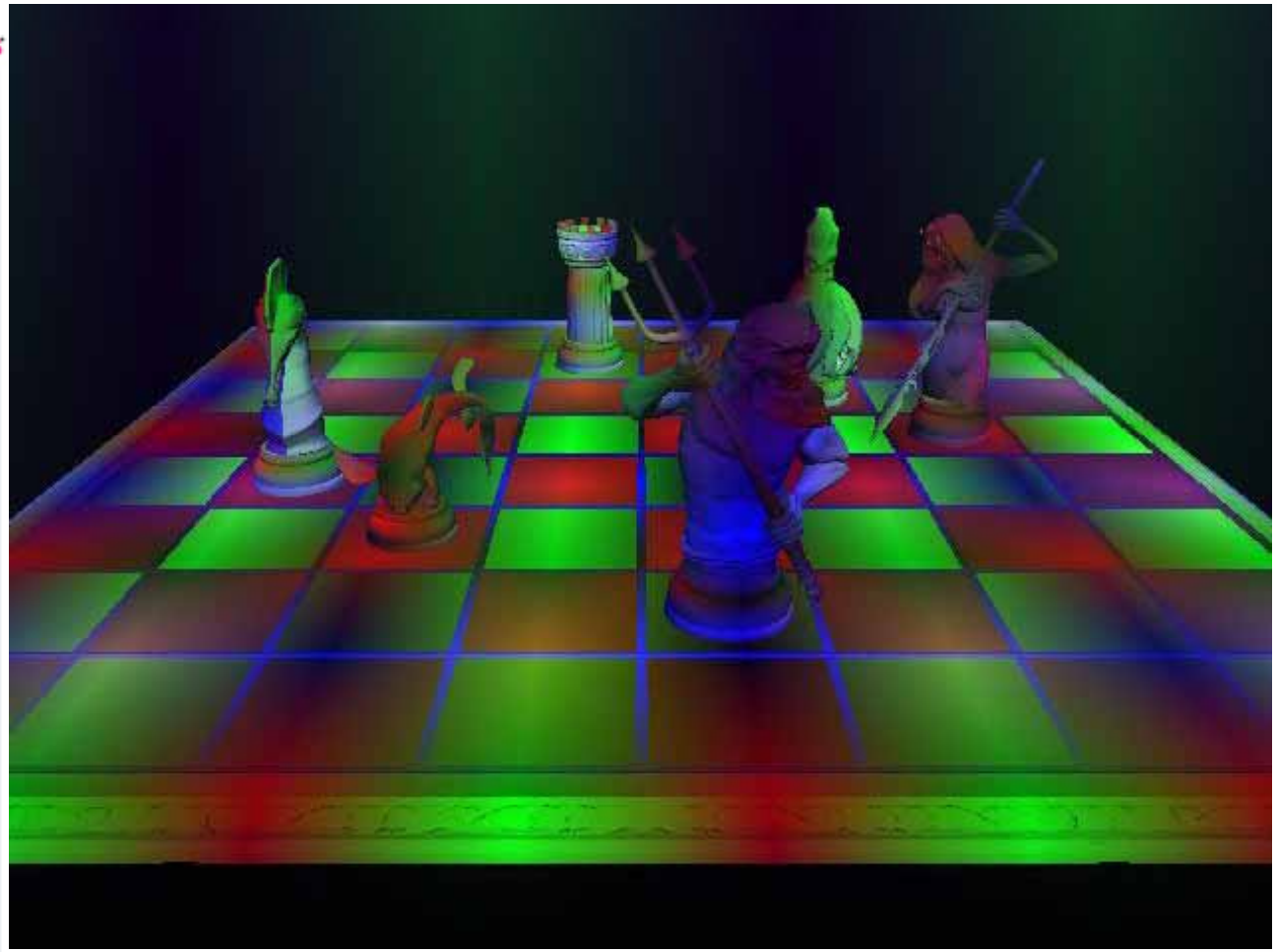
Blank Screen

- Overriding states can rule out issues early



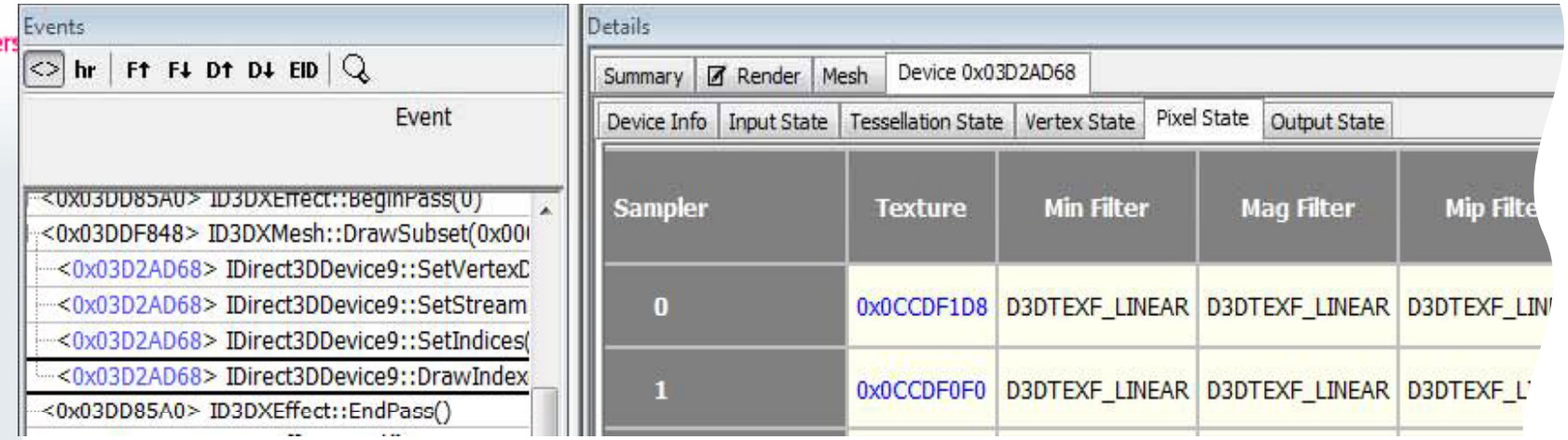
Blank Screen

- Overriding texture renders scene viewable



Blank Screen

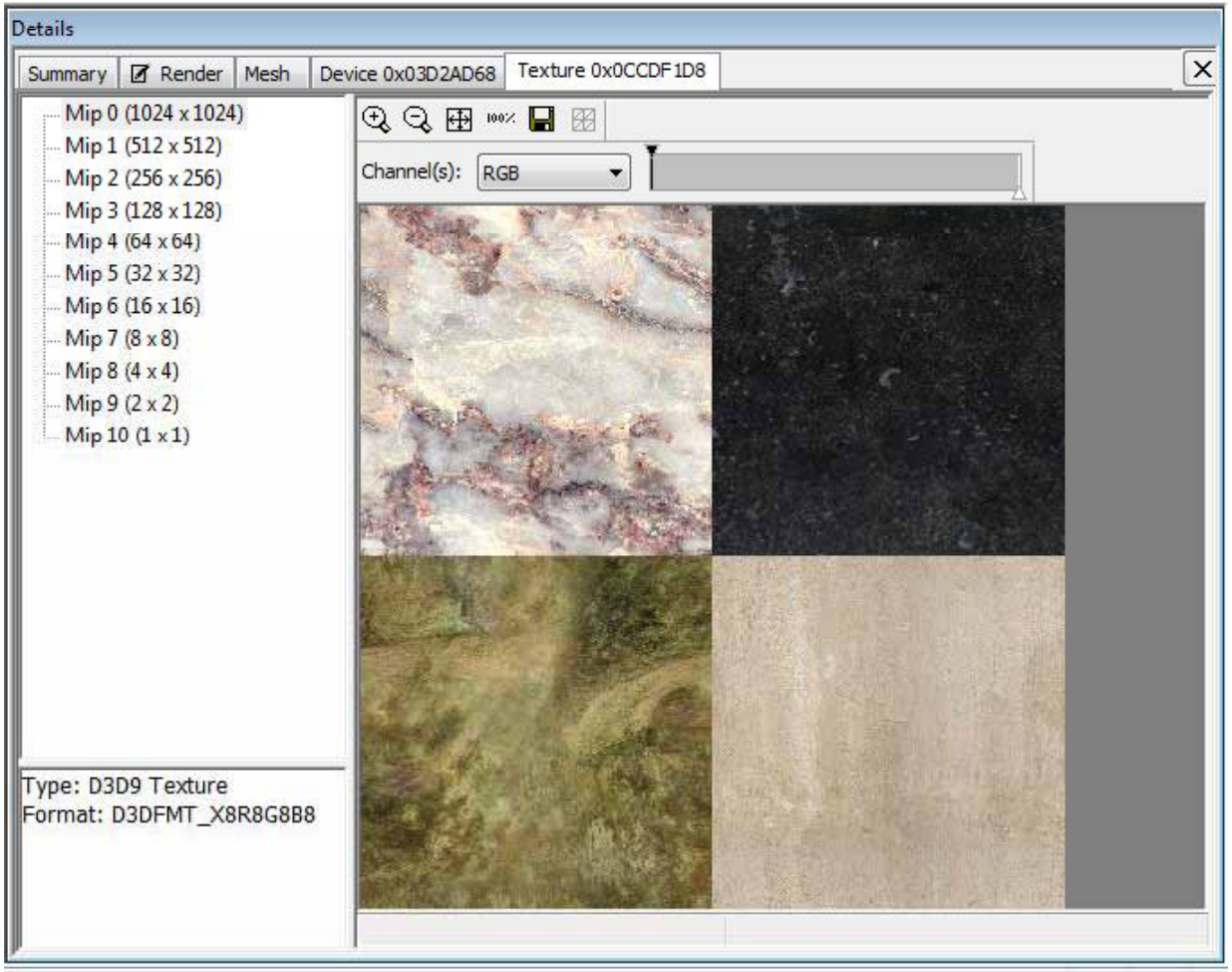
- Checking for sampler issues



- Samplers exist, values look ok

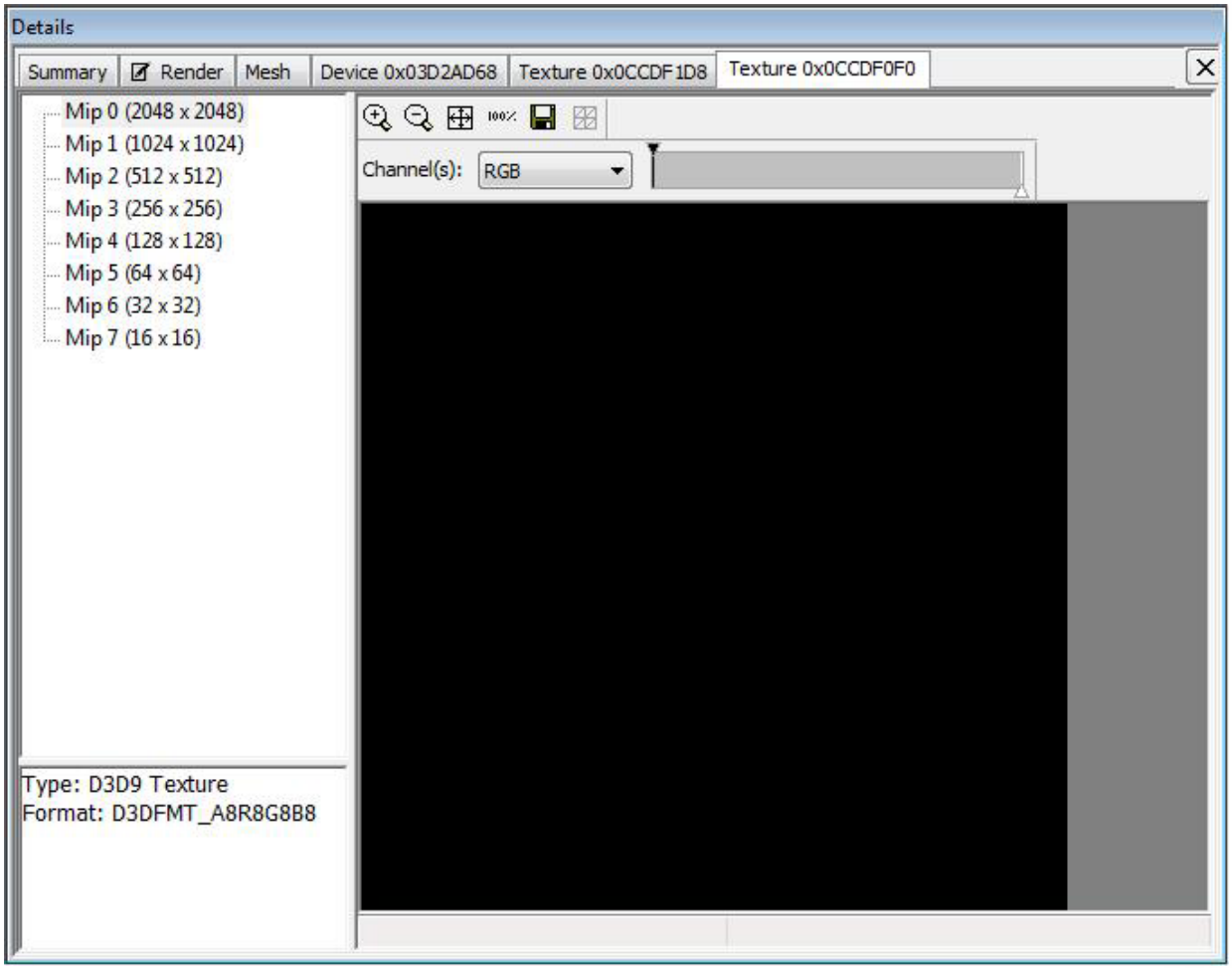
Blank Screen

- Check texture sampler 0 - OK



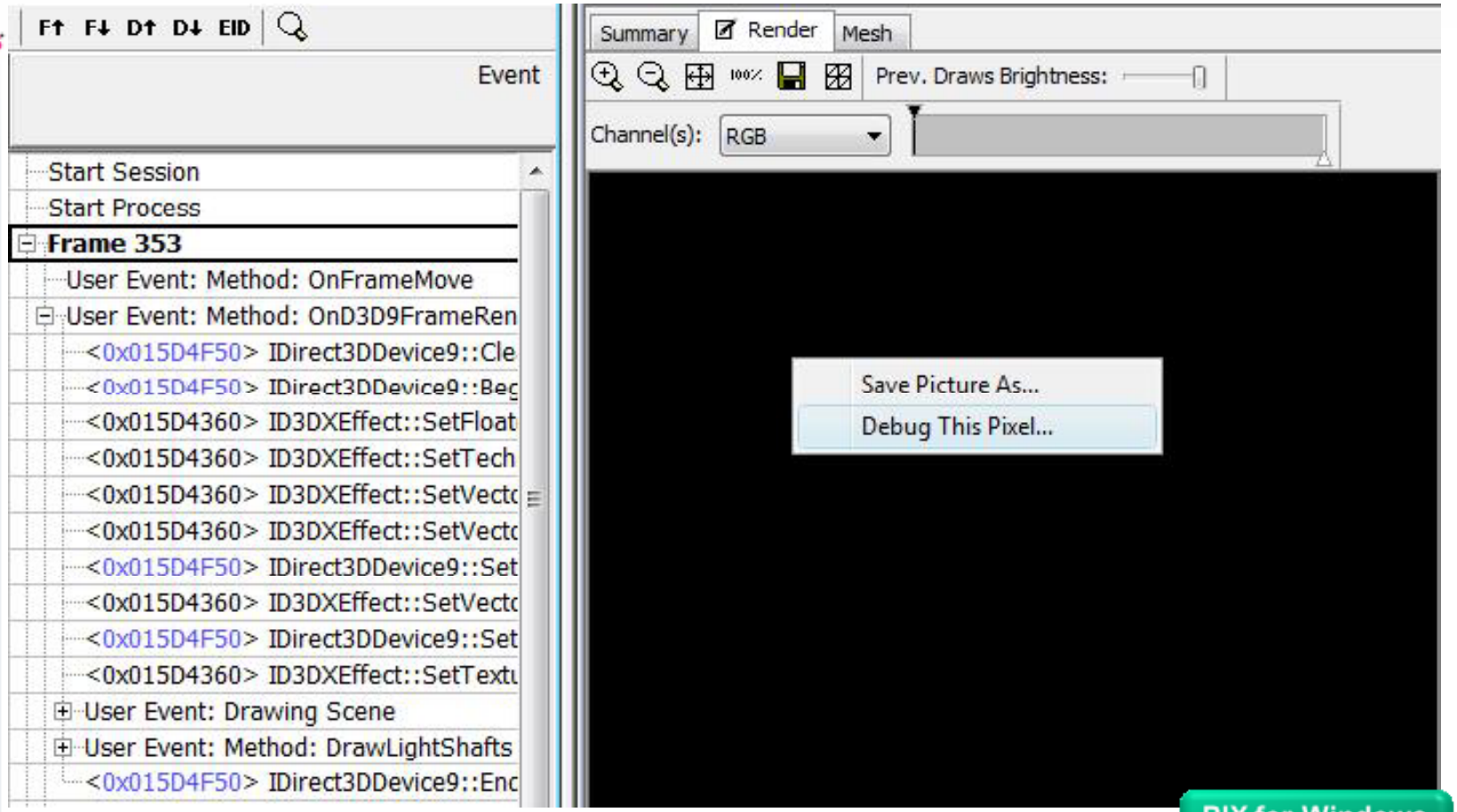
Blank Screen

- Sampler texture 1 should not be black



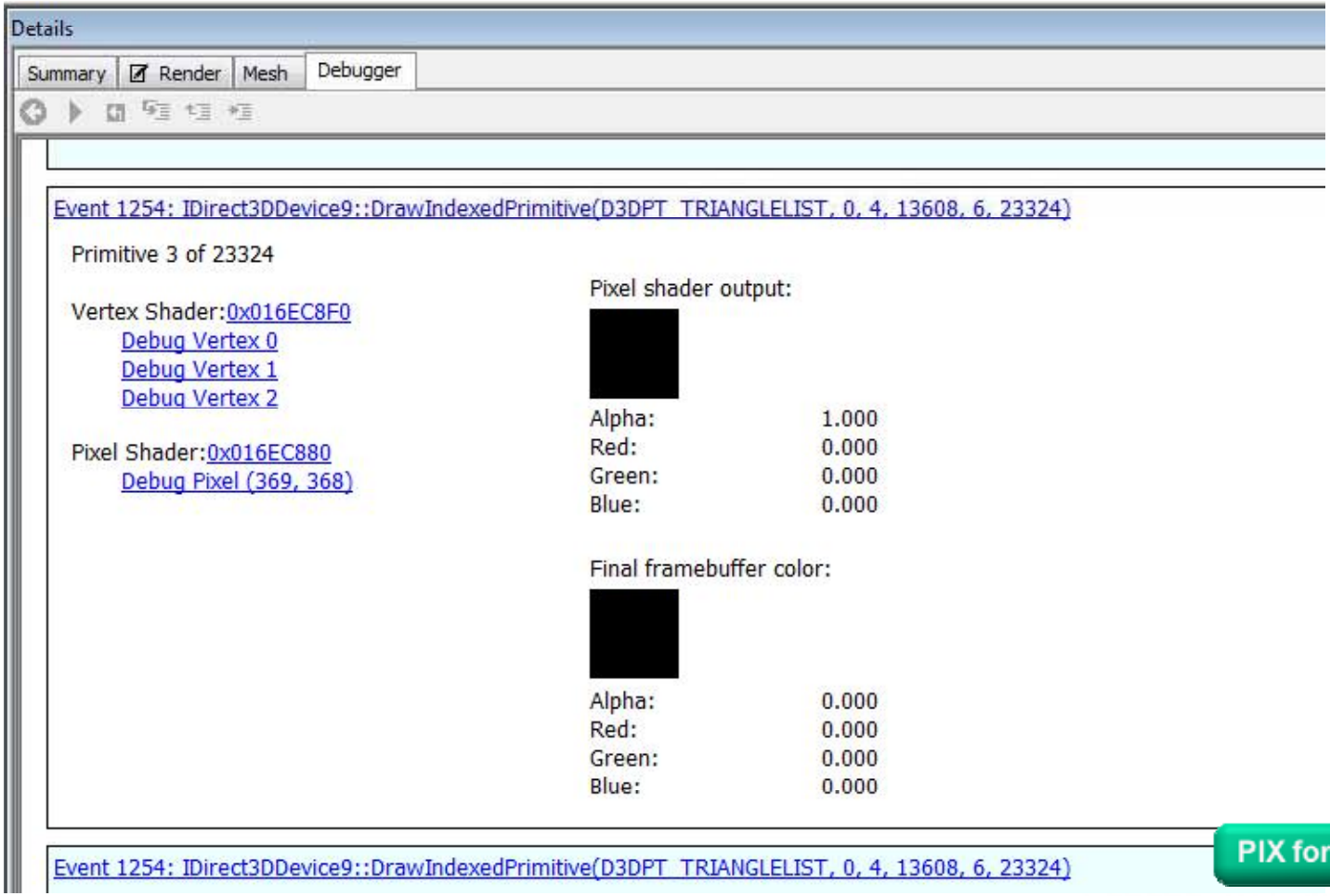
Blank Screen

- Render frame and select inaccurate pixel



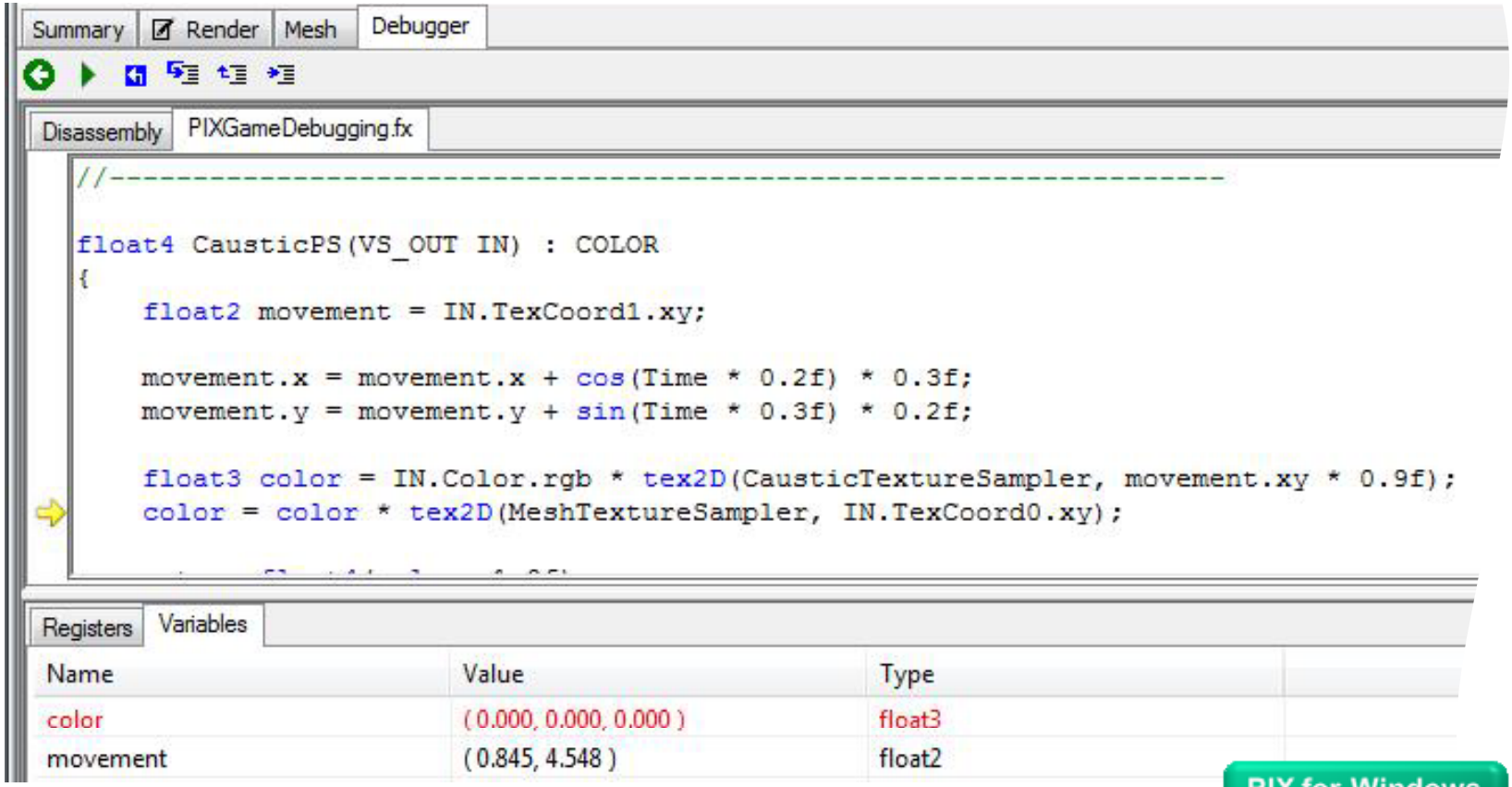
Blank Screen

- Pixel history shows all calls output black



Blank Screen

- Shader debugging proves black texture obliterates computed color



PIX for Windows

Blank Screen

Analysis:

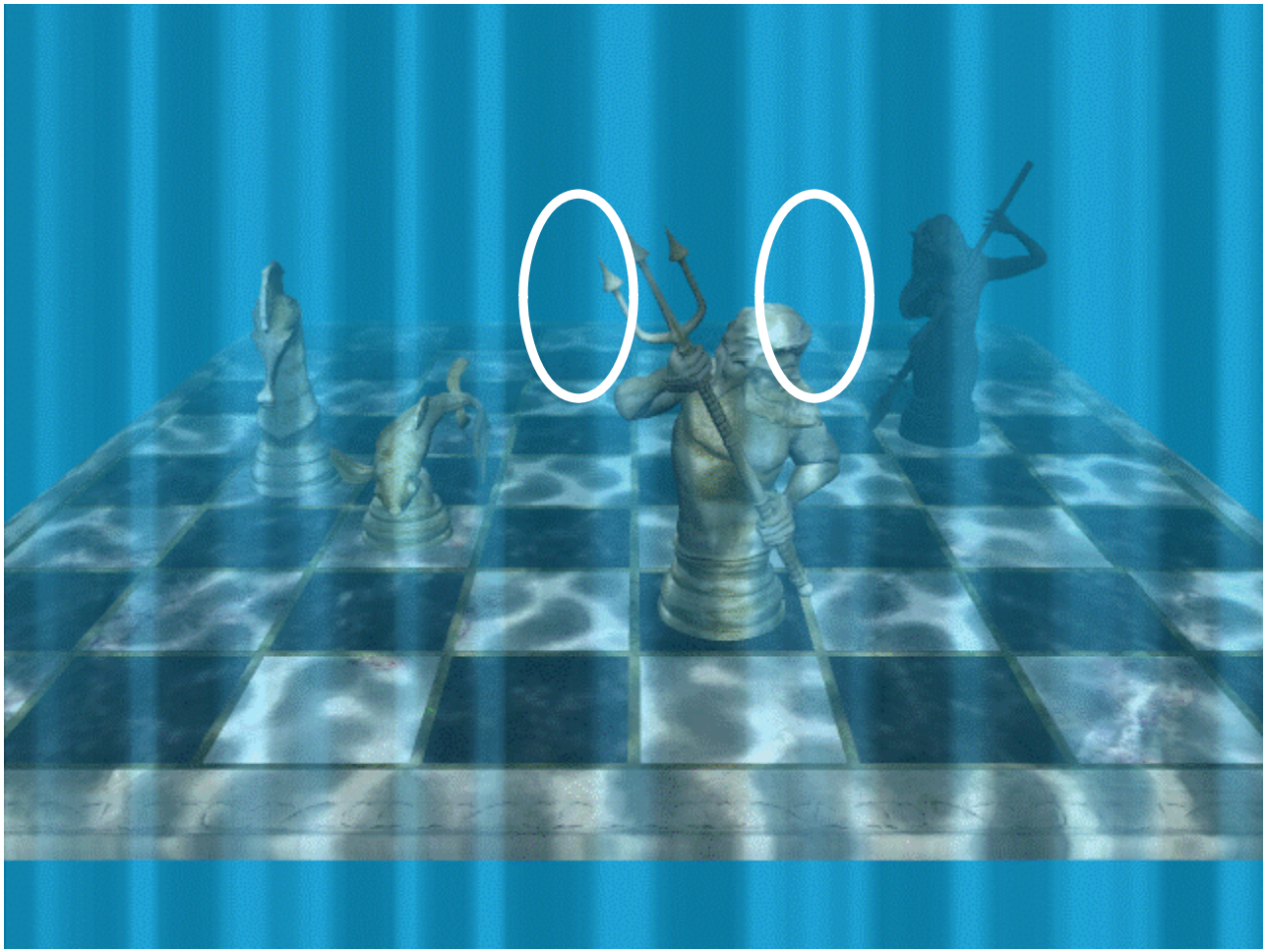
- Incorrect texture is used
- The texture is involved in all lighting operations, therefore everything is black
- Black is a common fallback for textures which were unable to be loaded at runtime

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



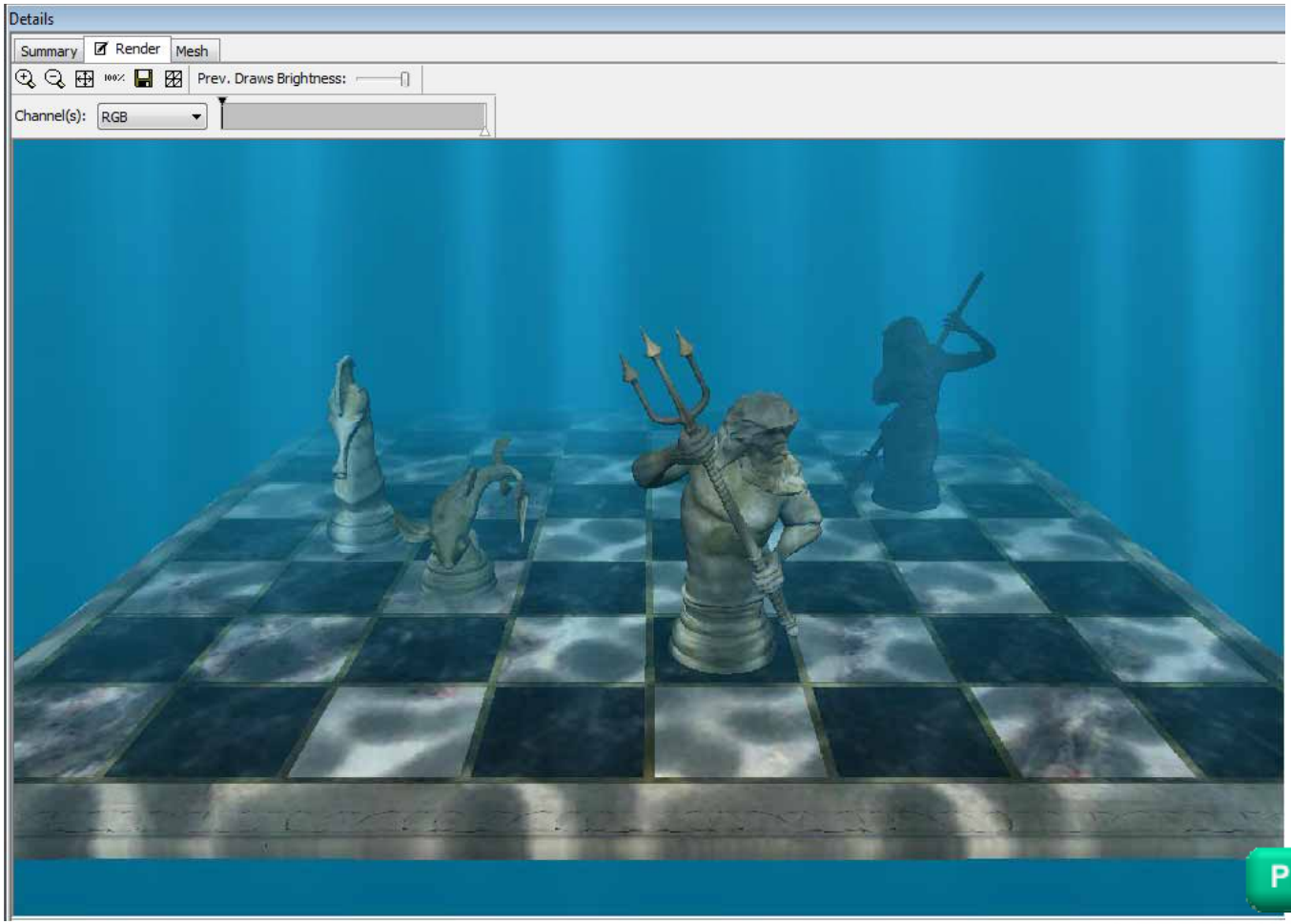
Missing Objects

Scenario:

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

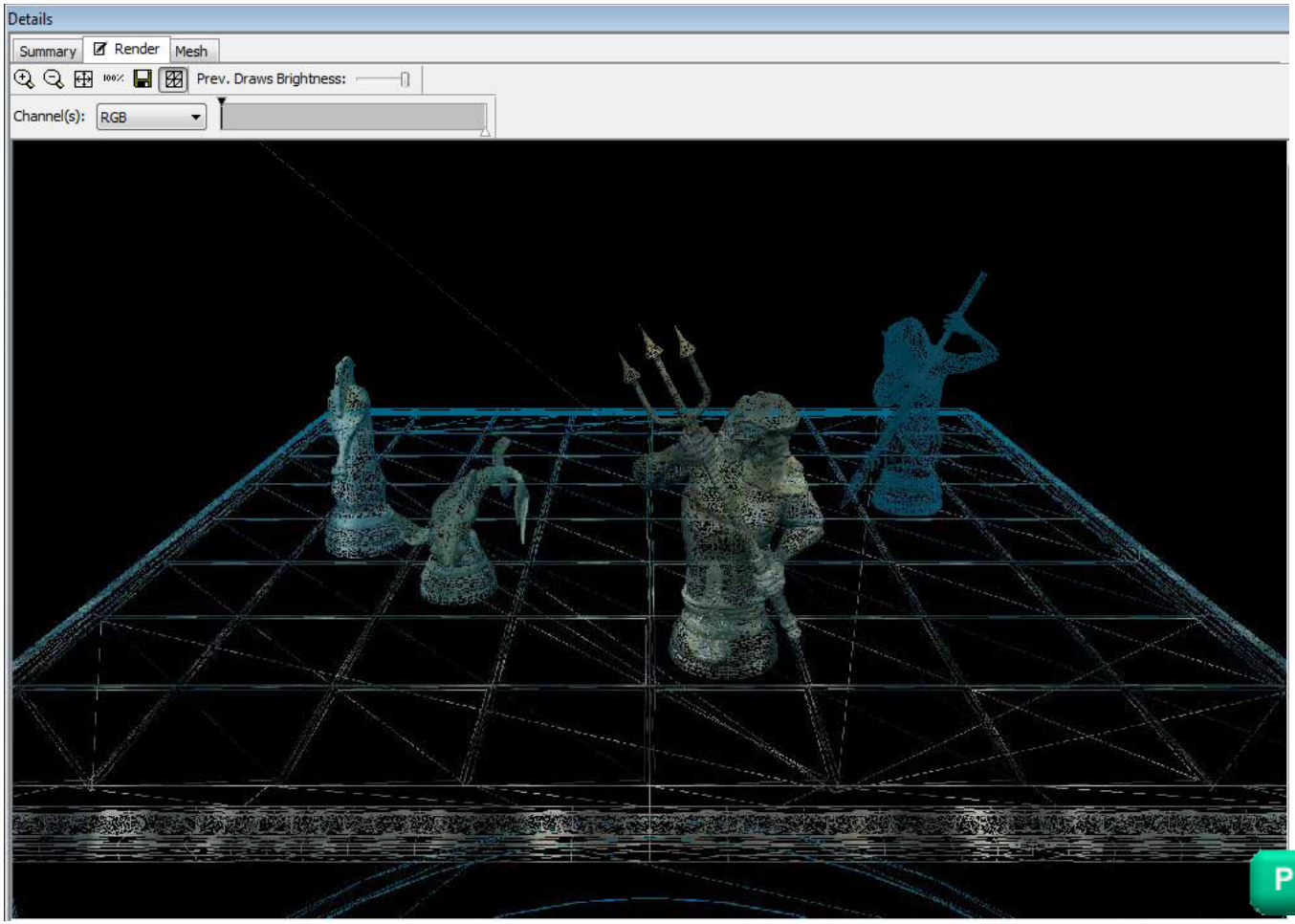
Missing Objects

- Rendered scene has missing objects



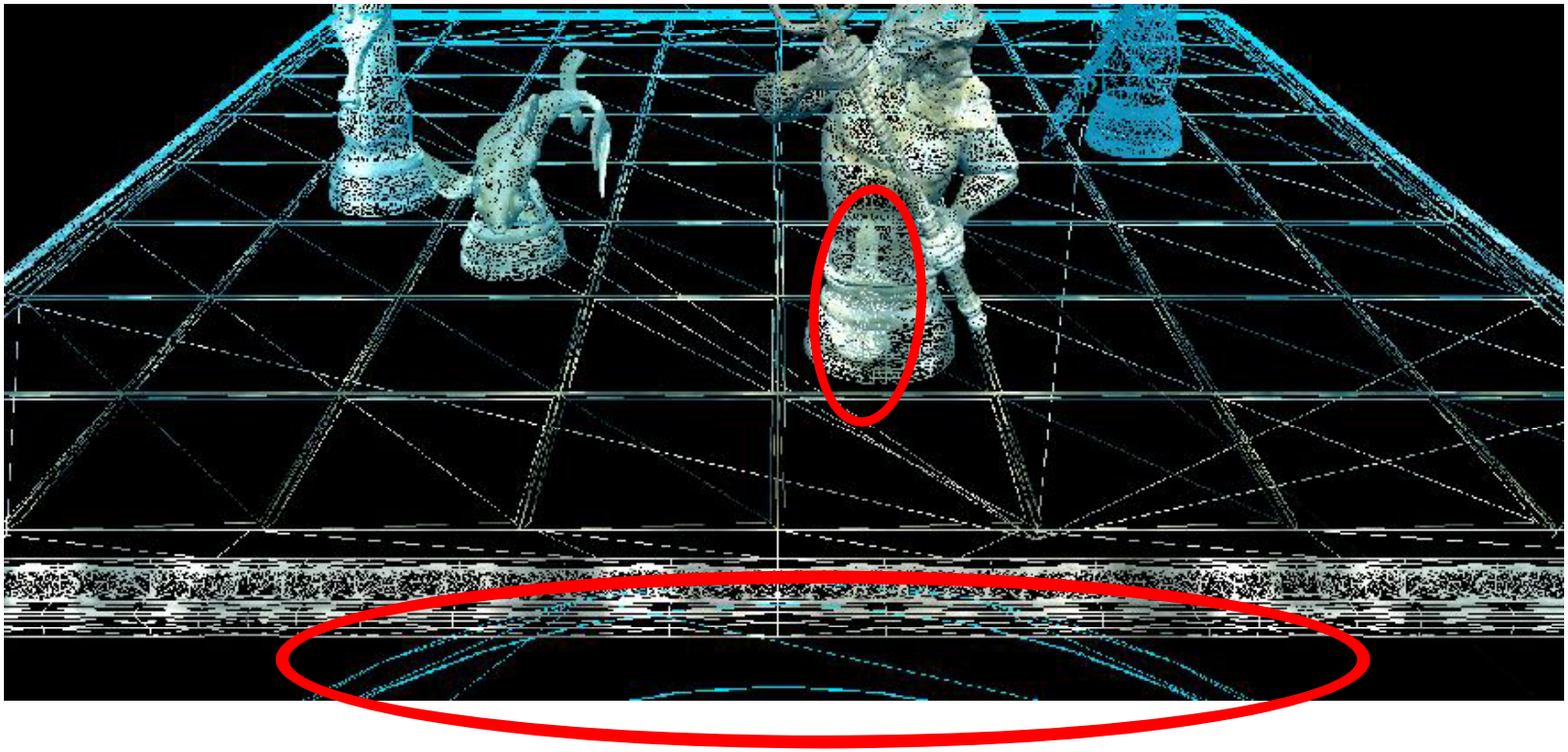
Missing Objects

- Check wireframe geometry of scene



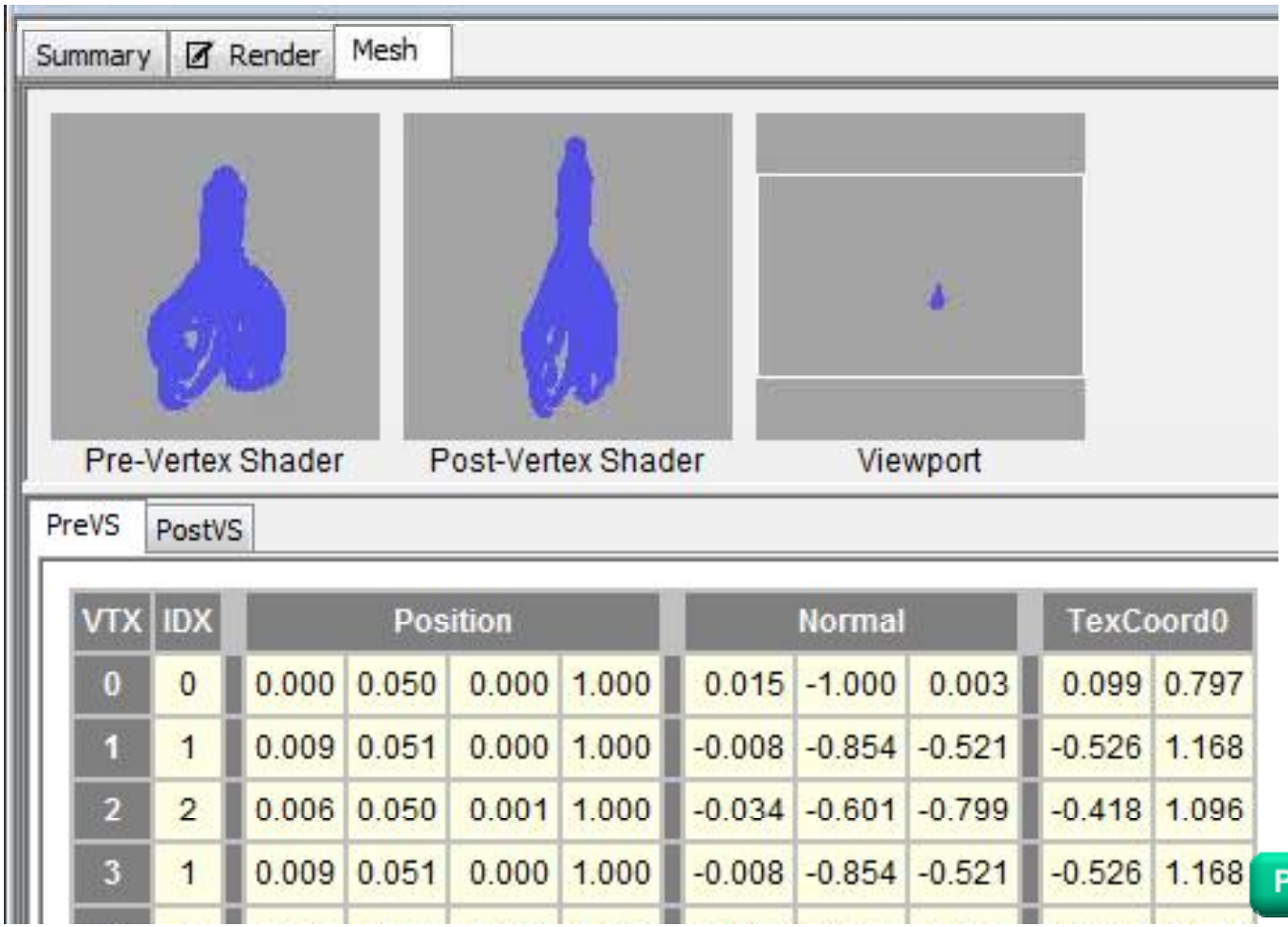
Missing Objects

- Suspicious artifacts present



Missing Objects

- Incorrect vertex shader input

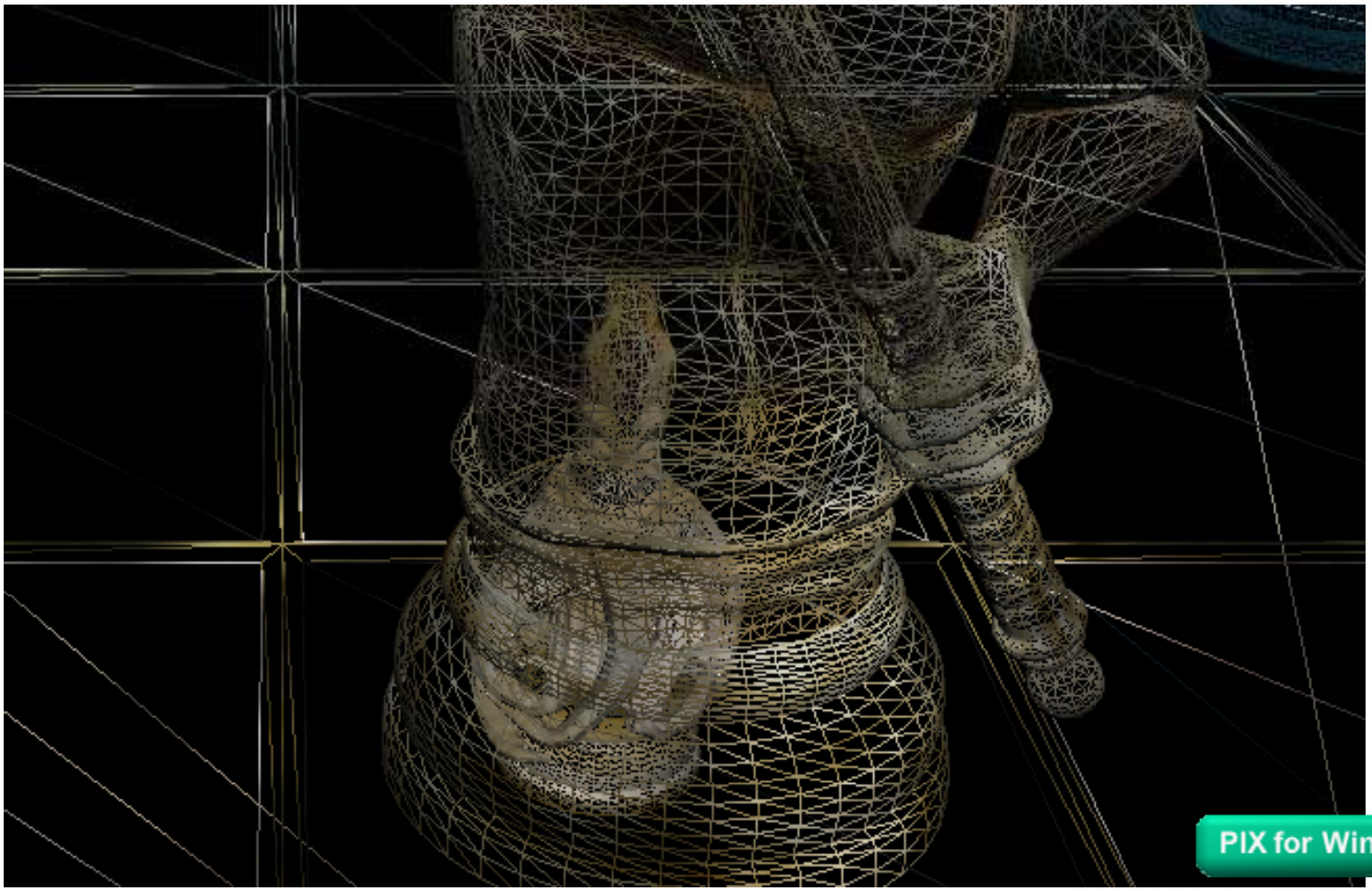


VTX	IDX	Position				Normal			TexCoord0	
0	0	0.000	0.050	0.000	1.000	0.015	-1.000	0.003	0.099	0.797
1	1	0.009	0.051	0.000	1.000	-0.008	-0.854	-0.521	-0.526	1.168
2	2	0.006	0.050	0.001	1.000	-0.034	-0.601	-0.799	-0.418	1.096
3	1	0.009	0.051	0.000	1.000	-0.008	-0.854	-0.521	-0.526	1.168

PIX for Windows

Missing Objects

- Yields unexpected output



Missing Objects

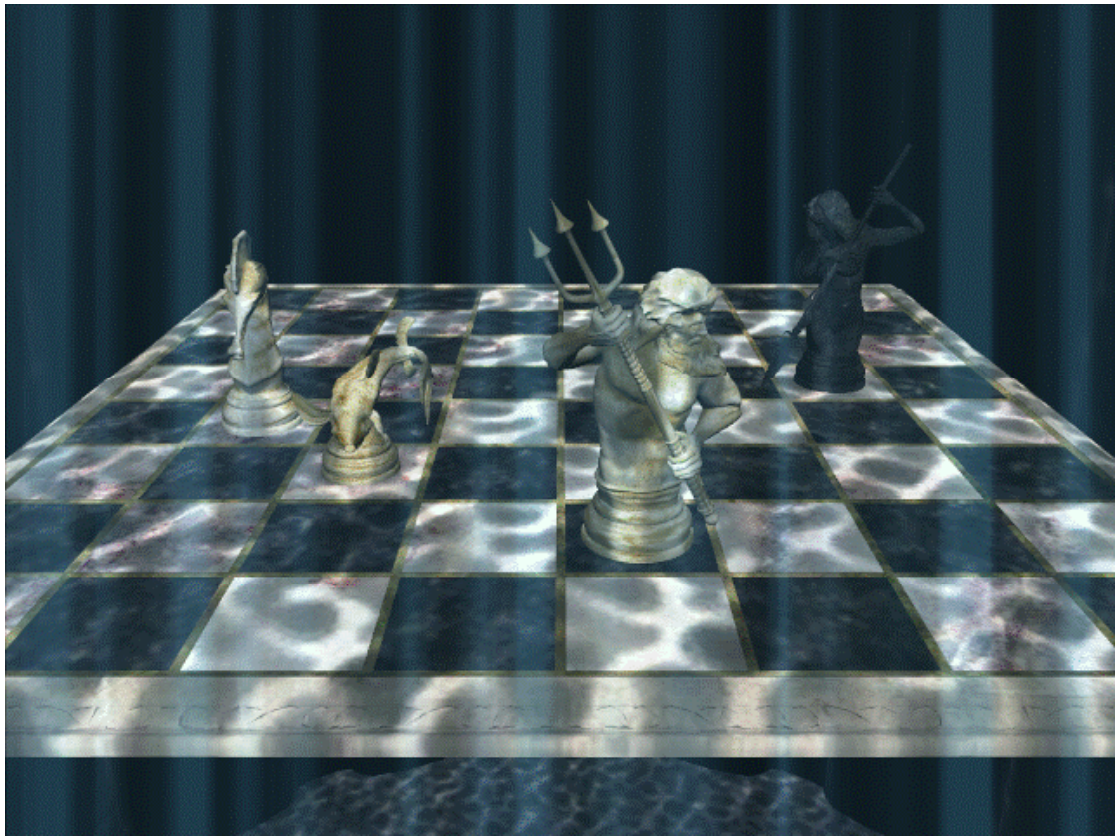
- Incorrect input & fogged out

PreVS | PostVS

Prim	VTX	IDX	Position				Diffuse	Fog	TexCoord0		TexCoor	
P0	0	5158	-13.523	-33.604	10.854	10.953	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.155	0.000	0.500	-6.722	8.
	1	5159	-13.357	-34.861	9.552	9.651	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.090	0.006	0.422	-6.639	7.
	2	5160	-13.357	-32.347	12.156	12.255	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.218	0.006	0.578	-6.639	9.
P1	3	5160	-13.357	-32.347	12.156	12.255	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.218	0.006	0.578	-6.639	9
	4	5159	-13.357	-34.861	9.552	9.651	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.090	0.006	0.422	-6.639	
	5	5161	-12.049	-29.956	14.632	14.731	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.336	0.054	0.727	-5.989	1
	6	5160	-13.357	-32.347	12.156	12.255	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.218	0.006	0.578	-6.639	
P2	7	5161	-12.049	-29.956	14.632	14.731	D3DCOLOR_ARGB(0x00,0x00,0x00,0x00)	-2.336	0.054	0.727	-5.989	

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

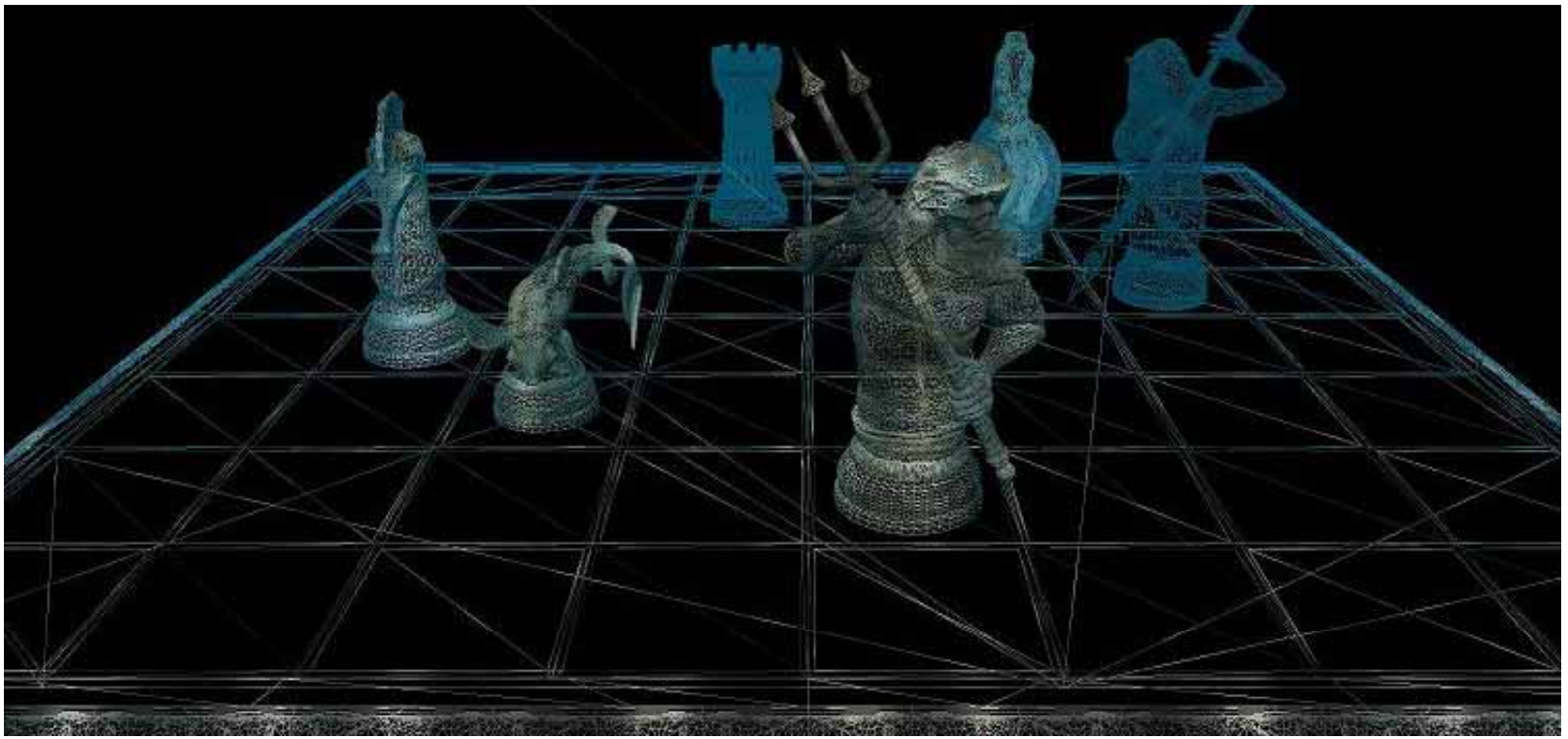
Flickering

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

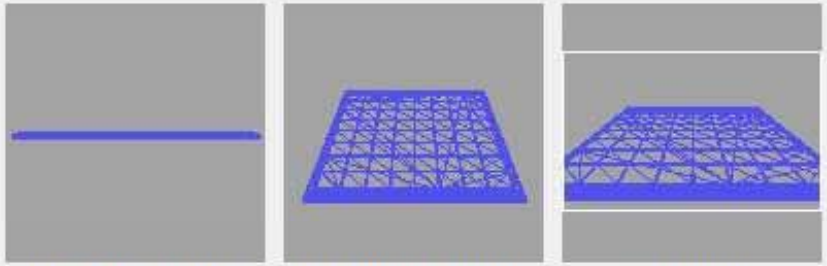
Flickering

- Examine wireframe for obvious z-fighting



Flickering

- Examine mesh view for hidden artifacts

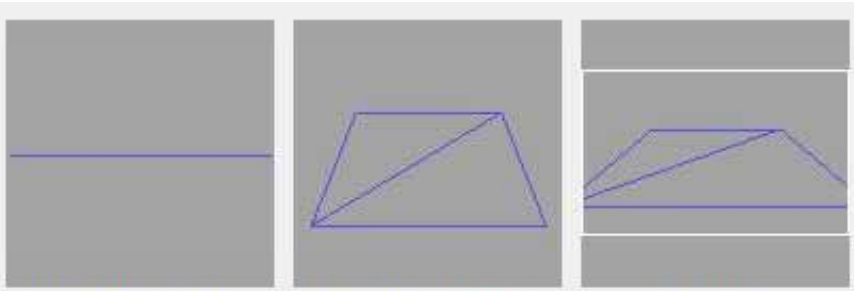


PreVS | PostVS

Prim	VTX	IDX	Position				Diffuse	Fog	TexCoord0		TexCoord1	
P0	0	4	-3.647	0.206	5.252	5.351	D3DCOLOR_ARGB(0x00,0x33,0x33,0x33)	0.387	0.756	0.968	-1.813	4.014
	1	5	-3.647	0.745	5.810	5.910	D3DCOLOR_ARGB(0x00,0x33,0x33,0x33)	0.292	0.701	0.968	-1.813	4.432
	2	6	-3.647	1.285	6.369	6.468	D3DCOLOR_ARGB(0x00,0x33,0x33,0x33)	0.195	0.645	0.968	-1.813	4.851
P1	3	7	-3.647	0.206	5.252	5.351	D3DCOLOR_ARGB(0xa3,0xd6,0xd6,0xd6)	0.387	0.756	0.968	-1.813	4.014
	4	8	-3.647	1.285	6.369	6.468	D3DCOLOR_ARGB(0xa3,0xd6,0xd6,0xd6)	0.195	0.645	0.968	-1.813	4.851
	5	9	-3.647	-0.872	4.135	4.235	D3DCOLOR_ARGB(0xa3,0xd6,0xd6,0xd6)	0.566	0.866	0.968	-1.813	3.176
	6	10	-3.647	-0.872	4.135	4.235	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.566	0.866	0.968	-1.813	3.176

Flickering

- Hidden mesh subset uncovered



PreVS | PostVS

Prim	VTX	IDX	Position				Diffuse	Fog	TexCoord0		TexCoord1	
P0	0	0	3.602	-1.883	3.026	3.126	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.726	1.000	1.000	1.790	2.344
	1	1	-3.602	-1.883	3.026	3.126	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.726	0.000	1.000	-1.790	2.344
	2	2	3.602	2.397	7.459	7.558	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.002	1.000	0.000	1.790	5.669
P1	3	3	-3.602	2.397	7.459	7.558	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.002	0.000	0.000	-1.790	5.669
	4	2	3.602	2.397	7.459	7.558	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.002	1.000	0.000	1.790	5.669
	5	1	-3.602	-1.883	3.026	3.126	D3DCOLOR_ARGB(0xb9,0xec,0xec,0xec)	0.726	0.000	1.000	-1.790	2.344

Flickering

Conclusion:

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

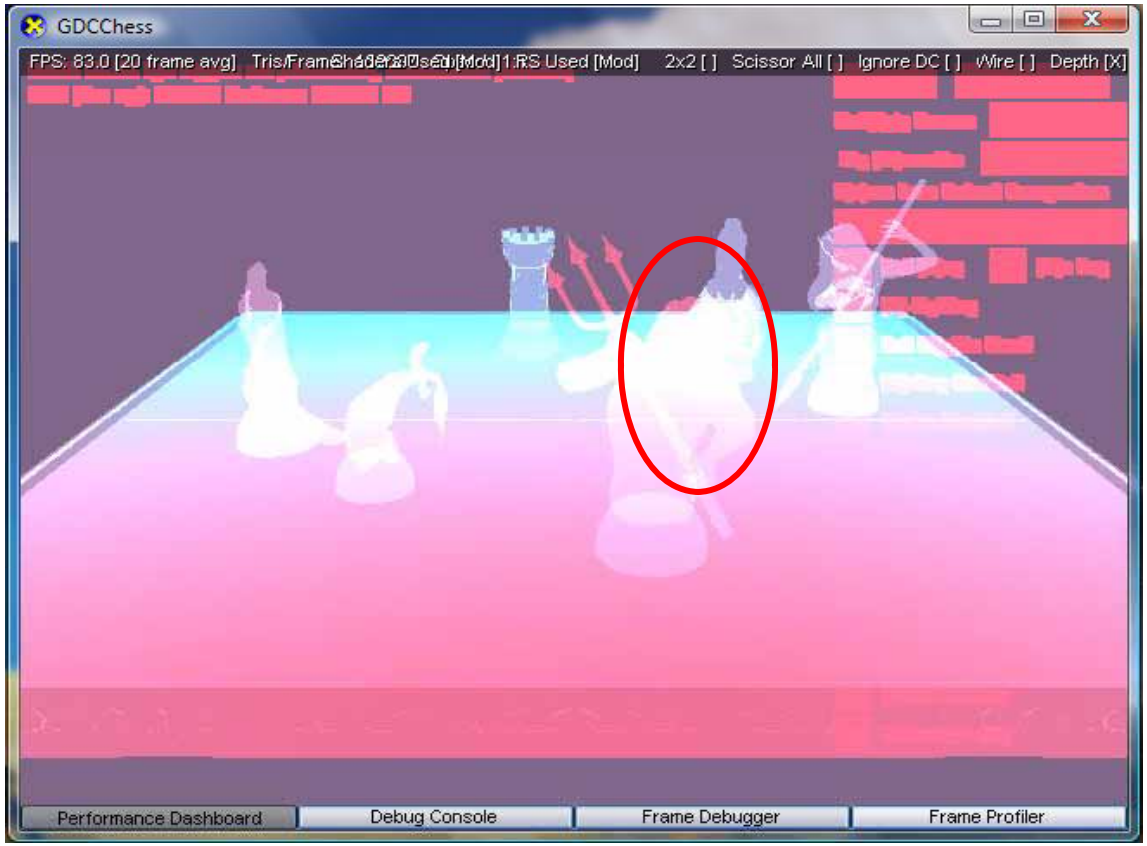
Bottleneck Analysis

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

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

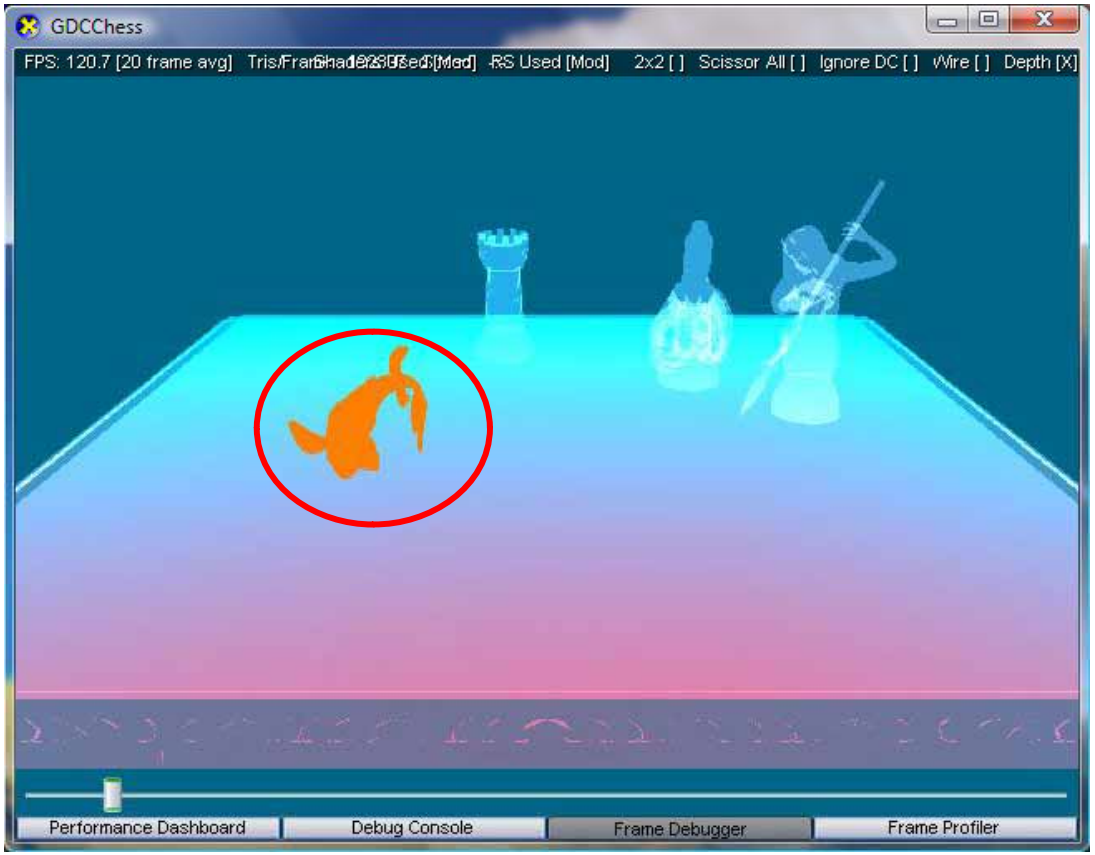
Culling & Render Order

- Look at the overdraw in the tool



Culling & Render Order

- Scroll through the draw calls to see how the frame is composed

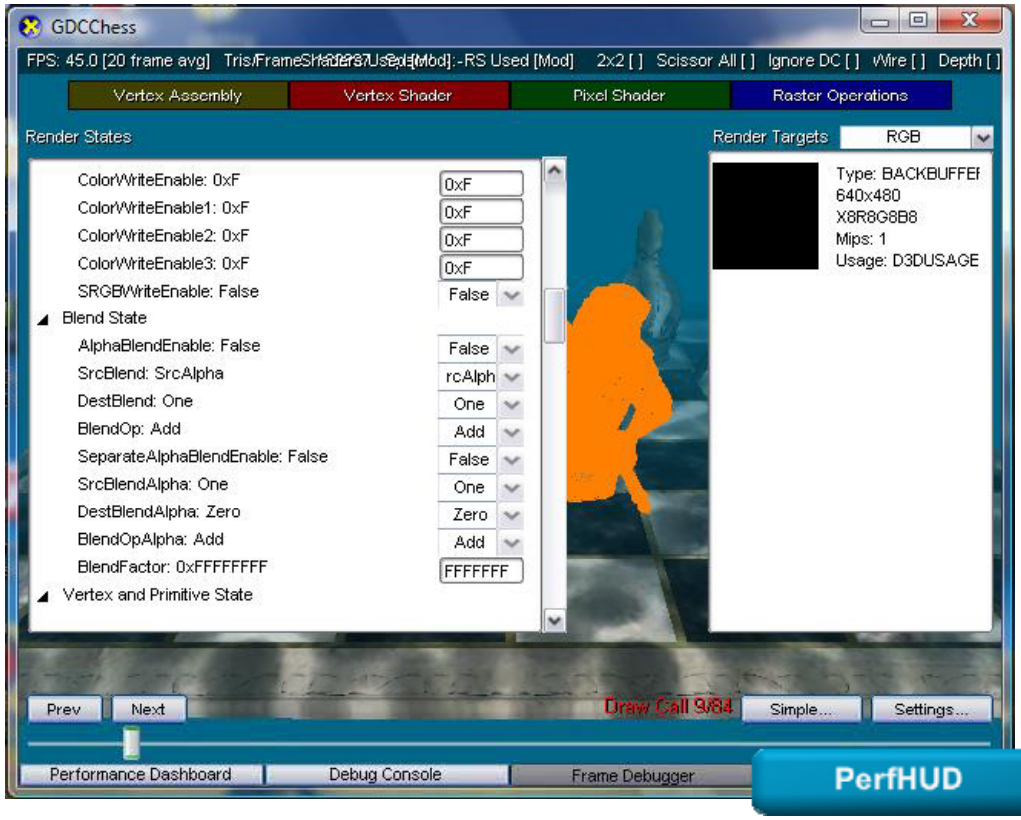


Culling & Render Order

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

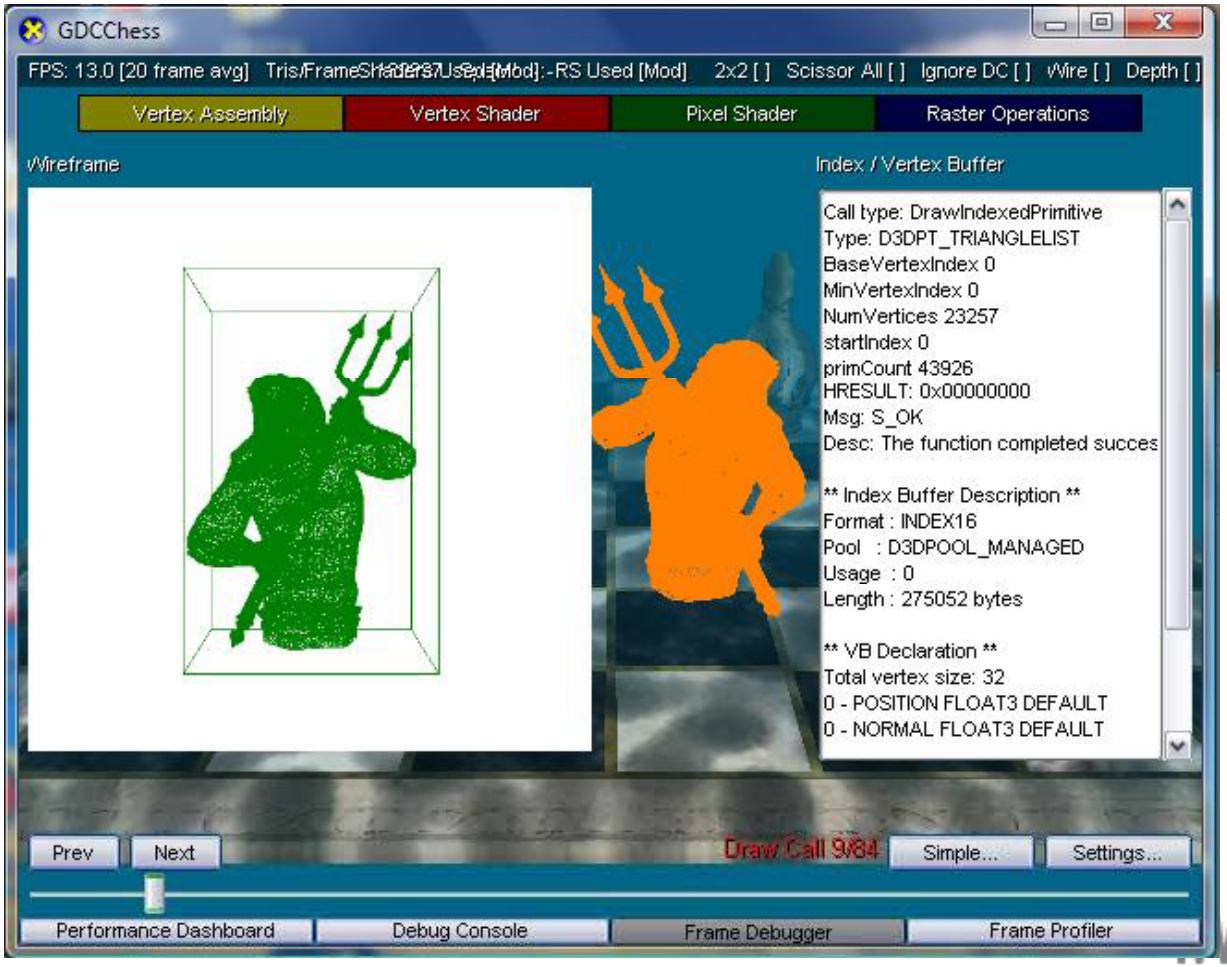
Culling & Render Order

- 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

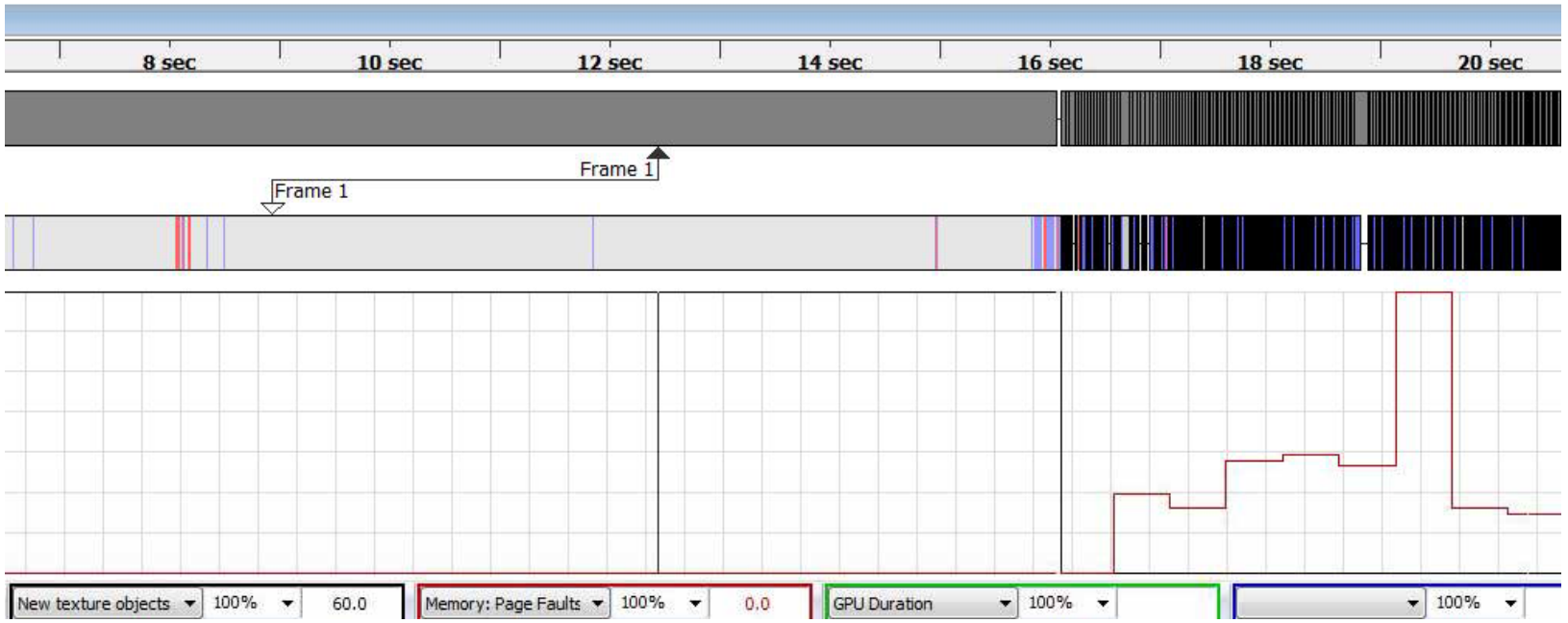
- Performance is slow
- But everything looks correct
- Thrashing of system resources

Buffer Sizes

- There could be lots of swapping occurring

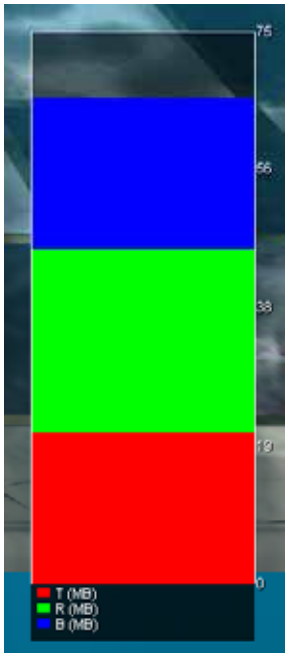
Buffer Sizes

- 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

Objects

Address	Type	Created By	Creation	Destruction	Status	App Refs	Size	Pool	Usage	Format	Width	Height	Depth	Mips
0x029B8680	D3D9 Vertex Buffer	Application	1	649	Alive	1	435,584 bytes	Managed		D3DFMT_VERTEXDATA				
0x090CDD10	D3D9 Surface	Direct3D	1	649	Alive	0	8 bytes	Managed		D3DFMT_DXT1	4	4		
0x0AA0D598	D3D9 Surface	Direct3D	1	649	Alive	0	16 bytes	Managed		D3DFMT_X8R8G8B8	2	2		
0x02904AF0	D3D9 Surface	Direct3D	1	649	Alive	0	1,228,800 bytes	Default	DepthStencil	D3DFMT_D24X8	640	480		
0x0AB4DC28	D3D9 Surface	Direct3D	1	649	Alive	0	262,144 bytes	Managed		D3DFMT_X8R8G8B8	256	256		
0x090CBE30	D3D9 Texture	Application	1	649	Alive	1	2,744 bytes	Managed		D3DFMT_DXT1	64	64		7
0x0AB4EBE8	D3D9 Surface	Direct3D	1	649	Alive	0	1,024 bytes	Managed		D3DFMT_X8R8G8B8	16	16		
0x090CDDB8	D3D9 Surface	Direct3D	1	649	Alive	0	8 bytes	Managed		D3DFMT_DXT1	2	2		
0x0AA0E5E0	D3D9 Index Buffer	Application	1	649	Alive	1	49,152 bytes	Default	WriteOnly	D3DFMT_INDEX16				
0x0AA0D640	D3D9 Surface	Direct3D	1	649	Alive	0	4 bytes	Managed		D3DFMT_X8R8G8B8	1	1		
0x0AB4DCD0	D3D9 Surface	Direct3D	1	649	Alive	0	65,536 bytes	Managed		D3DFMT_X8R8G8B8	128	128		
0x0AB4EC90	D3D9 Surface	Direct3D	1	649	Alive	0	256 bytes	Managed		D3DFMT_X8R8G8B8	8	8		
0x0AD17F00	D3D9 State Block	Application	1	649	Alive	1								

169 of 169 objects displayed

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

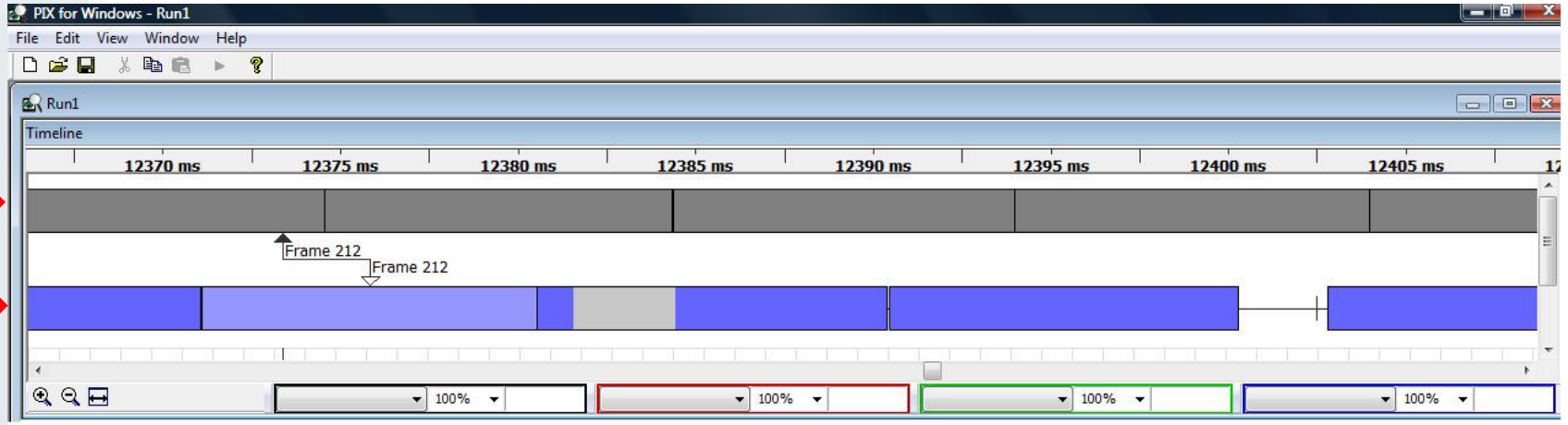
Buffer Sizes

- Only use data where necessary
- Pack data buffers with a smaller vdecl
- Use LOD techniques to reduce the amount of data needed

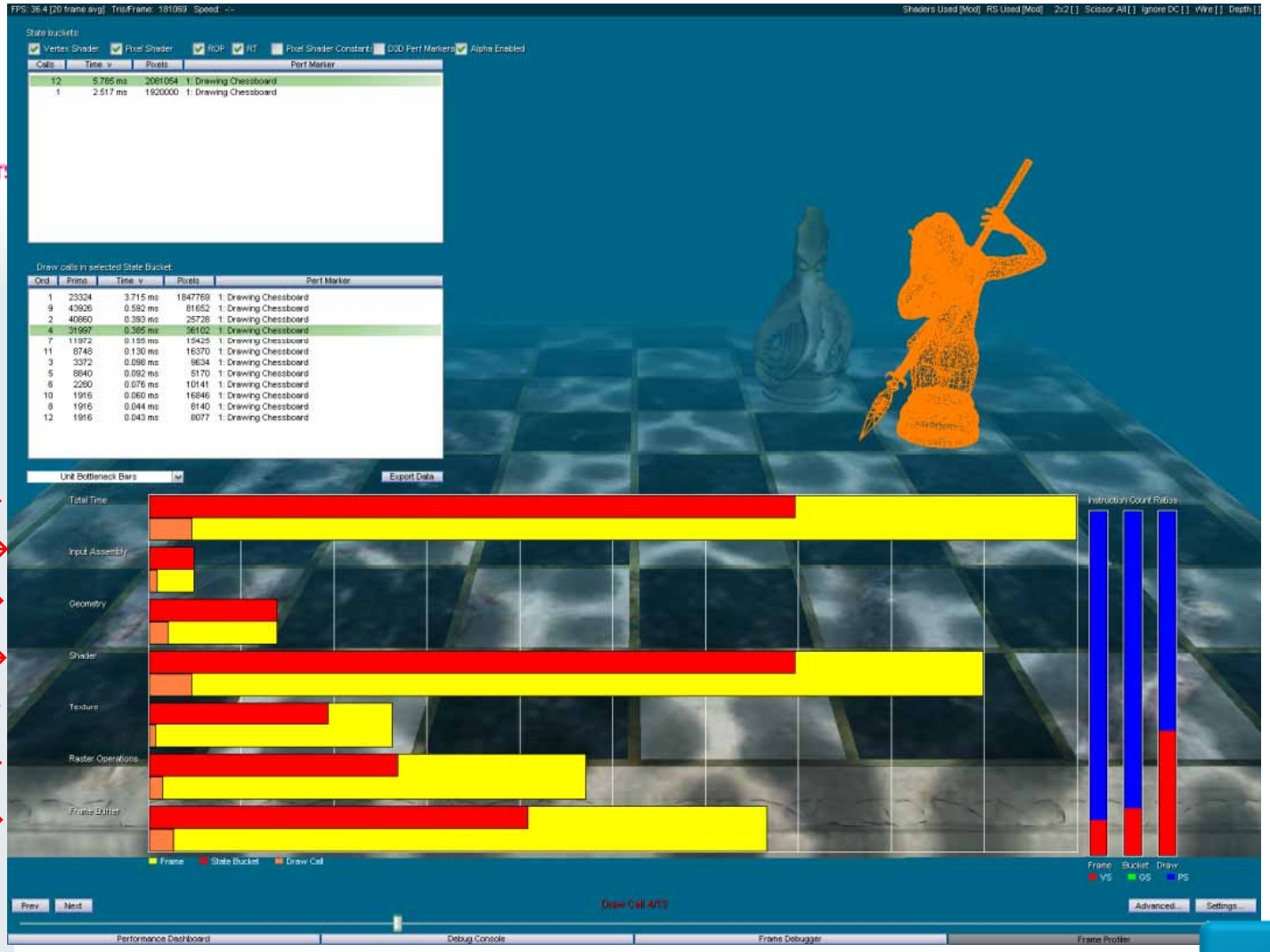
- Use a paging algorithm for loading data
- Reuse Render targets when possible

Inefficient Code

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

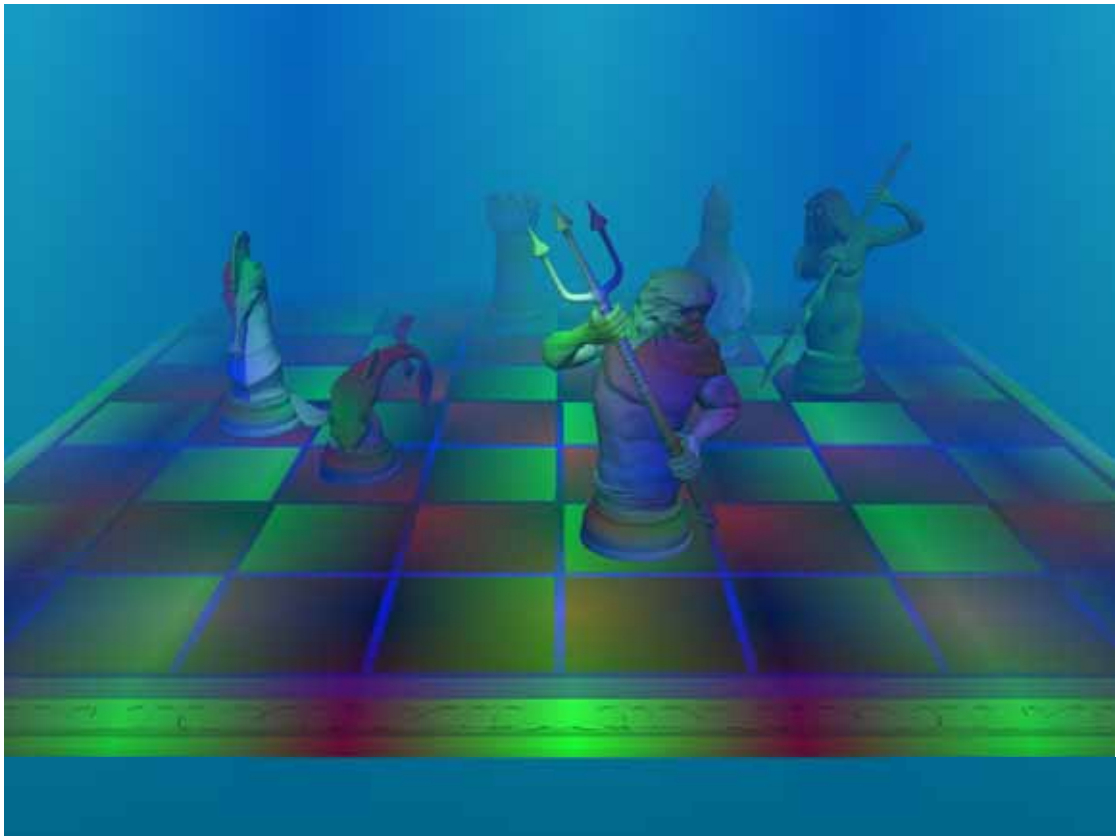


Inefficient Code



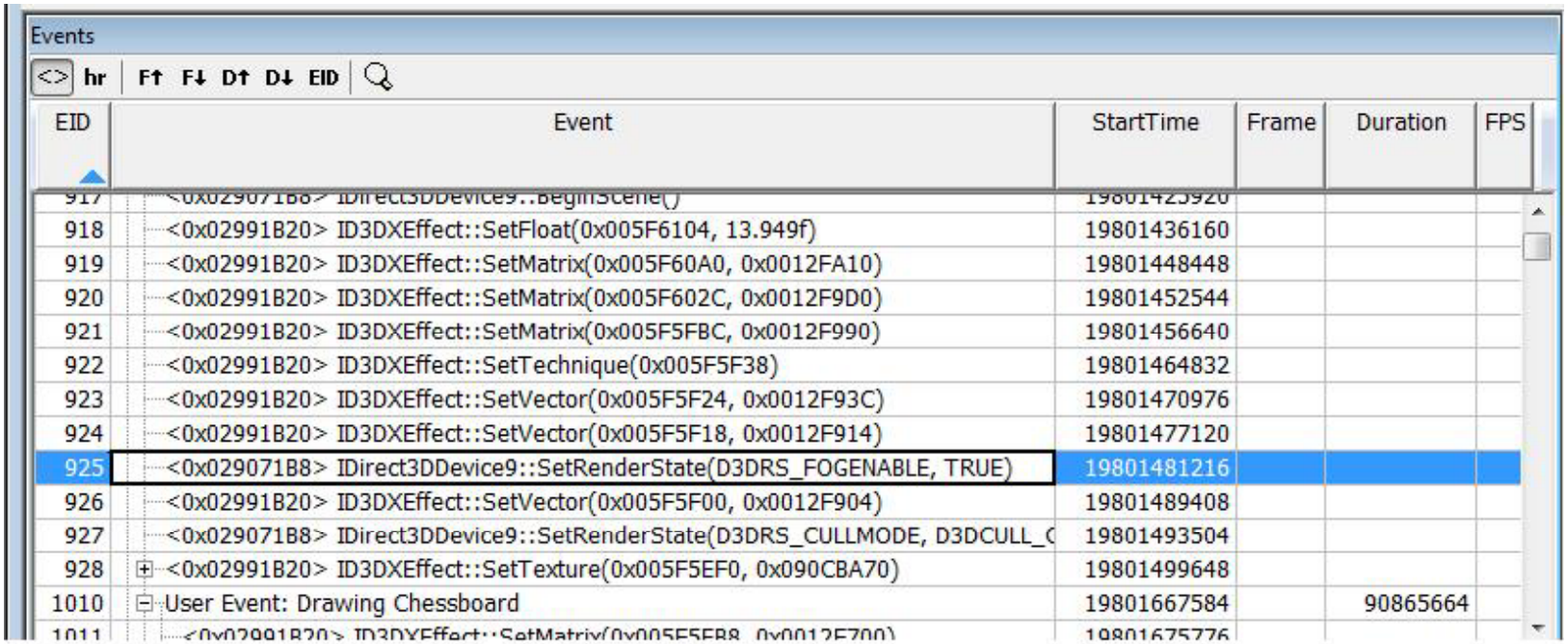
Inefficient Code

- Adjust
  - render size, texture sizes, cull objects



Inefficient Code

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

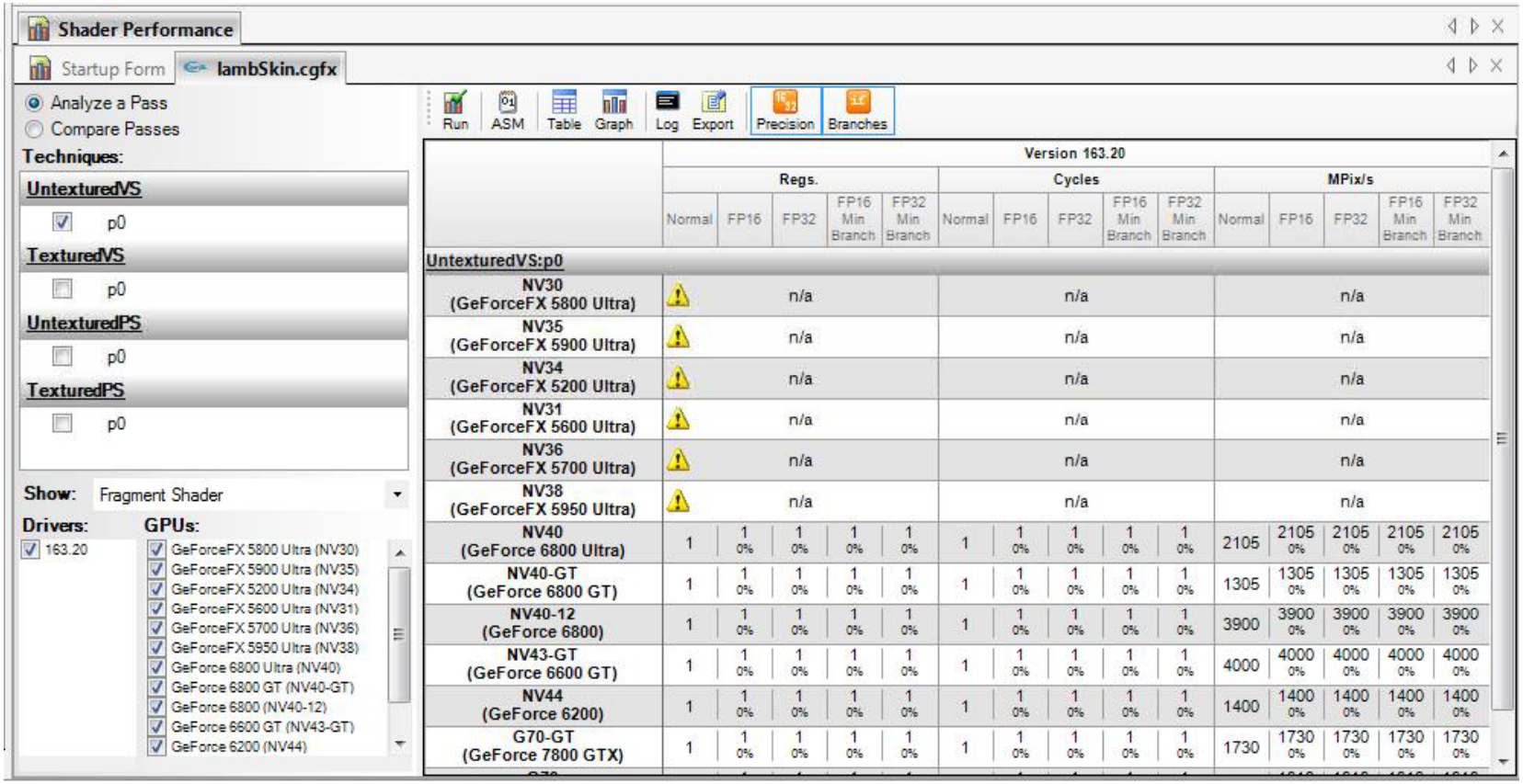


PIX for Windows

WWW.GDCONF.COM

Inefficient Shaders

- Use a tool to analyze your shader

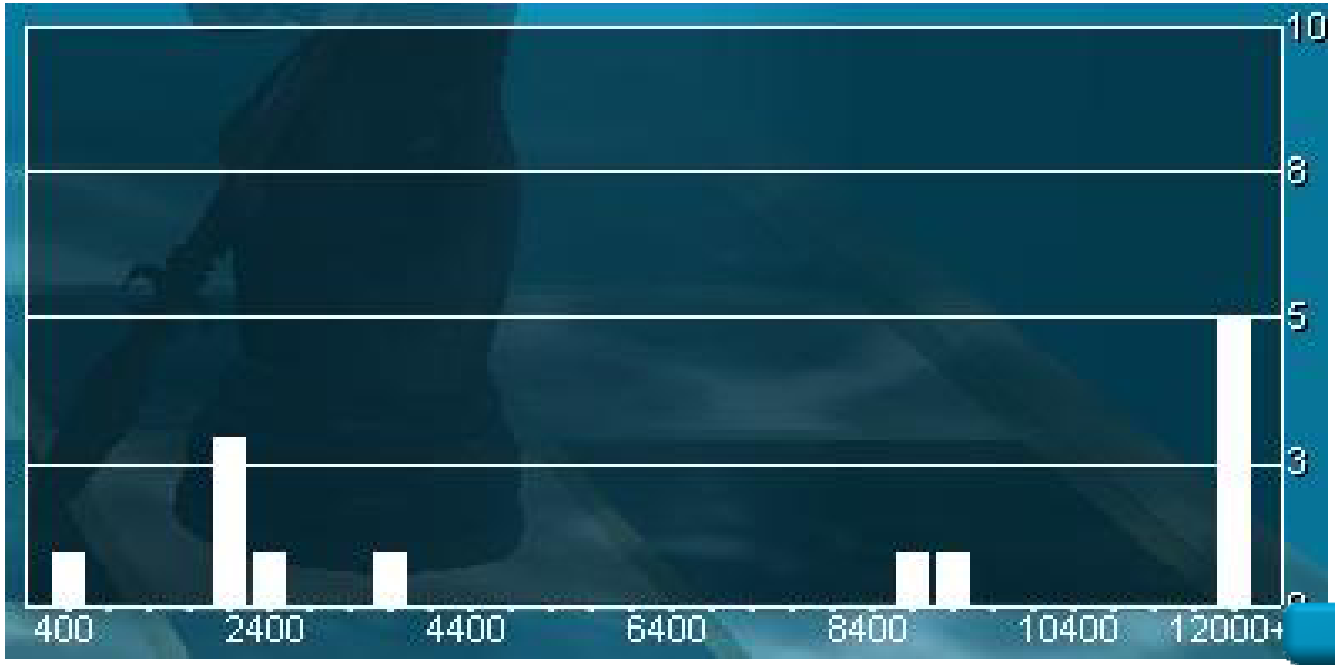


Inefficient Shaders

- 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