



NVIDIA®

Performance Tools

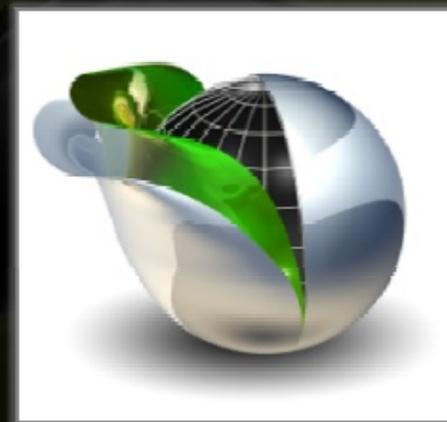
Jeff Kiel

Manager, Graphics Tools

Performance Tools Agenda



- **Eric Preisz: Optimizing Marble Blast Ultra**
- **Introducing PerfHUD 6.0**
- **PerfKit**
 - PerfSDK: GPU & Driver Performance Data
 - Graphic Remedy's gDEDebugger
- **ShaderPerf**
- **FX Composer 2.5: Shader Debugger**





Optimizing Marble Blast Ultra

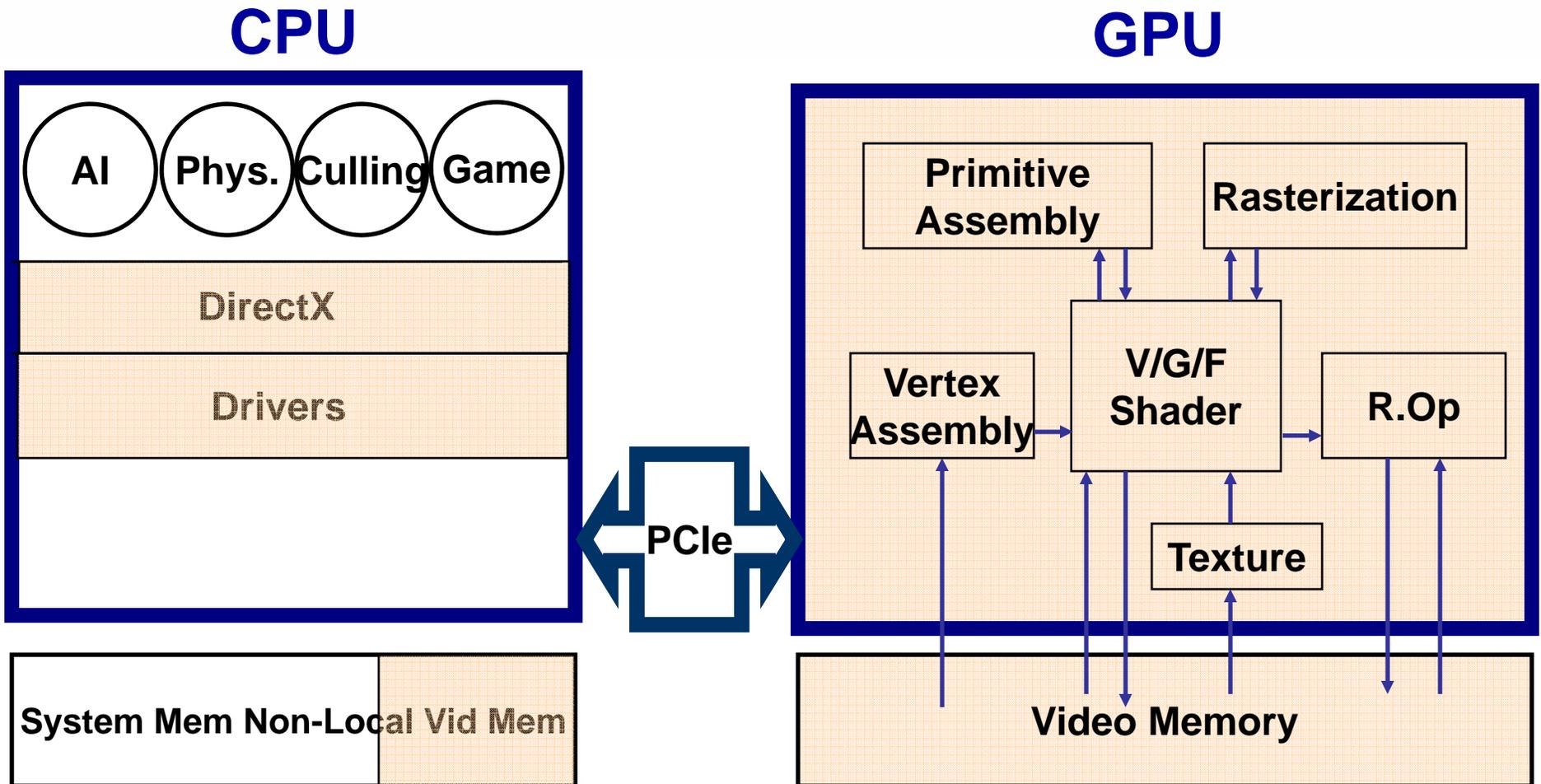
By: Eric Preisz

Marble Blast Ultra



130 Hz

The Game Pipeline



Levels of Optimization

System - % of hardware utilization

Application - Class/function/algorithm

Micro - Line by line optimizations.

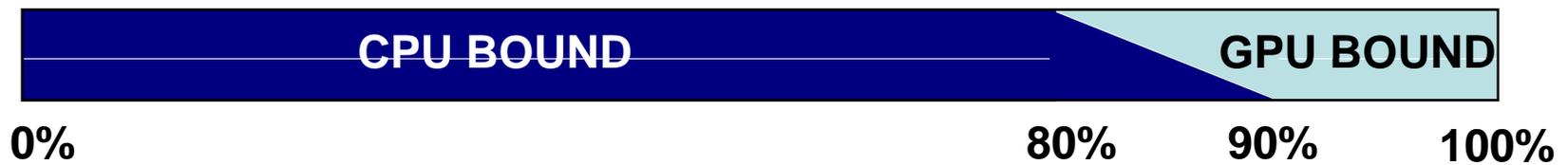
Simple Mistakes - Resource Allocations

The screenshot displays a 3D game engine interface. At the top, a status bar shows performance metrics: FPS: 43.7 [20 frame avg], Tris/Frame: 33827, Speed: --, Shaders Used [Mod], RS Used [Mod], Scissor All [], Ignore DC [], vWire [], and Depth []. A large digital timer in the center of the scene reads 00:15.52. The scene itself features a green checkered floor, brown stone walls with arches, and a blue sky with stars. A semi-transparent wireframe sphere is visible in the background. In the bottom-left corner, a white debug console window is open, displaying the message: "Time: 656113.89 ms, IDirect3DDevice9::CreateVertexBuffer(216,0,16,1)". To the right of the console, a settings panel includes checkboxes for "Clear Log Each Frame" (unchecked), "Stop Logging" (unchecked), and "Display Messages From:" with "Application" and "PerHDD" checked. A "Save" button is located below these options. At the bottom of the interface, a navigation bar contains four tabs: "Performance Dashboard", "Debug Console", "Frame Debugger", and "Frame Profiler".

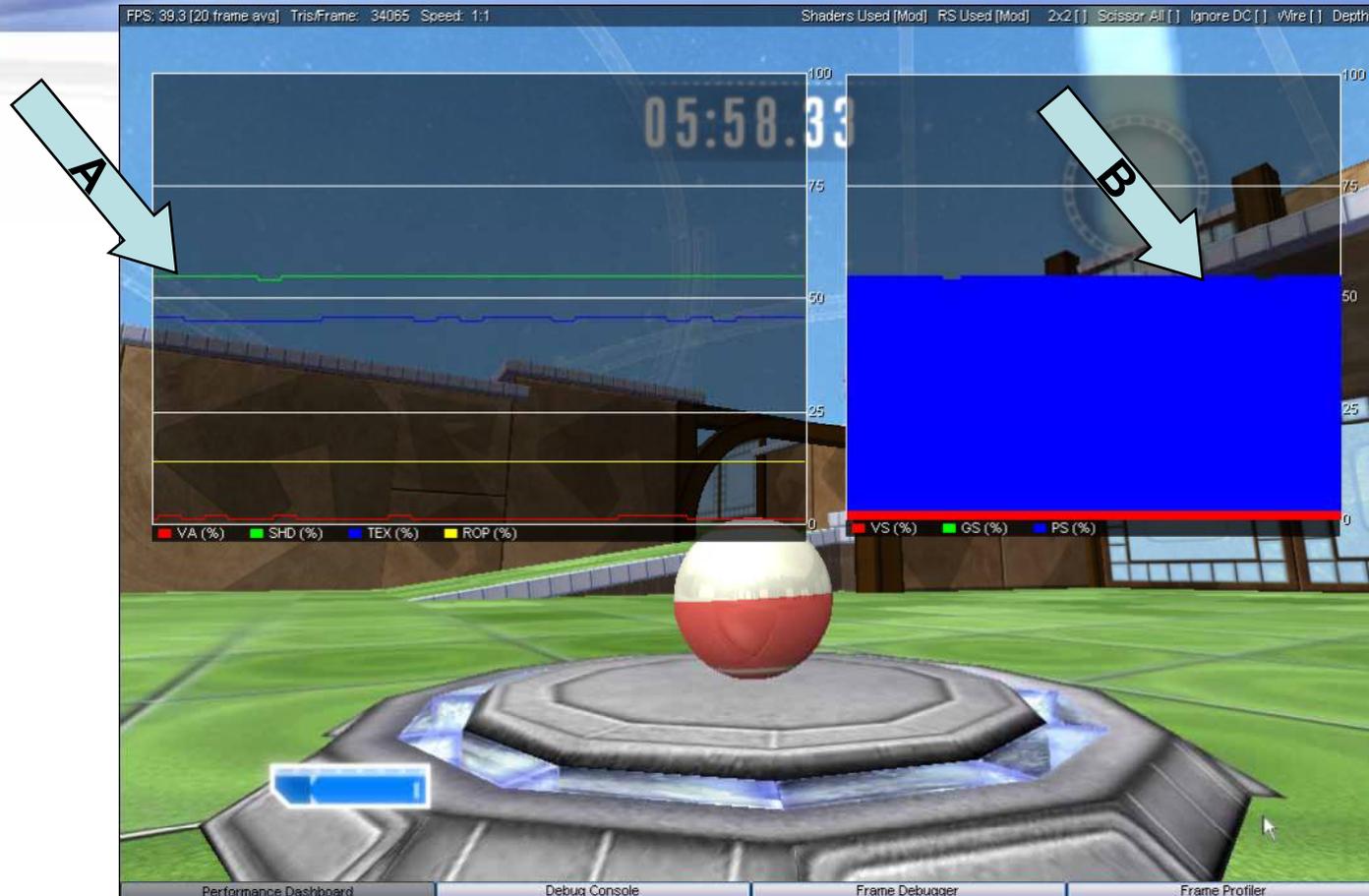
CPU vs GPU Performance Dashboard



GPU Busy % for the best FPS increase



GPU System Level Performance Dashboard



- VA(%) ~5%
- Shader(%) ~55%
- Texture(%) ~45%
- ROP(%) ~25%

- VS(%) ~5%
- GS(%) ~0%
- PS(%) ~55%

GPU Application Level Frame Profiler

FPS: 44.5 [20 frame avg] Tris/Frame: 33829 Speed: -- Shaders Used [Mod] RS Used [Mod] 2x2 [] Scissor All [] Ignore DC [] Wire [] Depth []

State buckets:

Vertex Shader Pixel Shader ROP RT Pixel Shader Constants D3D Perf Markers Alpha Enabled

Calls	Time v	Pixels	Perf Marker
14	4.125 ms	307695	0:
63	2.815 ms	1246752	0:
91	1.617 ms	57542	0:
7	1.515 ms	175600	0:
9	1.093 ms	146743	0:
4	0.828 ms	262144	0:
5	0.400 ms	36036	0:

Draw calls in selected State Bucket:

Ord	Prims	Time v	Pixels	Perf Marker
130	282	3.247 ms	257662	0: (null)
131	229	0.444 ms	36470	0: (null)
109	282	0.091 ms	4096	0: (null)
40	282	0.048 ms	2029	0: (null)
23	282	0.048 ms	2040	0: (null)
71	282	0.048 ms	1984	0: (null)
2	282	0.045 ms	1976	0: (null)

Unit Bottleneck Bars Export Data

Instruction Count Ratios

Legend: Frame (yellow), State Bucket (red), Draw Call (orange)

Legend: Frame (red), Bucket (green), Draw (blue)

Legend: VS (red), GS (green), PS (blue)

Draw Call 130/229

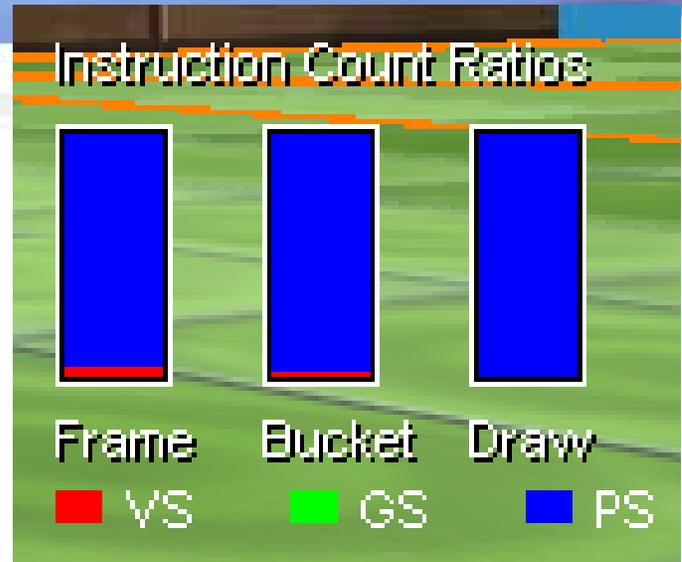
Prev Next Advanced... Settings...

Performance Dashboard | Debug Console | Frame Debugger | Frame Profiler

GPU Application Level Frame Profiler

State Bucket = 4.125 ms, 307,695 pixels

Draw Call = 3.247 ms, 257,662 pixels



GPU Micro Level

The screenshot shows a GPU performance analysis tool window titled "Marble Blast Ultra". The top status bar displays "FPS: 44.3 [20 frame avg] Tris/Frame: 36351 Speed: -:-". The right side of the status bar includes options: "Shaders Used [Mod] RS Used [Mod] 2x2 [] Scissor All [] Ignore DC [] Wire [] Depth []".

Below the status bar, there are two rows of performance metrics:

- Row 1: Draw Calls: 14 Time: 2.213 ms Pixels: 211542
- Row 2: Time: 1.537 ms Pixels: 165106

A horizontal bar chart is positioned below these metrics, with a red segment on the left and a yellow segment on the right.

The main interface is divided into four colored tabs: "Vertex Assembly" (dark green), "Vertex Shader" (dark red), "Pixel Shader" (green), and "Raster Operations" (dark blue). The "Pixel Shader" tab is currently selected.

On the left side, there is a "Textures" panel with a "RGB" dropdown menu. It lists three textures:

- Texture 1: Type: 2D [0x74cbec0], 512x512, A8R8G8B8, Mips: 1, Usage: D3DUSAGE_AUTOGENMIPMAP
- Texture 2: Type: 2D [0x74cc020], 256x256, X8R8G8B8, Mips: 1, Usage: D3DUSAGE_AUTOGENMIPMAP
- Texture 3: Type: 2D [0xad88120], 64x64, X8R8G8B8, Mips: 1, Usage: D3DUSAGE_AUTOGENMIPMAP

The main area of the "Pixel Shader" tab displays a shader code editor with the following code:

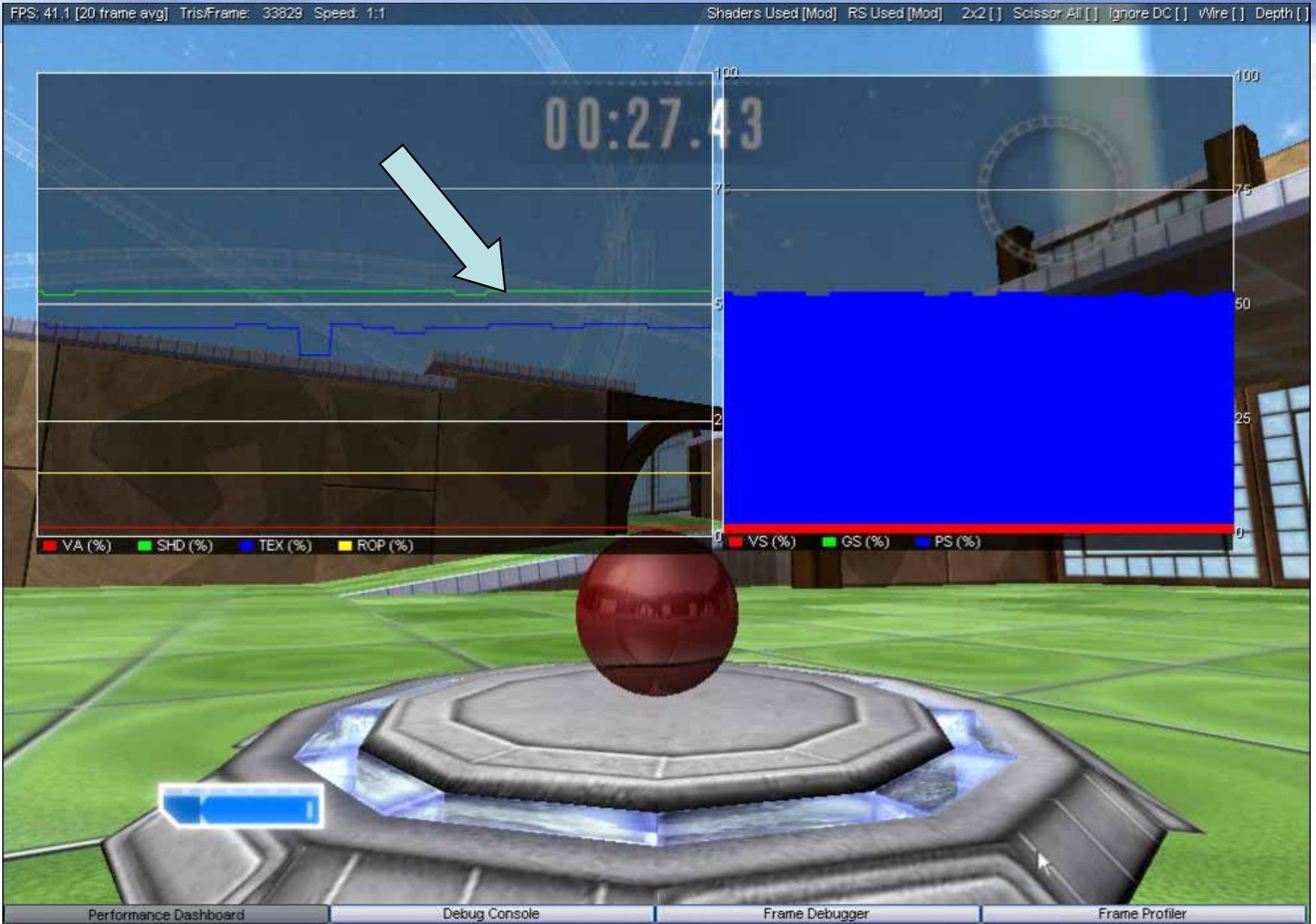
```
Unknown
// $specularPower c1 1
// $ambient c3 1
// $diffuseMap s0 1
// $bumpMap s1 1
// $noiseMap s2 1
//
ps_2_0
def c2, 2, -1, -0.015625, -0.5
def c4, 0.0158730168, 0.0078125, 0, 0
dcl v0
dcl v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, v48, v49, v50, v51, v52, v53, v54, v55, v56, v57, v58, v59, v60, v61, v62, v63, v64, v65, v66, v67, v68, v69, v70, v71, v72, v73, v74, v75, v76, v77, v78, v79, v80, v81, v82, v83, v84, v85, v86, v87, v88, v89, v90, v91, v92, v93, v94, v95, v96, v97, v98, v99
```

Below the code editor is a "Text to Find" search box and a scroll bar. Further down are sections for "Messages" and "Shader Constants". The "Shader Constants" section shows:

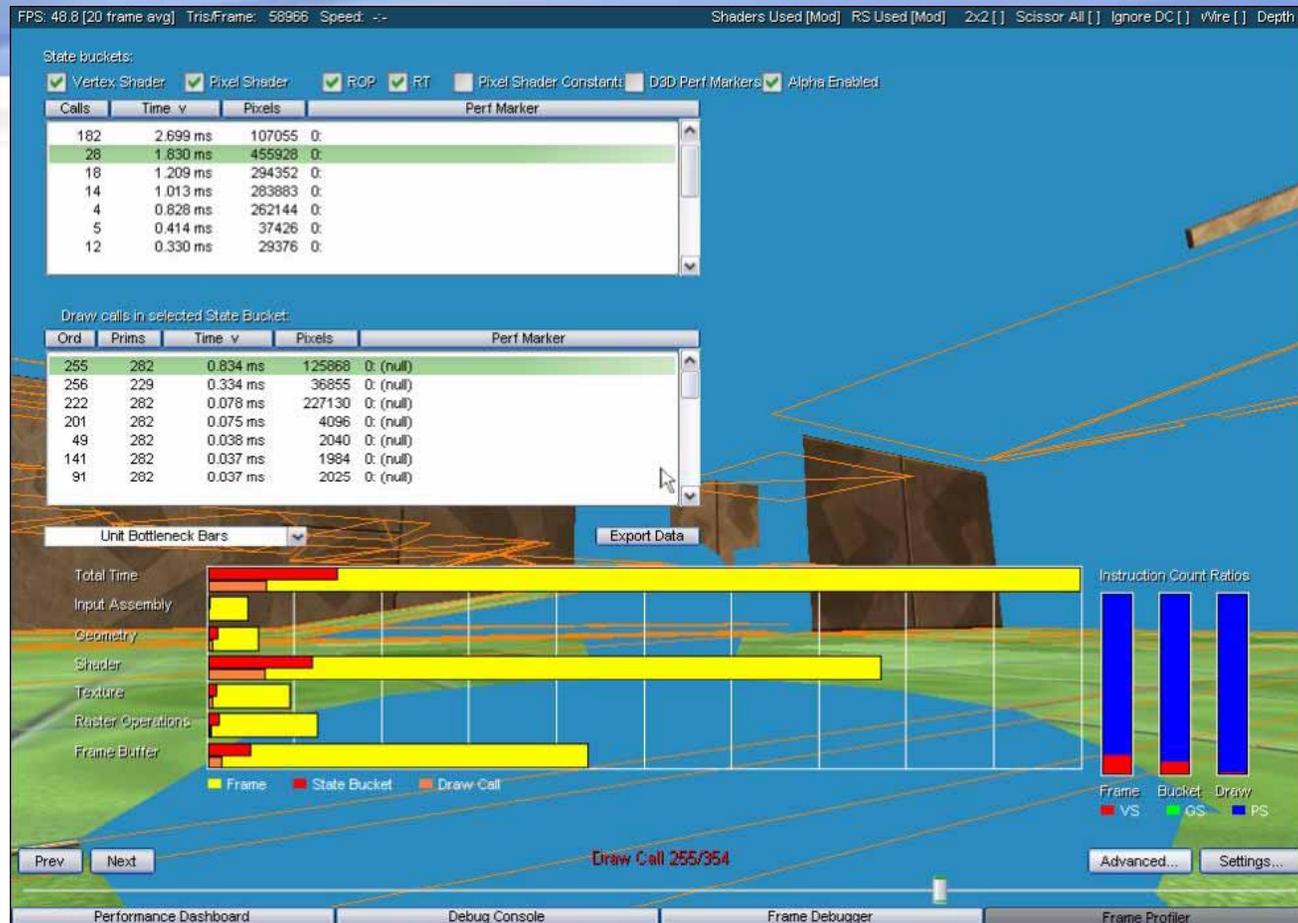
```
Float:
c0: (1.000000, 1.000000, 1.000000, 1.000000)
```

At the bottom of the window, there is a 3D view of a green field with a road. A red text overlay in the center reads "Draw Call 123/296". On the bottom left are "Prev" and "Next" buttons. On the bottom right are "Simple..." and "Settings..." buttons.

GPU System Level Performance Dashboard



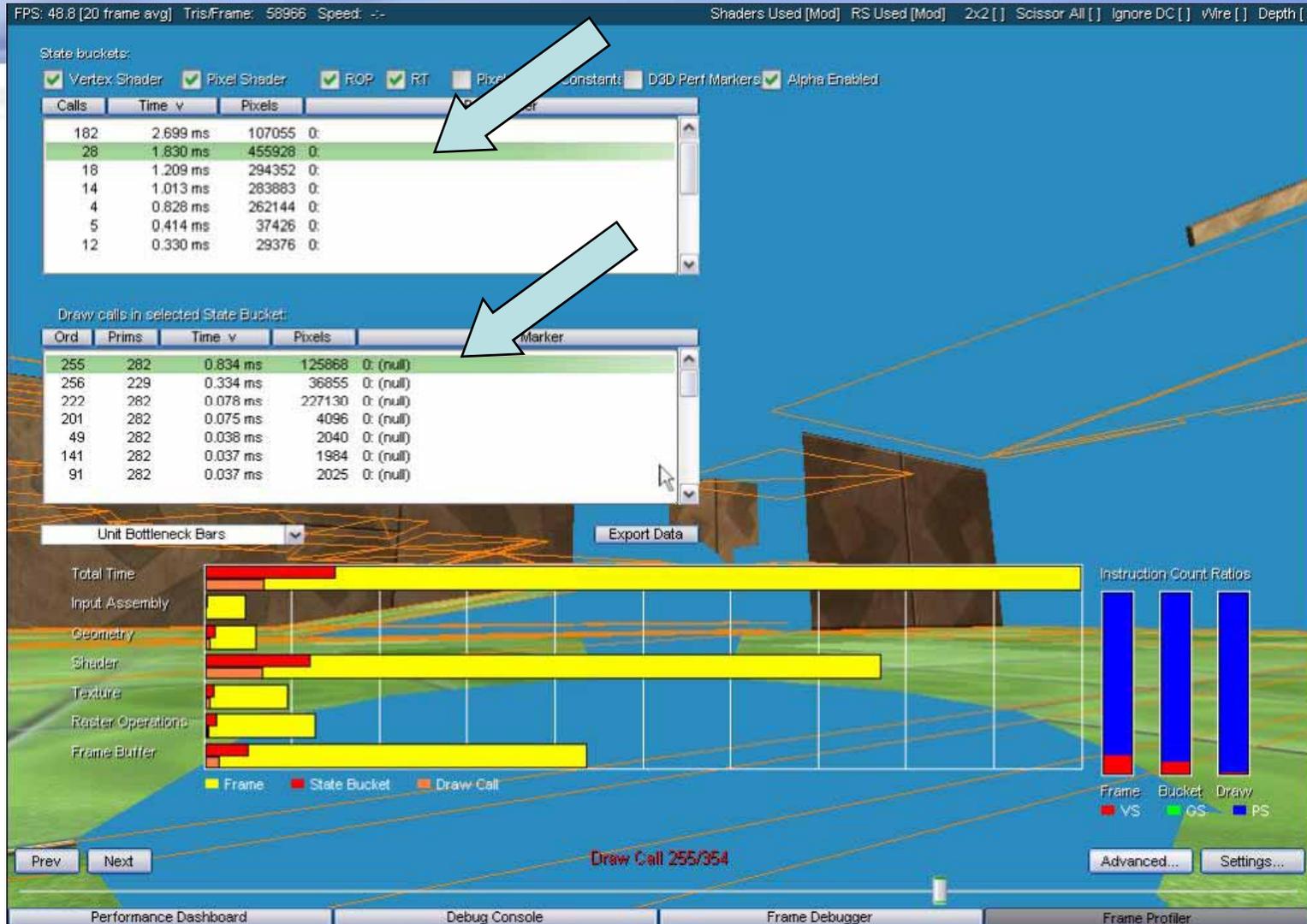
GPU Application Level



Problem: Inexpensive geometry, low number of draw calls, high number of expensive pixels.....

Solution: Render scene twice and disable color writes on the first draw!

GPU Application Level



State Bucket = 4.828 ms, 463,028 pixels

Draw Call = 0.834 ms, 125,868 pixels

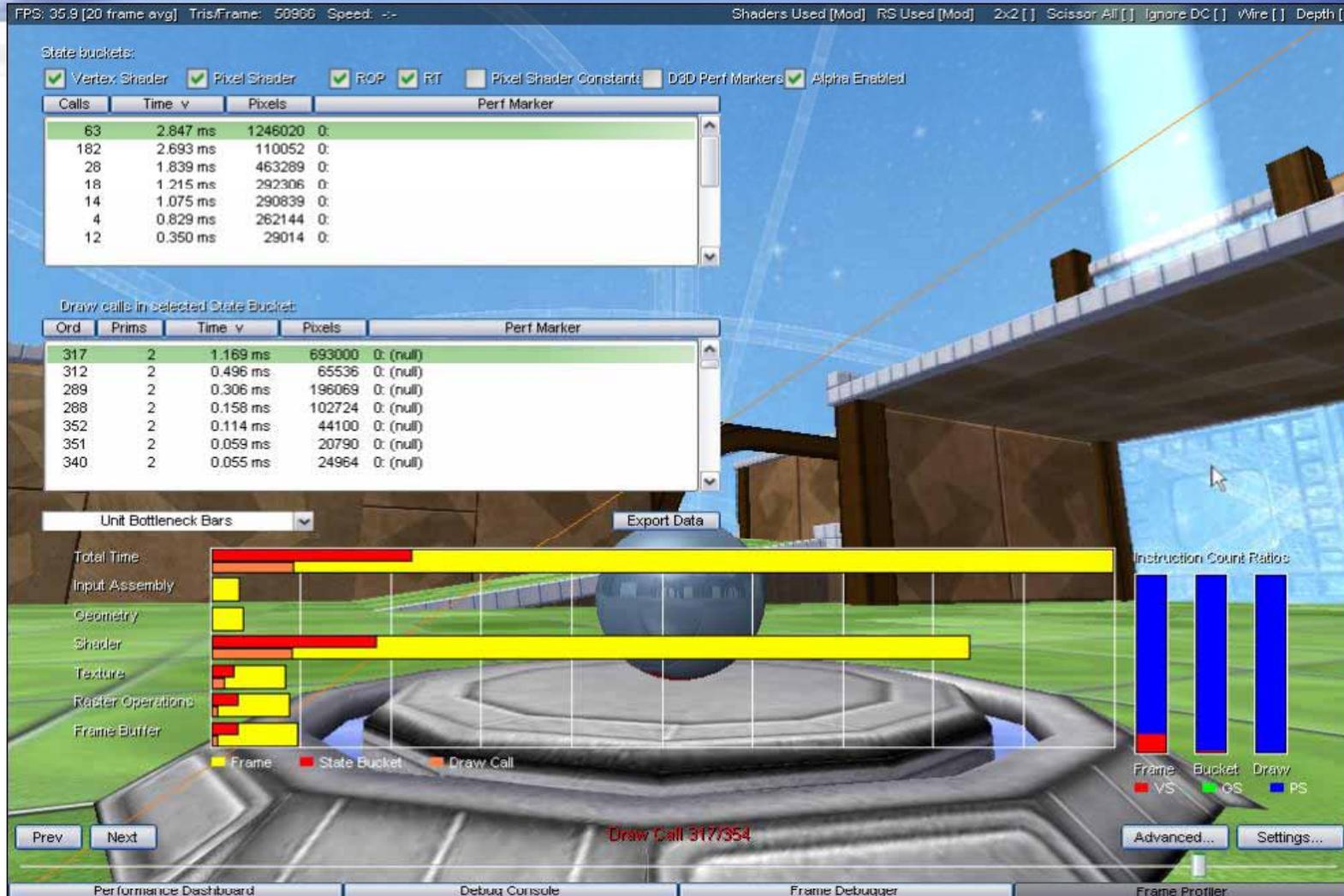
GPU System Level

Performance Dashboard



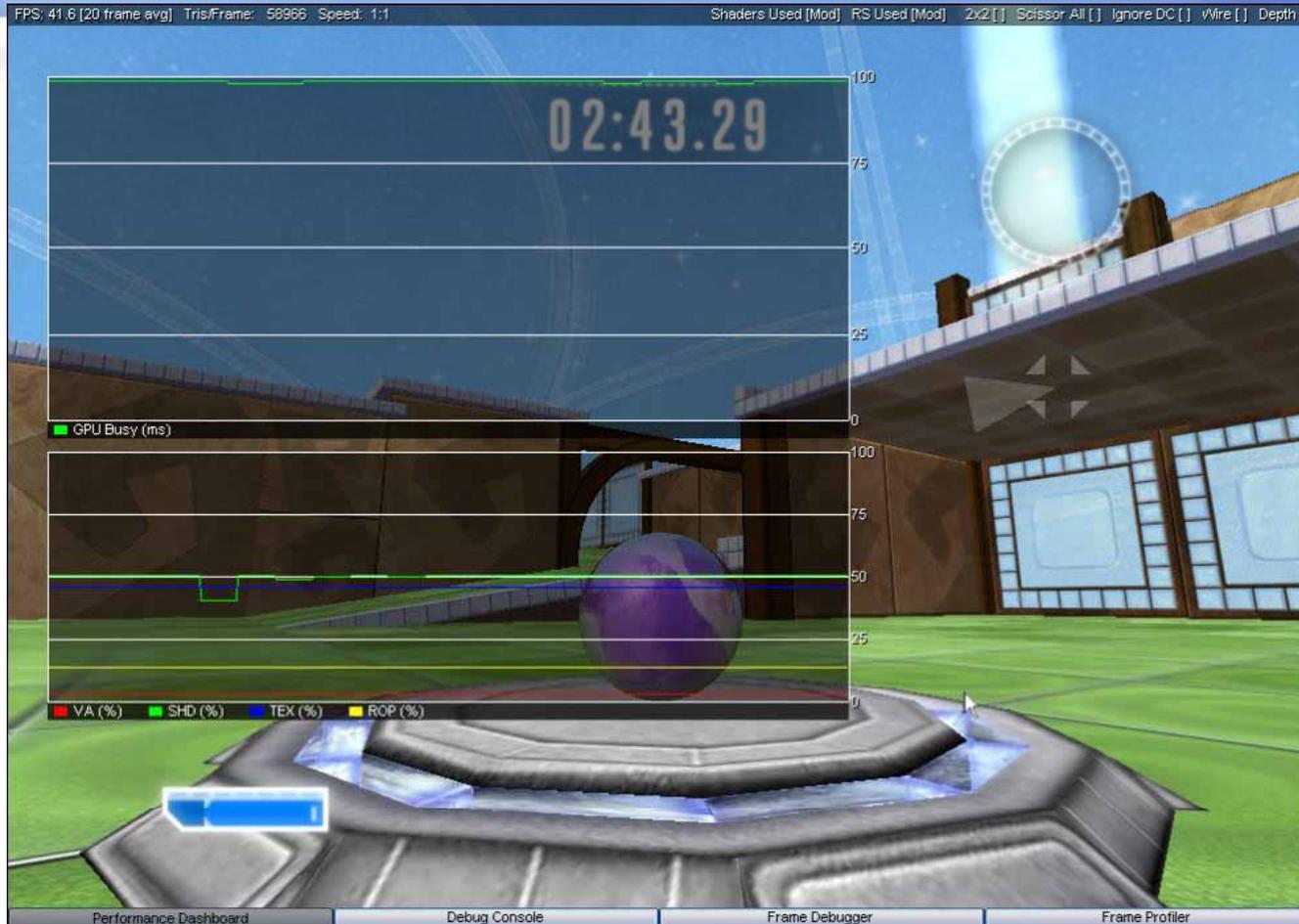
**Still very GPU Bound (A),
Still very shader and
texture bound (B)...**

GPU Application Level Frame Profiler



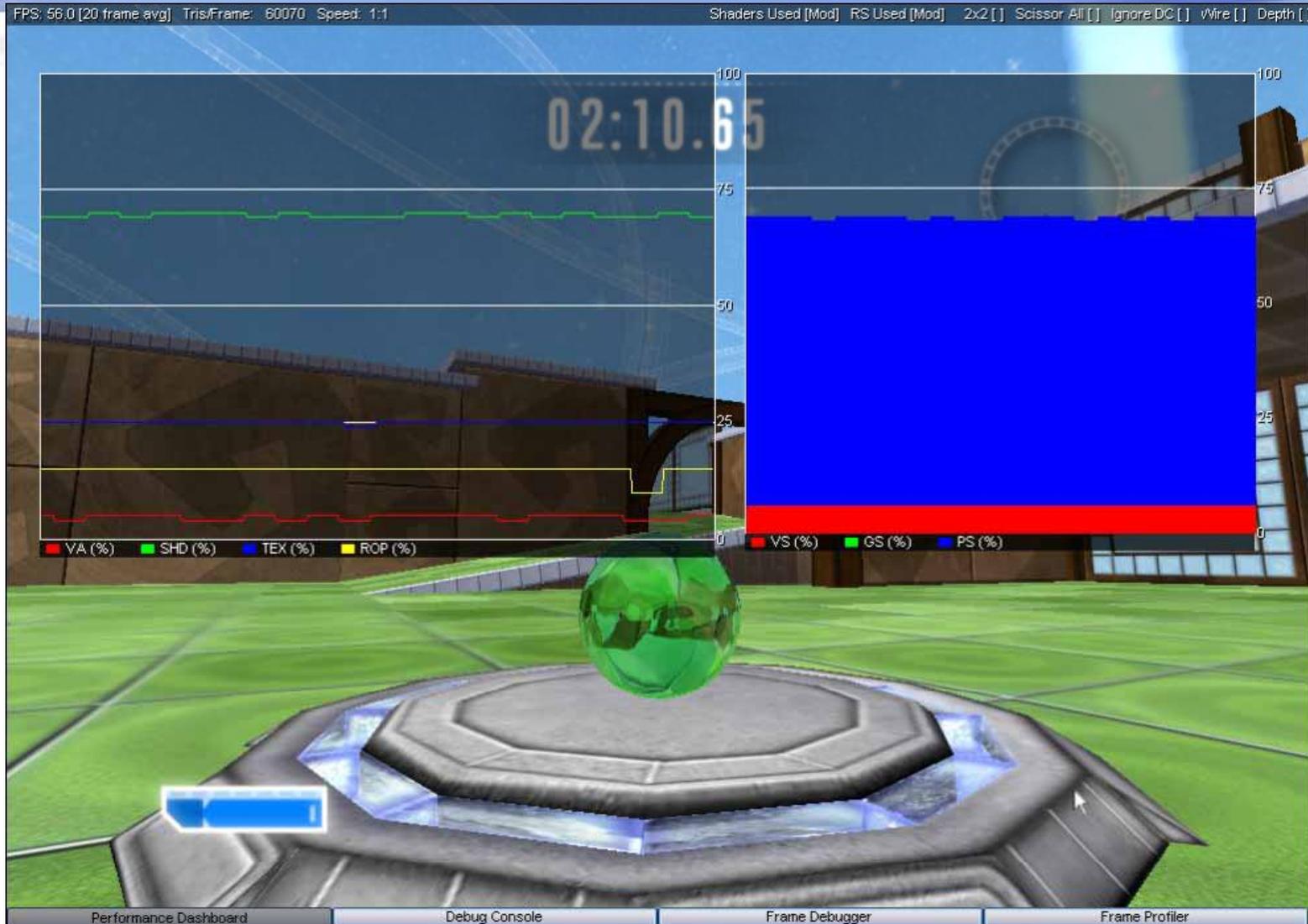
Now we are limited by a FFP call. By using the “scrubber” we determine that the call renders the glow buffer quad to the screen.

GPU System Level Performance Dashboard



Moving on to a texture optimization

GPU System Level Performance Dashboard



A StretchRect optimization reduces texture work, making pixel optimizations even more valuable.

GPU System Level Performance Dashboard



A driver optimization increases GPU Utilization.

GPU System Level

Performance Dashboard

What's next...

- Investigate more pixel optimizations – Scissor test confirms.
- Compressed texture usage – forcing 2x2 gives ~5 frame improvement.
- SLI Optimizations
- Investigate CPU optimizations, on the 8800 GPU utilization is only 25%

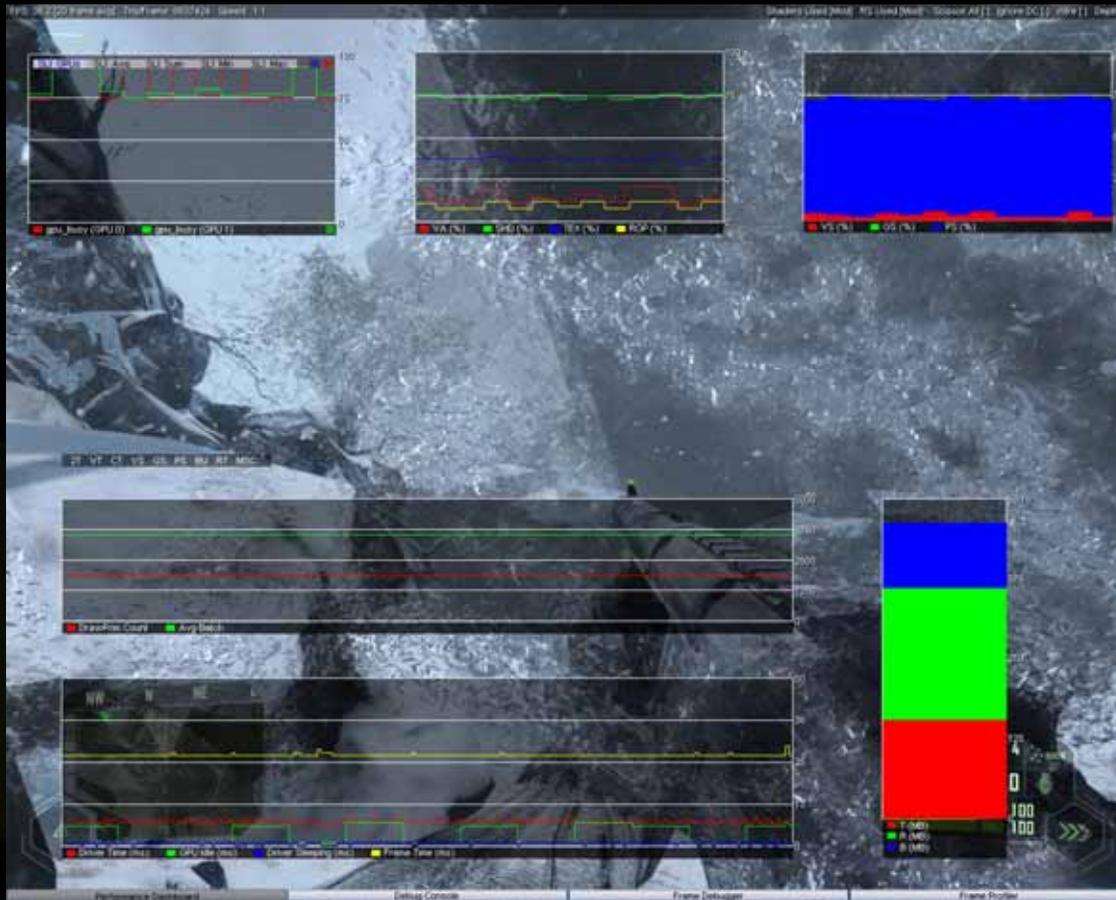
Thanks Eric!!!

Introducing PerfHUD 6.0!



- **Unified Driver on Vista: use any release driver!**
- **Comprehensive SLI Support**
 - Graphs for SLI specific data
 - Insight into SLI performance gotchas
- **Powerful new debugging features**
 - Texture visualization modes
 - API Call data mining and analysis
 - Shader visualization
- **Usability Features**
 - All new hot key support
 - Rich use of Direct3D PerfMarkers

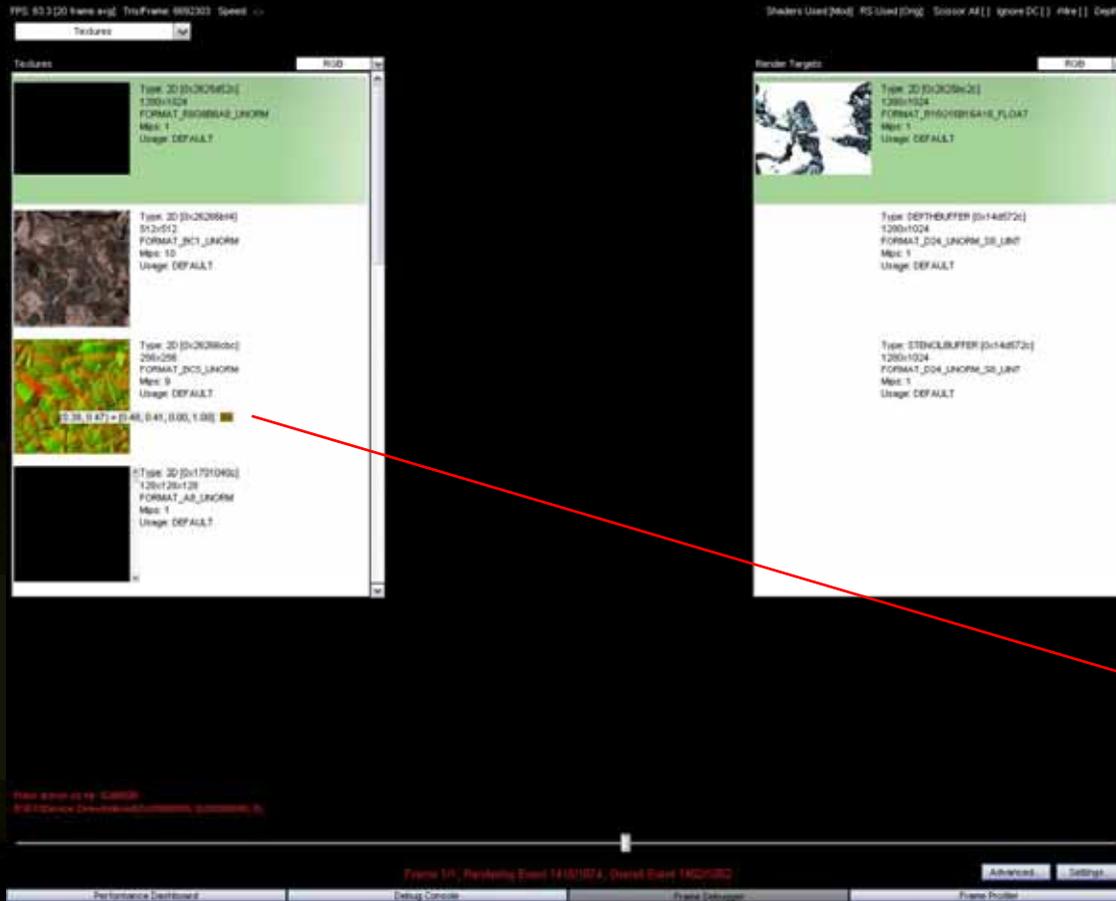
PerfHUD 6: Performance Dashboard



Graph GPU and driver data
Edit to suit your needs
SLI Graph for multi-GPU
API usage statistics

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Frame Debugger

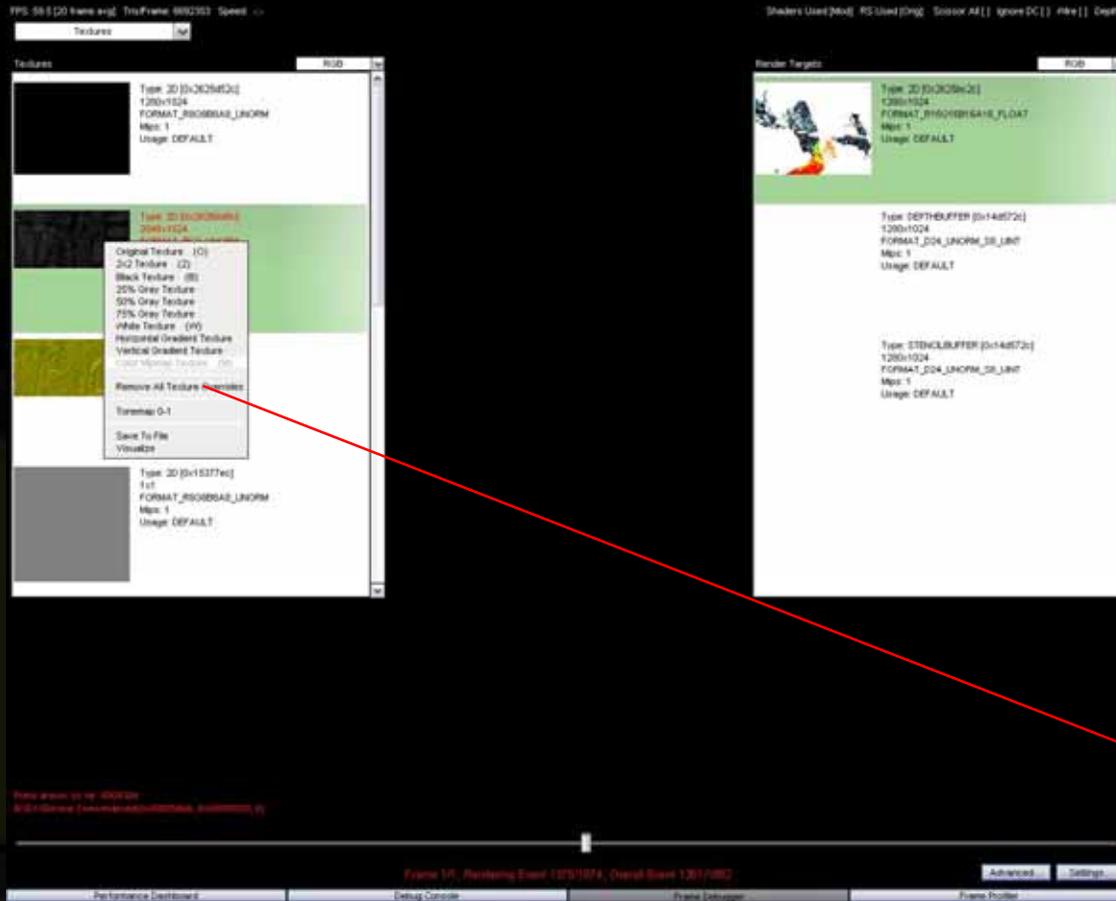


Scrub through scene
Visualize draw call info
Textures and RTs
Tooltips on buffers

[0.39, 0.47] - [0.40, 0.41, 0.00, 1.00]

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Frame Debugger



Texture analysis: substitute precomputed textures Controllable via Perf Markers

- Original Texture (O)
 - 2x2 Texture (Z)
 - Black Texture (B)
 - 25% Gray Texture
 - 50% Gray Texture
 - 75% Gray Texture
 - White Texture (W)
 - Horizontal Gradient Texture
 - Vertical Gradient Texture
 - Color Mipmap Texture (M)
- Remove All Texture Overrides
- Tonemap 0-1
- Save To File
- Visualize

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD6: Frame Debugger

Buffer Visualization



Visualize any buffer
full screen
2D/3D/Cube/Arrays
Pan/Zoom
Change mipmap level

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD6: Frame Debugger

API Call List



The screenshot displays the PerfHUD6 Frame Debugger interface. On the left, a list of API calls is shown, with a red arrow pointing to a specific event. On the right, a tooltip provides detailed information for the selected event, including its type, format, and usage.

API Call List (Left Panel):

```
Event 8273: ID3D10Device::DrawIndexed(0x00000000, 0x00000000, 0)
Event 8274: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456a0)
Event 8275: ID3D10Buffer::Unmap()
Event 8276: ID3D10Buffer::Unmap()
Event 8277: ID3D10Buffer::Unmap()
Event 8278: ID3D10Device::OMSetBlendState(0x0f6a96, -1 #GNAND, 0x00000001)
Event 8279: ID3D10Device::PSSetShader(0x0924980)
Event 8280: ID3D10Device::PSSetSampler(0x00000000, 0x00000001, 0x020456a0)
Event 8281: ID3D10Device::PSSetShaderResources(0x00000000, 0x00000001, 0x020456a0)
Event 8282: ID3D10Device::VSSetShader(0x0548280)
Event 8283: ID3D10Device::IASetVertexBuffers(0x00000000, 0x00000001, 0x020456a0, 0x020456b4, 0x020456c4)
Event 8284: ID3D10Device::IASetIndexBuffer(0x00000001, 0x00000001, 0x020456bc, 0x020456c0, 0x020456c4)
Event 8285: ID3D10Device::IASetIndexBuffer(0x00000001, 0x00000001, 0x020456bc, 0x020456c0, 0x020456c4)
Event 8286: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456bc)
Event 8287: ID3D10Buffer::Unmap()
Event 8288: ID3D10Device::VSSetConstantBuffer(0x00000000, 0x00000001, 0x020456b0)
Event 8289: ID3D10Device::VSSetConstantBuffer(0x00000000, 0x00000001, 0x020456b0)
Event 8290: ID3D10Device::PSSetConstantBuffer(0x00000000, 0x00000001, 0x020456b8)
Event 8291: ID3D10Device::DrawIndexed(0x000)
Event 8292: ID3D10Buffer::Map(0x4, 0x00000000) Elapsed Time: 71.28 ticks
Event 8293: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456bc)
Event 8294: ID3D10Buffer::Unmap()
Event 8295: ID3D10Buffer::Unmap()
Event 8296: ID3D10Device::OMSetBlendState(0x45ed7348, -1 #GNAND, 0x00000001)
Event 8297: ID3D10Device::PSSetShader(0x0924980)
Event 8298: ID3D10Device::PSSetSampler(0x00000000, 0x00000001, 0x020456a0)
Event 8299: ID3D10Device::PSSetShaderResources(0x00000000, 0x00000001, 0x020456a0)
Event 8300: ID3D10Device::VSSetShader(0x45ed6ea0)
Event 8301: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456b0)
Event 8302: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456c0)
Event 8303: ID3D10Buffer::Unmap()
Event 8304: ID3D10Buffer::Unmap()
Event 8305: ID3D10Device::DrawIndexed(0x00000276, 0x00000816, 0)
Event 8306: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456b0)
Event 8307: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456c4)
Event 8308: ID3D10Buffer::Unmap()
Event 8309: ID3D10Buffer::Unmap()
Event 8310: ID3D10Device::OMSetBlendState(0x0f6a96, -1 #GNAND, 0x00000001)
Event 8311: ID3D10Device::PSSetShader(0x45ed6c0)
Event 8312: ID3D10Buffer::Map(0x4, 0x00000000, 0x020456b0)
Event 8313: ID3D10Device::PSSetShaderResources(0x00000001, 0x00000001, 0x020456fc)
Event 8314: ID3D10Device::PSSetShaderResources(0x00000002, 0x00000001, 0x02045700)
Event 8315: ID3D10Device::PSSetSampler(0x00000000, 0x00000001, 0x02045704)
```

Tooltip Details (Right Panel):

Type: DEPTHBUFFER [0x14872c]
12001004
FORMAT_D3D10_UNORM_32_UINT
Map: 1
Usage: DEFAULT

Type: INDEXBUFFER [0x14872c]
12001004
FORMAT_D3D10_UNORM_32_UINT
Map: 1
Usage: DEFAULT

Type: INDEXBUFFER [0x14872c]
12001004
FORMAT_D3D10_UNORM_32_UINT
Map: 1
Usage: DEFAULT

Based on a frame capture
See frame events
including parameters
Tooltips for details
Connected to scrubber

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Frame Debugger

Draw Call Dependencies



Dependencies

Producers

- Event 5439: @0010Device: ClearRenderTargetView(0x0f5ee396, 0, 0, 0, 0, 0)
- Event 5450: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5464: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5470: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5478: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5490: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5520: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5521: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5534: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5551: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5560: @0010Device: DrawIndexed(0x00002212, 0x00000000, 0)
- Event 5571: @0010Device: DrawIndexed(0x00002212, 0x00000000, 0)
- Event 5597: @0010Device: Draw(0x00000004, 0x0000000c)
- Event 5601: @0010Device: ClearRenderTargetView(0x0f5ee340, 0, 0, 0, 0, 0)
- Event 5610: @0010Device: Draw(0x00000004, 0x0000000c)

Consumers

- Event 13270: @0010Device: Draw(0x00000004, 0x00000244)
- Event 13607: @0010Device: Draw(0x00000004, 0x00000248)

Render Targets

(0.72, 0.88) = (2.06, 2.22, 2.44, 0.63)

Frame 1/1, Rendering Event 13571/1375, Drawn Event 1361/1383

Performance Dashboard | Debug Console | Frame Debugger | Frame Profiler

Show **producers** & **consumers** dependencies for each call
These can hurt single GPU and SLI performance

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Adv Frame Debugger

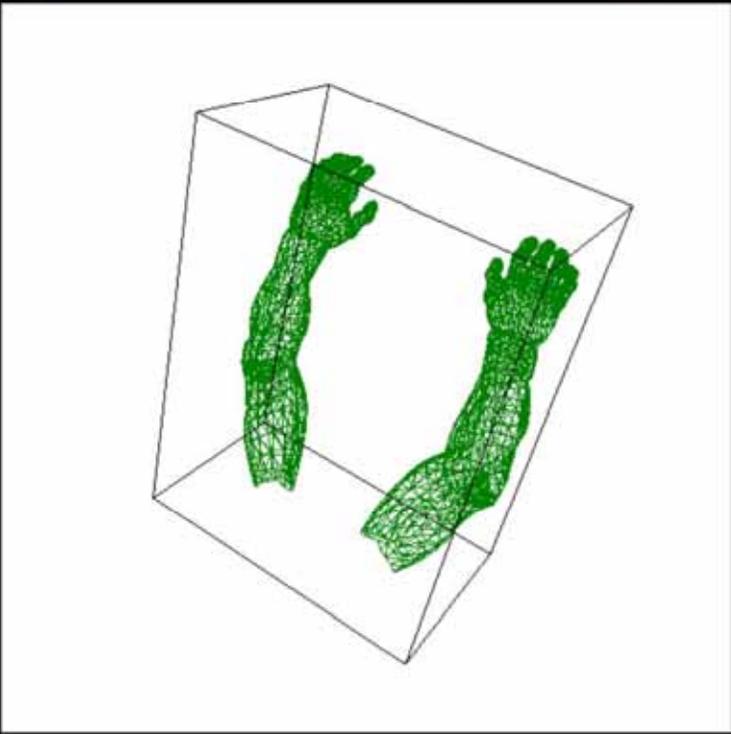
Vertex Assembly



FPS: 28.4 (20 frame avg) Tris/Frame: 692353 Speed: <> Shaders Used (Max): RS Used (Orig): Shader All[] Ignore DC[] Attr[] Depth[]

Vertex Assembly | Vertex Shader | Geometry Shader | Pixel Shader | Raster Operations

Wireframe



Index / Vertex Buffer

Cell Type: DrawIndexed
IndexCount: 20712
StartIndexLocation: 0
BaseVertexLocation: 0
Topology: TRIANGLELIST

Index Buffer

Pointer: 0x2617884, Format: R16_UInt, Offset: 0

Vertex Buffers

0) Pointer: 0x261788c, Stride: 24, Offset: 0
1) Pointer: 0x262568c, Stride: 16, Offset: 87104
2) Pointer: 0x26179c4, Stride: 20, Offset: 0
3) Pointer: 0x175e7ac, Stride: 20, Offset: 94300

Pixel Layout

Rounding: bot
Min: (0.210641, 0.000000, 0.000000)
Max: (-0.210641, -0.022586, -0.378402)
Color: Red[] = (1, 1, 1, 1023, 1023)

Frame: 10, Rendering Error: 18571074, Shared Error: 12617882

Performance Dashboard | Debug Console | Frame Debugger | Frame Profiler

Geometry Preview
Vertex and index buffer setup

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Adv Frame Debugger

Vertex, Geometry and Pixel Shaders



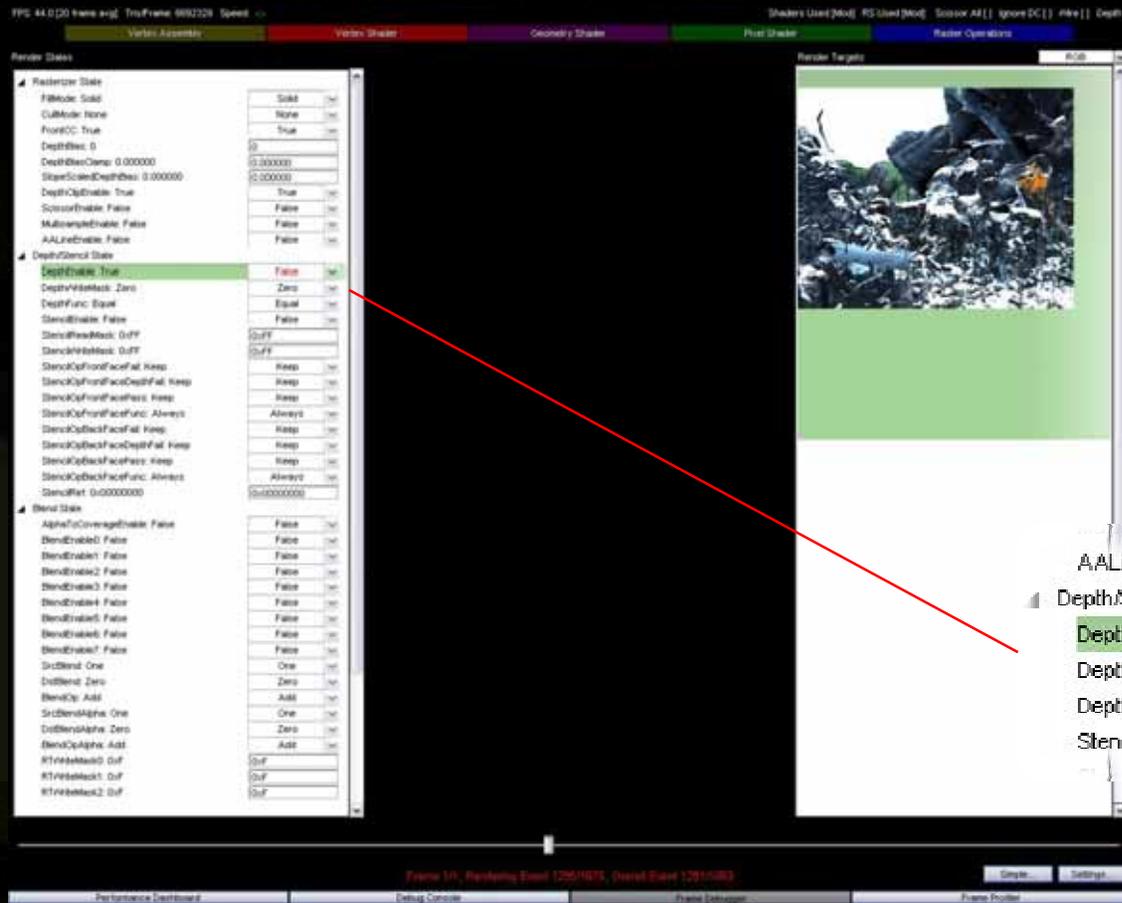
The screenshot displays the PerfHUD 6 interface with several panels:

- Textures:** A list of textures including 'Type 2D [0x025450] 128x128', 'Type 2D [0x026960] 2048x1024', 'Type 2D [0x026960] 1024x512', and 'Type 2D [0x15377c] 1x1'.
- Pixel Shader Samplers:** A list of samplers for 'PS Sampler 0' with various settings like 'Filter: Anisotropic', 'AddressU: Wrap', 'AddressV: VWrap', 'AddressW: Wrap', 'MipLODBias: 0.000000', and 'MaxAnisotropy: 8'.
- Shader Code:** A window showing HLSL code generated by Microsoft (R) HLSL Shader Compiler. It includes buffer definitions for 'PER_FRAME' and 'PER_MATERIAL', and constant buffer definitions for 'PER_FRAME' and 'PER_MATERIAL'.
- Messages:** A panel for displaying messages.
- Shader Constants:** A panel for displaying constant buffer values.

Edit & Continue Shaders
Visualize input textures
Constants
Sampler overrides

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Adv Frame Debugger



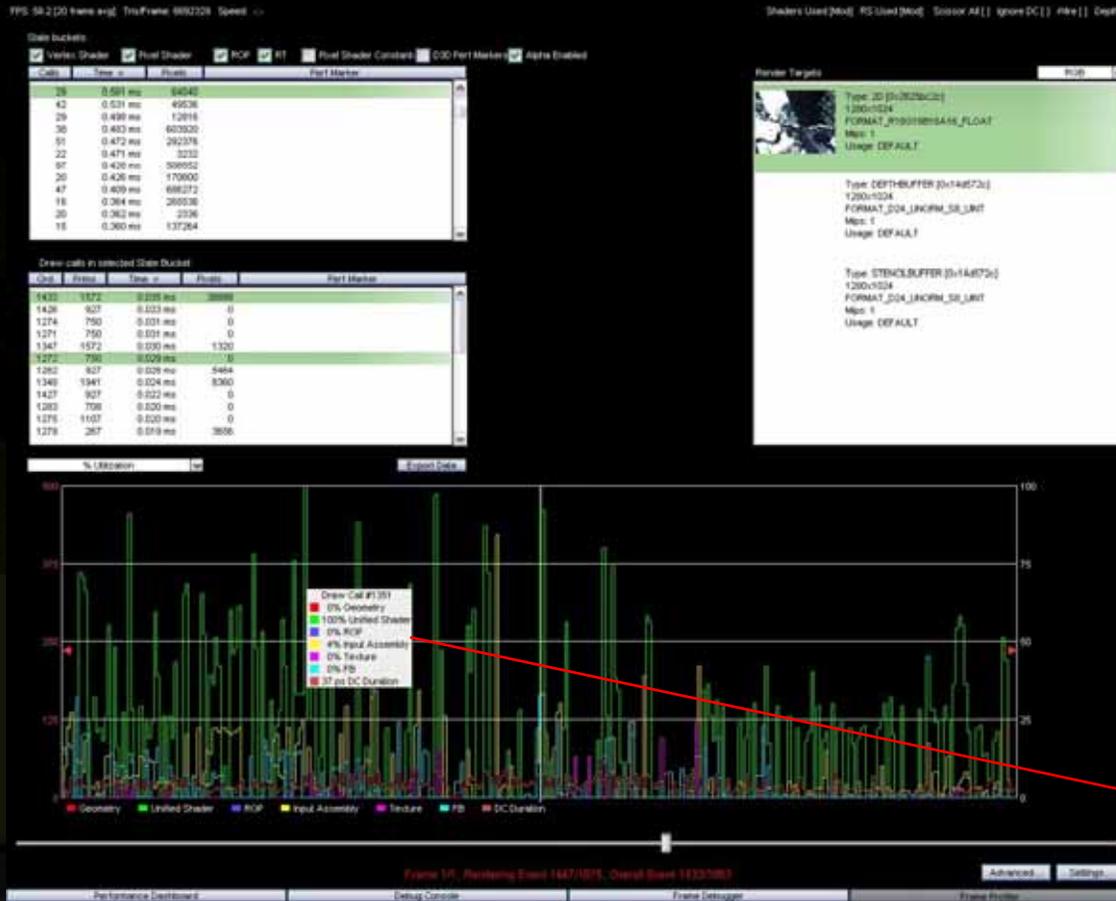
Display and modify all render state settings
Render targets displayed

AALineEnable: False	False	▼
Depth/Stencil State		
DepthEnable: True	False	▼
DepthWriteMask: Zero	Zero	▼
DepthFunc: Equal	Equal	▼
StencilEnable: False	False	▼

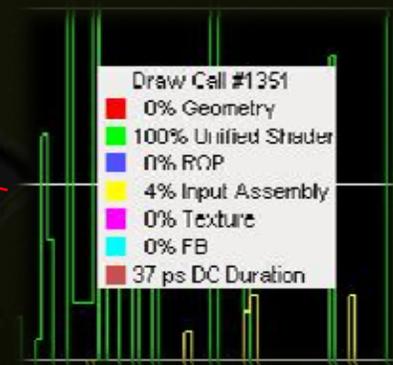
Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Frame Profiler

One button bottleneck determination



All draw calls profiled
Draw calls grouped by State Buckets: multiply performance optimizations
Multiple result graphs



Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6: Adv Frame Profiler



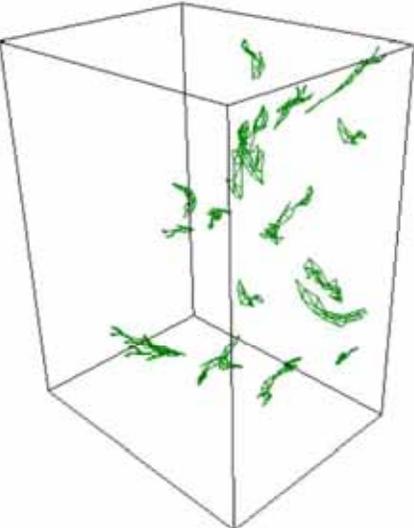
FPS: 24.8 [20 frame avg] Tris/Frame: 693229 Speed: <> Shaders Used [246] RS Used [246] Scissor AA[] Ignore DC[] #Tex[] Depth[]

Draw Calls: 1 Time: 24.305 ms Points: 180240

Total: 0.043 ms Points: 0

Vertex Assembly Vertex Shader Geometry Shader Pixel Shader Pixel Operations

Viewport



Index / Vertex Buffer

Call type: DrawIndexed
IndexCount: 1572
IndexBufferLocation: 1404
BaseVertexLocation: 0
Topology: TRIANGLELIST

Index Buffer

Pointer: 0x260307ac, Format: R16_UINT, Offset: 0

Vertex Buffers

0) Pointer: 0x17099c1c, Stride: 24, Offset: 26180
1) Pointer: 0x2618632c, Stride: 16, Offset: 137588

Input Layout

Rounding box:
Min: (-2.462505, 1.360373, 2.462471)
Max: (1.021595, -2.766482, -0.227261)
Scissor Rect[0] = (1, 1, 1520, 1520)

Frame: 1/1, Rendering Speed: 1447/1975, Drawn: 646/1423/5803

Performance Dashboard Debug Console Frame Debugger Frame Profiler

Simple Settings

Same advanced features now
in the profiling context

Crysis used with permission from Crytek. © Crytek GmbH. All Rights Reserved. Crysis and CryENGINE are trademarks or registered trademarks of Crytek GmbH in the U.S and/or other countries.

PerfHUD 6



A taste of things to come!

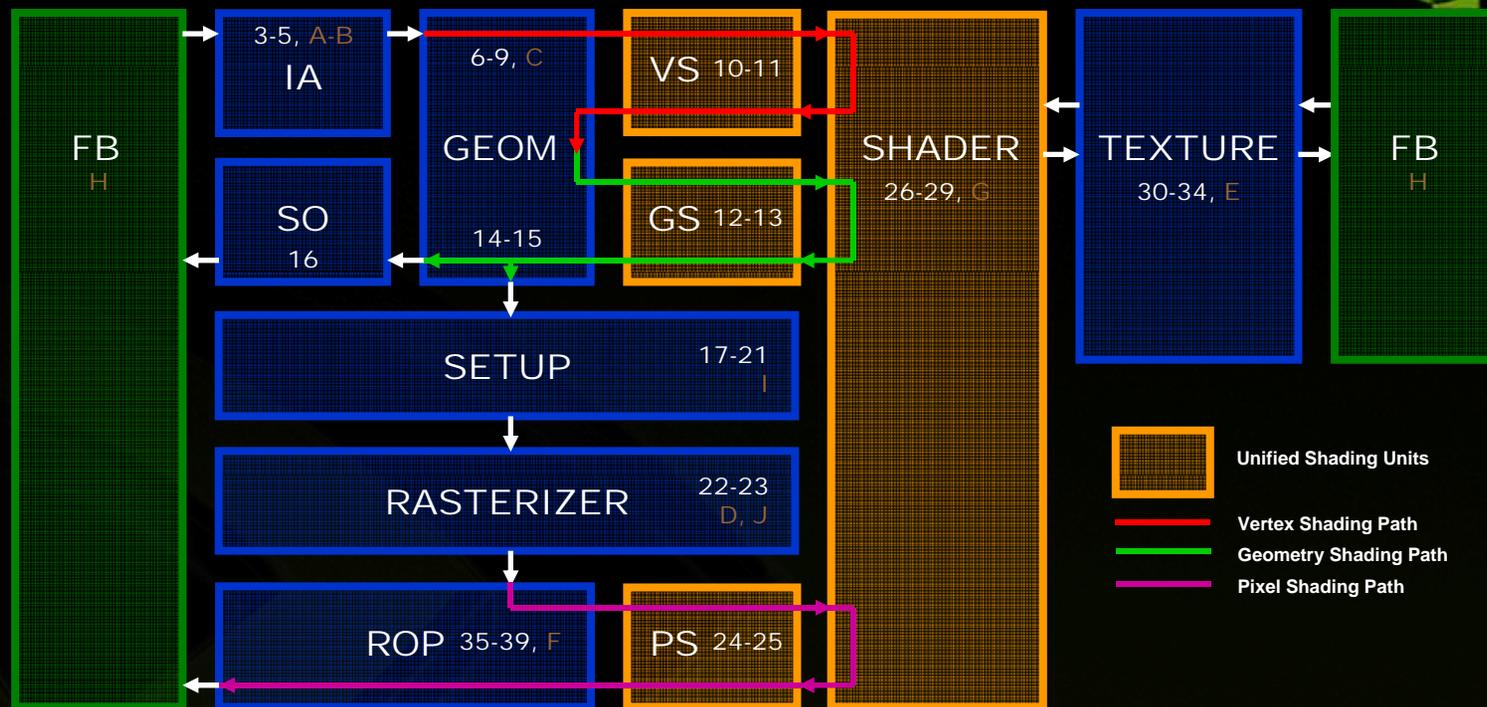
- **Better control via PerfMarkers: add them now!**
- **API time graph**
- **More performance hints: VSync on, windowed mode, event queries, not all render targets used, VBs not managed, etc.**
- **Subtotals in Frame Profiler**
- **Break (`_int 3`) on draw call**
- **32bit apps on 64bit OSs**

PerfKit: Features



- **PerfSDK**
 - Real time performance information in your game
 - Driver data, GPU counters, etc.
 - Simplified Experiments for easy bottleneck analysis
 - Simple API, code samples and helper classes
- **GLExpert**
 - Detailed feedback on pipeline setup
 - SLI performance feedback
 - Warnings for software fallback
 - VBO/FBO performance information
- **Microsoft PIX for Windows plugin**
 - GPU & driver counters alongside PIX data

Simplified G80+ Counter Diagram



1. gpu_idle (Global)
2. gpu_busy (Global)
3. input_assembler_busy
4. input_assembler_waits_for_fb
5. vertex_attribute_count
6. geom_busy
7. geom_waits_for_shader
8. geom_vertex_in_count
9. geom_primitive_in_count
10. vertex_shader_busy
11. vertex_shader_instruction_rate
12. geometry_shader_busy
13. geometry_shader_instruction_rate

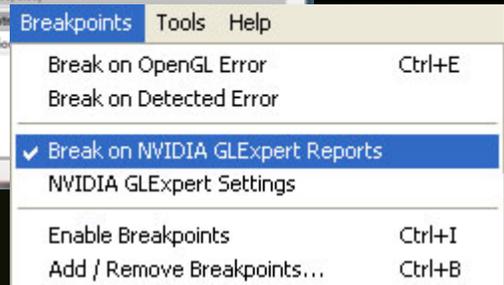
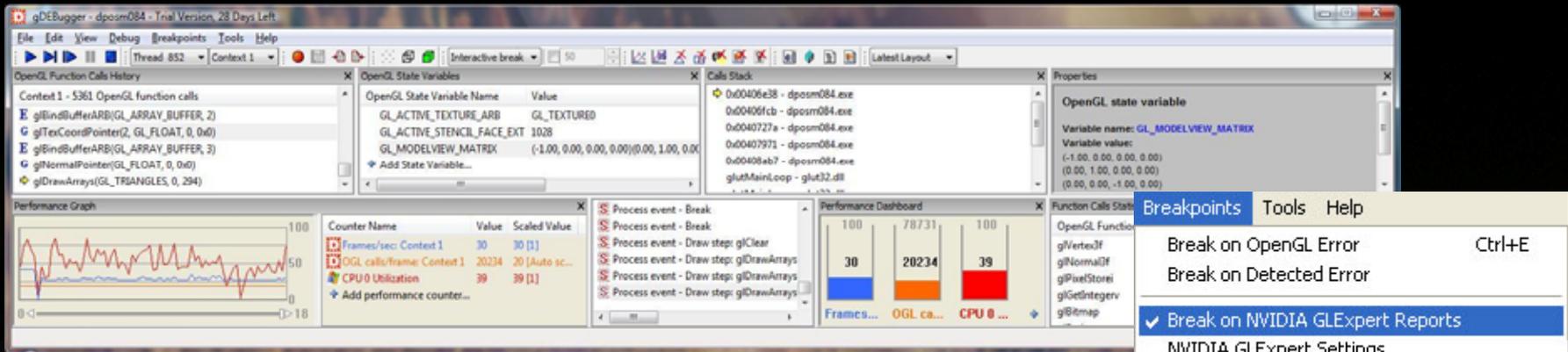
14. geom_vertex_out_count
15. geom_primitive_out_count
16. stream_out_busy
17. setup_primitive_culled_count
18. setup_primitive_count
19. setup_triangle_count
20. setup_point_count
21. setup_line_count
22. shaded_pixel_count
23. rasterizer_pixels_killed_zcull_count
24. pixel_shader_busy
25. pixel_shader_instruction_rate
26. shader_busy

27. shader_waits_for_texture
28. shader_waits_for_geom
29. shader_waits_for_rop
30. texture_busy
31. texture_waits_for_fb
32. texture_waits_for_shader
33. texture_sample_base_level_rate
34. texture_sample_average_level
35. rop_busy
36. rop_waits_for_fb
37. rop_waits_for_shader
38. rop_pixels_killed_earlyz_count
39. rop_pixels_killed_latez_count

- A. 2D Bottleneck and SOL
- B. IDX Bottleneck and SOL
- C. GEOM Bottleneck and SOL
- D. ZCULL Bottleneck and SOL
- E. TEX Bottleneck and SOL

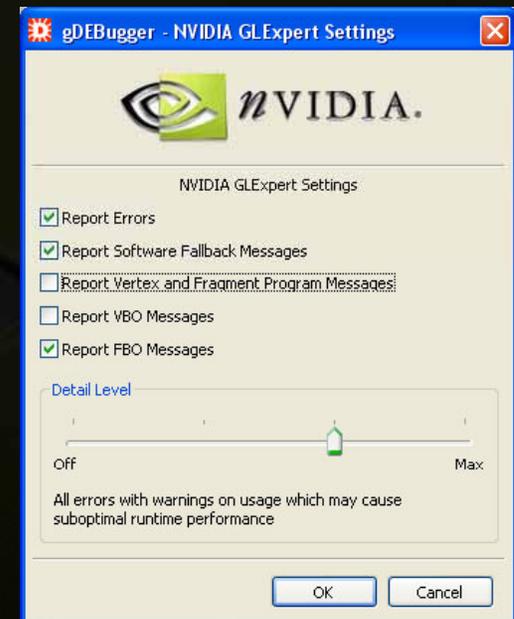
- F. ROP Bottleneck and SOL
- G. SHD Bottleneck and SOL
- H. FB Bottleneck and SOL
- I. Primitive Setup Bottleneck and SOL
- J. Rasterization Bottleneck and SOL

Graphic Remedy's gDEBugger



- OpenGL and OpenGL ES Debugger and Profiler
- NVIDIA PerfKit and GLEExpert integrated
- Shorten development time
- Improve application quality
- Optimize performance
- Texture/buffer viewer
- Windows XP & Vista, Linux too!
- Discounted academic licenses available

<http://www.gremedy.com>



Graphic Remedy's gDEBugger



The screenshot displays the gDEBugger application interface, which is used for debugging graphics drivers. It is divided into several main sections:

- Texture List:** A list of textures loaded in the application, including their names and addresses. For example, "2D Tex 8 (set 3, bound 20)".
- Properties of Texture #13:** A detailed view of a selected texture, showing its dimensions (Width: 512 px, Height: 512 px), depth, and format (GL_RGBA8).
- Texture Parameters:** A list of parameters for the selected texture, such as "GL_TEXTURE_MIN_FILTER" and "GL_TEXTURE_MAG_FILTER".
- Static Buffer Properties:** Information about the buffer type (Depth Buffer) and its dimensions (Width: 512 px, Height: 512 px).
- Data View:** A table showing the raw data of the texture or buffer. The top part shows a 10x10 grid of values for a texture, and the bottom part shows a list of values for a buffer. The buffer values are organized into columns labeled from 000 to 017.
- Image View:** A window showing the visual representation of the texture or buffer data. The top part shows a textured surface, and the bottom part shows a 3D model of a teapot with a green highlight on its base.

ShaderPerf 2.0

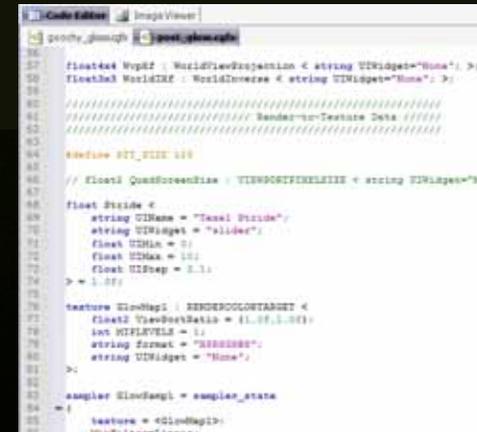
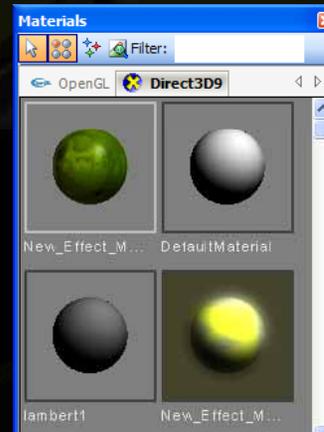


- **Determine shader performance**
 - Compare with shader cycle budgets
 - Test optimization opportunities
- **Automated regression analysis**
- **Integrated in FX Composer**
 - Interactive GUI
 - Artists/TDs code expensive shaders
 - Achieve optimum performance
- **GeForce 8 series performance data**
- **Beta 2.0 available now!**

FX Composer

Shader Authoring Made Easy!

- DirectX 10 backend
- Shader Debugger
- GeForce 8 Series Shader Performance
- Full-featured code editor
- Shader creation wizard with templates
- Integration with online Shader Library
- Materials panel to organize materials



FX Composer

HLSL10 Shader Debugging!



The screenshot displays the NVIDIA FX Composer 2.5 interface. The main window is titled "SimplePhong... - Debugging" and shows the following HLSL code:

```
189 colorOutput += colorDiffuse;
190
191 ///////////////////////////////////////////////////////////////////
192 // Enhance the specular effect
193 if(RdotV > 0.4)
194 {
195     RdotV = (RdotV - 0.4) * 1.1 + 0.4;
196 }
197
198 ///////////////////////////////////////////////////////////////////
199 // Compute and add the specular color
200 float RdotVPowN = pow(RdotV, SurfFloatShininess);
201 float4 colorSpecular = surfSpecular * (RdotVPowN * Light
202 colorOutput += colorSpecular;
203
204 ///////////////////////////////////////////////////////////////////
205 // Compute and add the transparency factors
206 colorOutput += surfReflective*SurfFloatReflectivity;
207 colorOutput.a = surfTransparent.a + surfTransparent.a*
208
209 return colorOutput;
210
211
212 ///////////////////////////////////////////////////////////////////
```

The debugger window shows the following table of variables:

Name	Value	Type
surfReflective		float4
surfTransparent		float4
colorEmission		float4
colorOutput		float4
normal		float3
x		float
y		float
z		float
eyeVec		float3
colorAmbient		float4
lightVec		float3
Material		float

The render view shows a 3D model of a character with a rainbow-colored top and a plaid skirt. The status bar indicates the pixel value at the cursor: Pixel: D=(293,Y=285) Value: [R=0.90065, G=0.20819, B=0.38143, A=1] Show As: RGBA

Questions?



- Online: downloads, videos, etc.

<http://developer.nvidia.com/PerfKit>

<http://developer.nvidia.com/PerfHUD>

<http://developer.nvidia.com/ShaderPerf>

<http://developer.nvidia.com/FXComposer>

Feedback and Support: <http://developer.nvidia.com/forums>