NVIDIA®

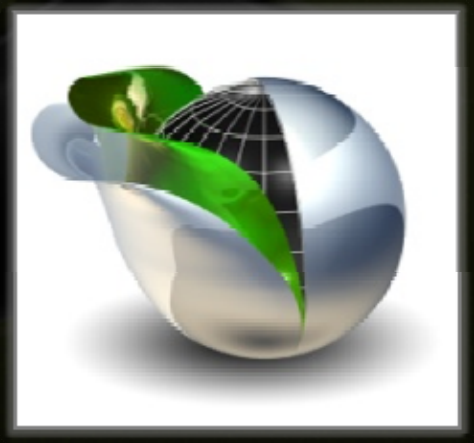
Performance Tools

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 gDEBugger
- ShaderPerf
- FX Composer 2.5: Shader Debugger



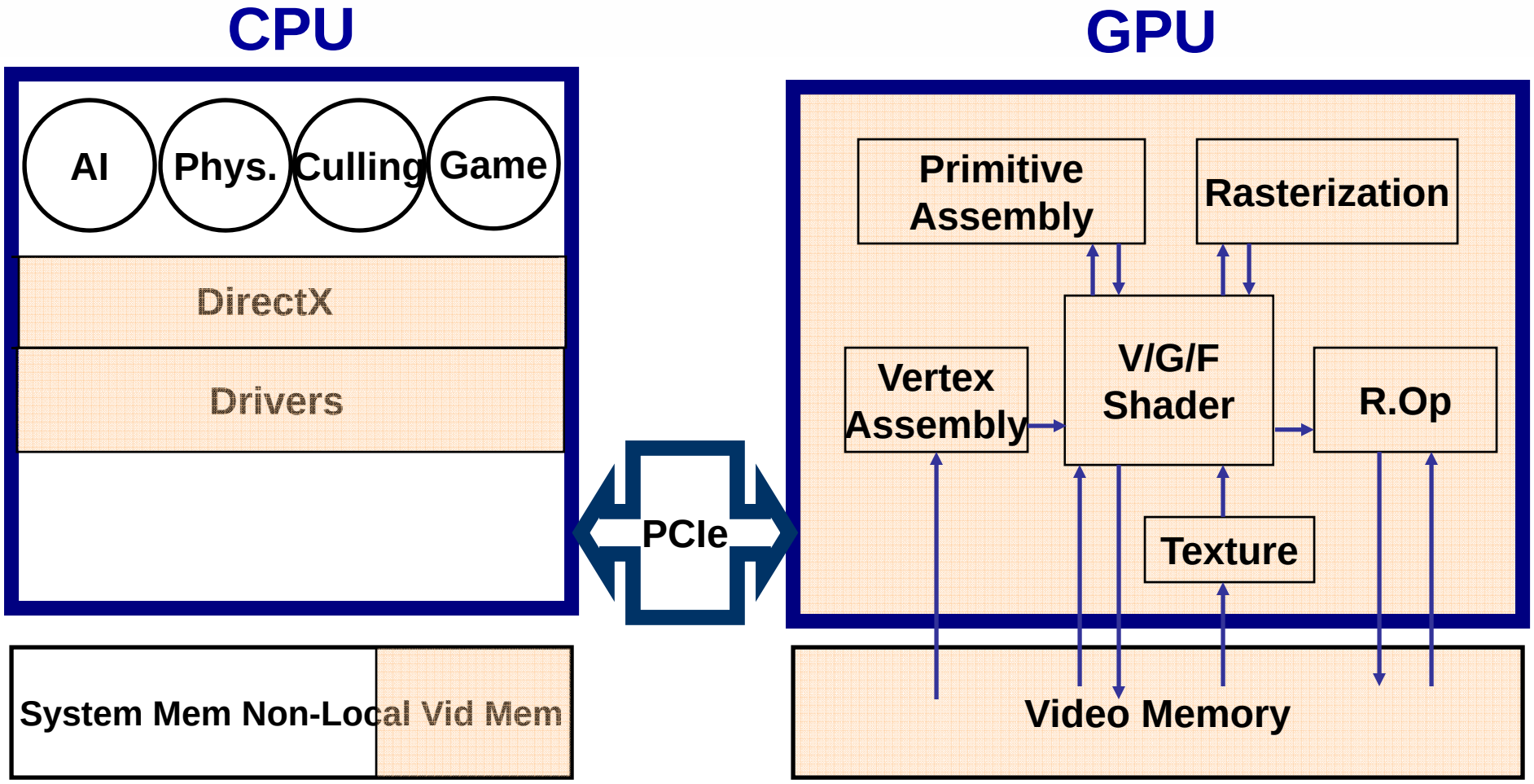
Optimizing Marble Blast Ultra

By: Eric Preisz

Marble Blast Ultra

138 Hz

The Game Pipeline



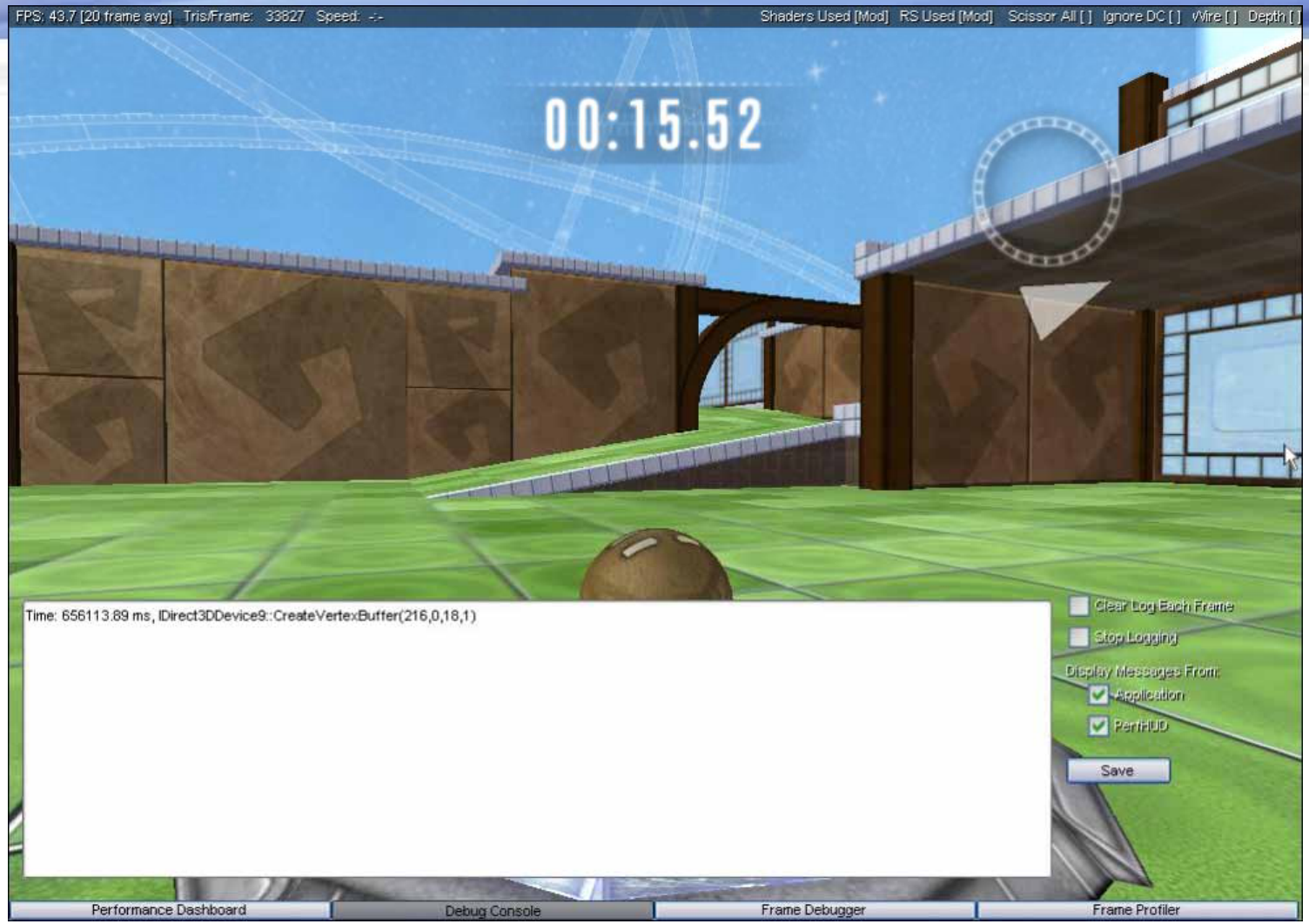
Levels of Optimization

System - % of hardware utilization

Application - Class/function/algorithm

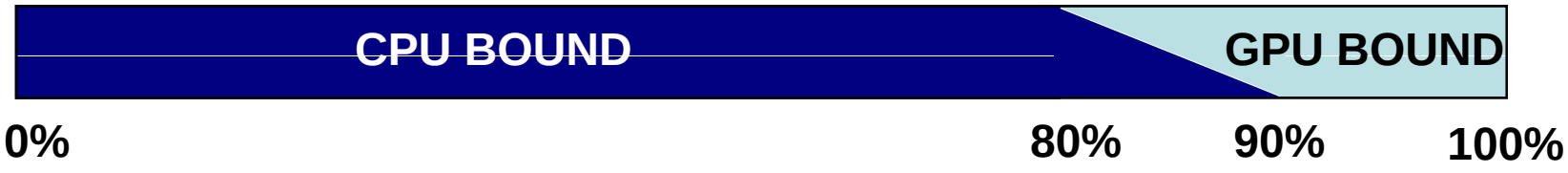
Micro - Line by line optimizations.

Simple Mistakes - Resource Allocations

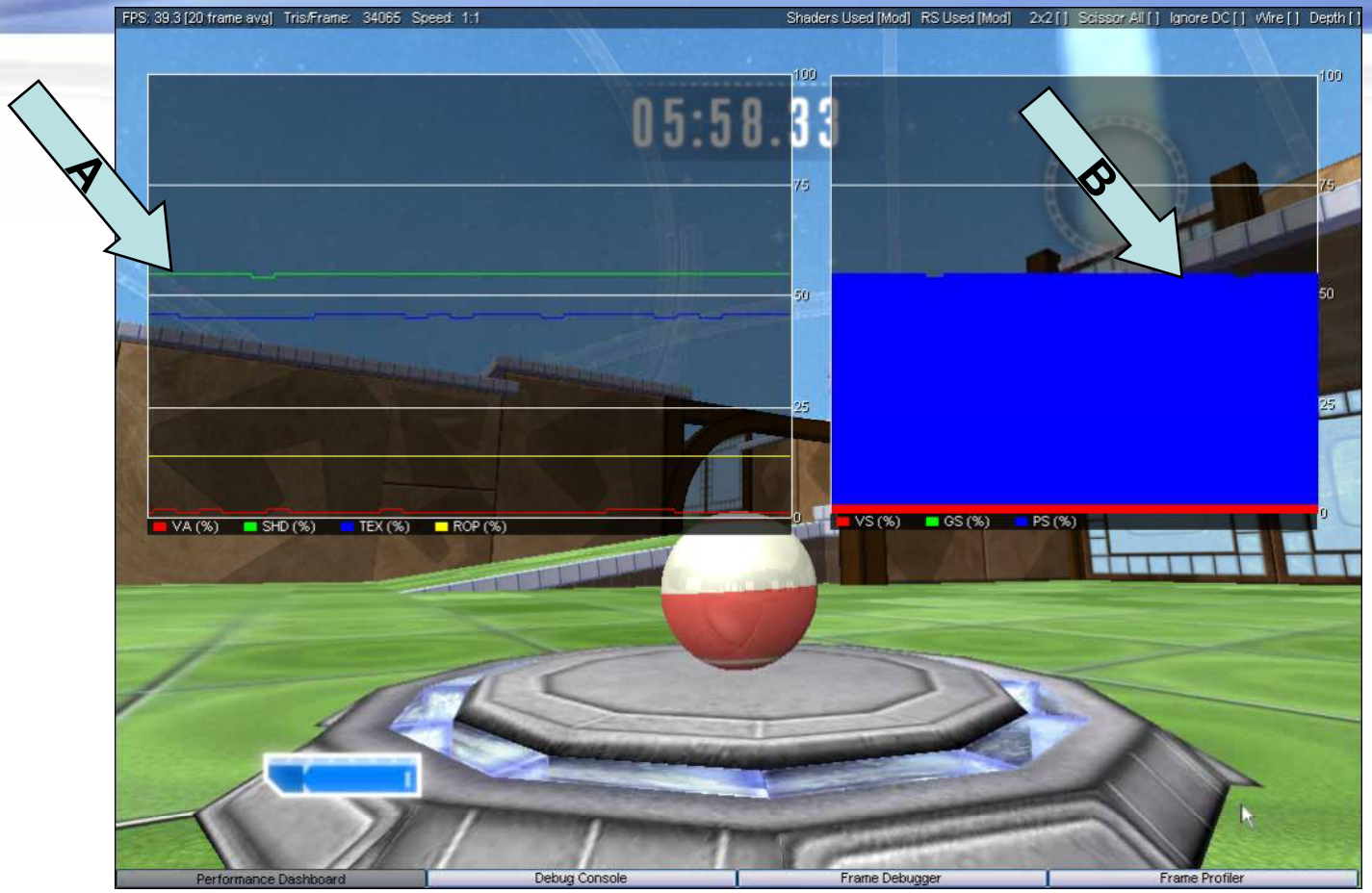


CPU vs GPU
Performance Dashboard

GPU Busy % for the best FPS increase



GPU System Level
Performance Dashboard

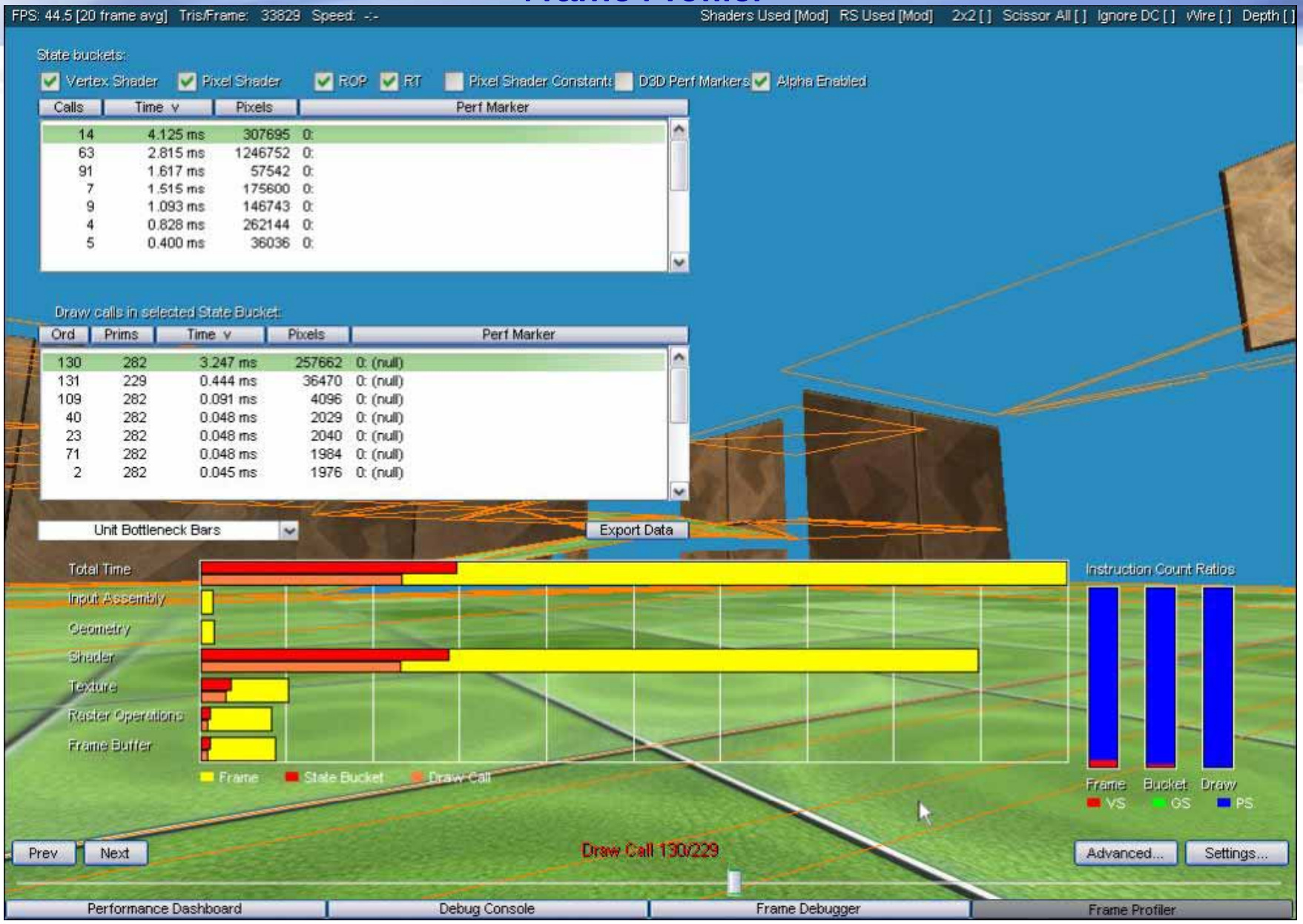


- VA(%) ~5%
- Shader(%) ~55%
- Texture(%) ~45%
- ROP(%) ~25%

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

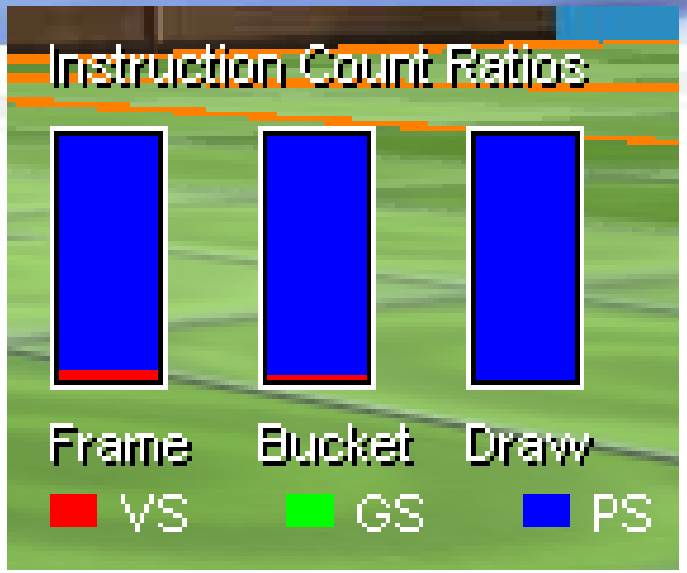
GPU Application Level

Frame Profiler



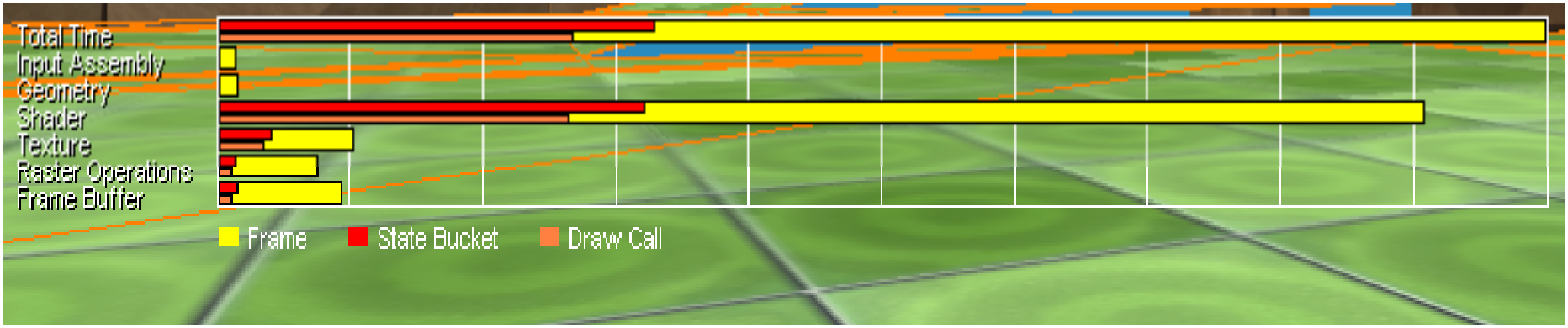
GPU Application Level

Frame Profiler

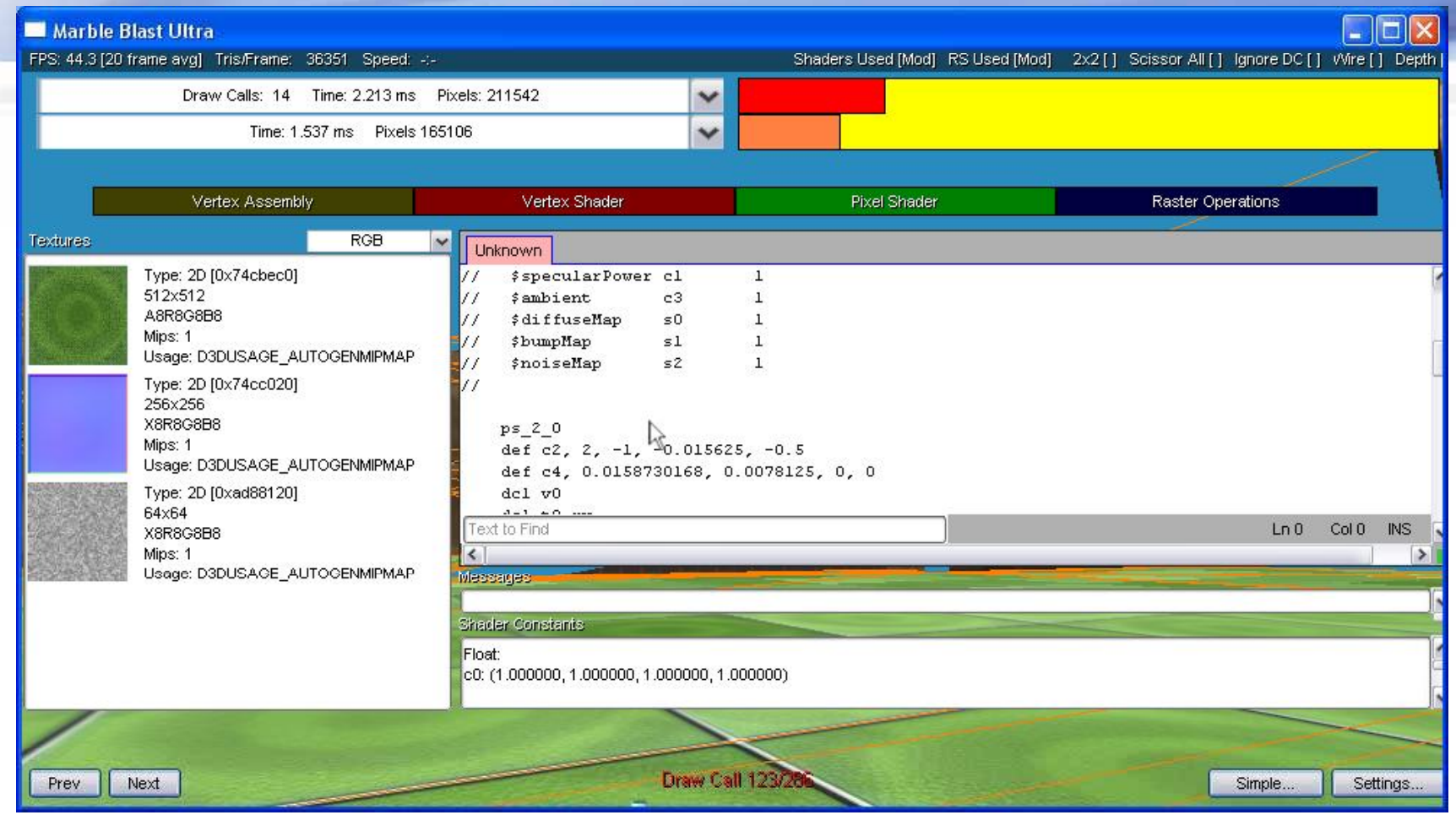


State Bucket = 4.125 ms, 307,695 pixels

Draw Call = 3.247 ms, 257,662 pixels

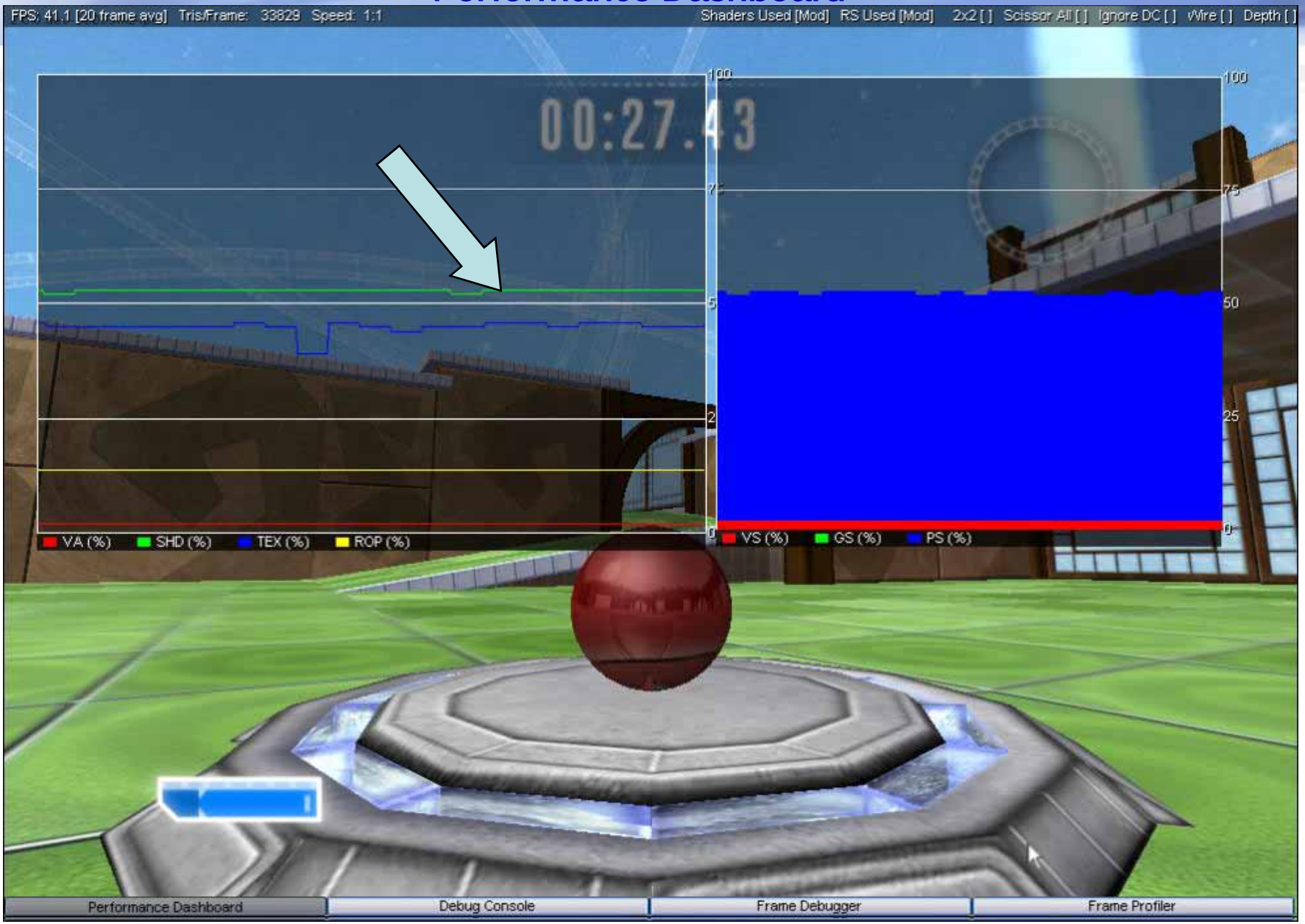


GPU Micro Level

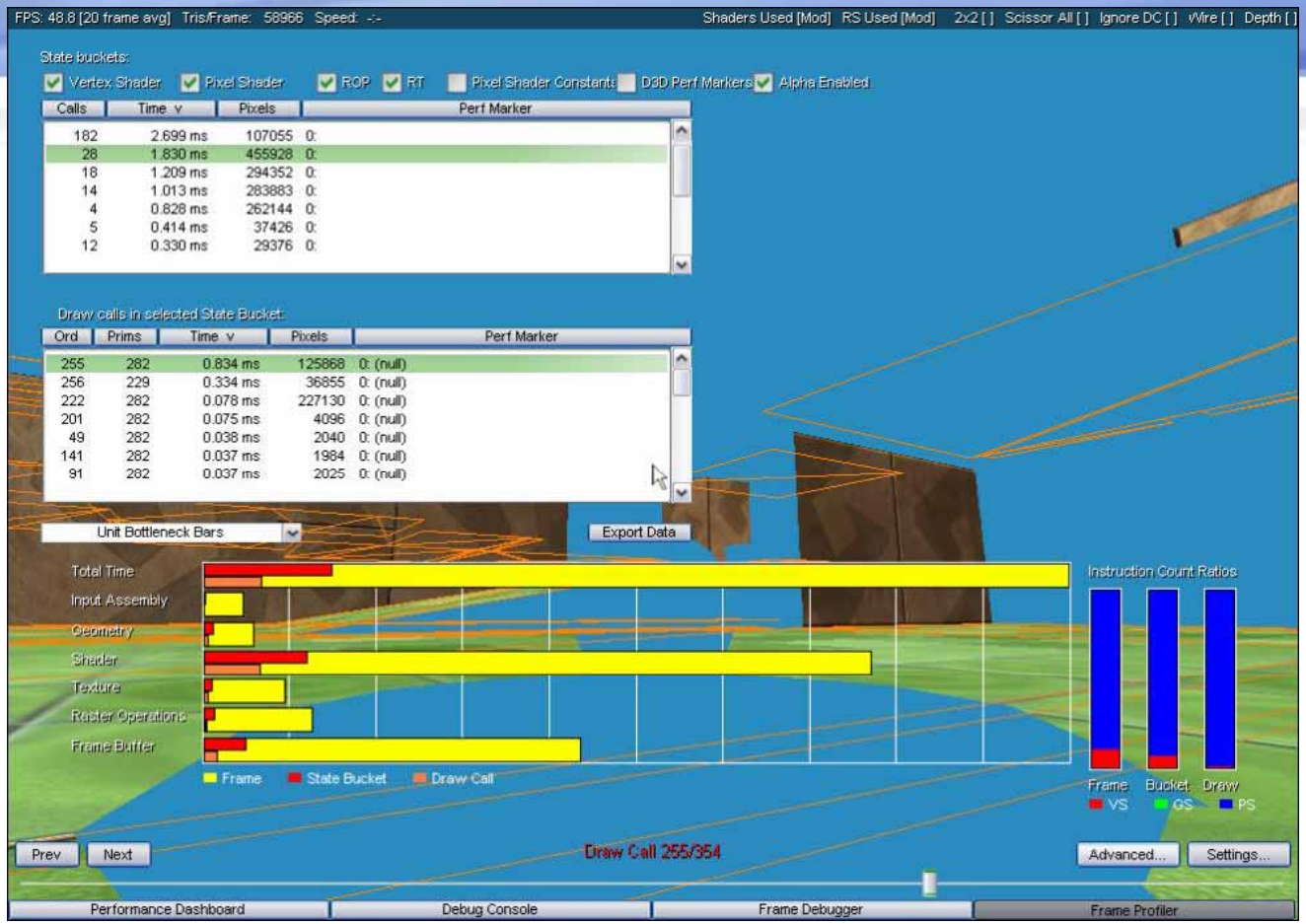


GPU System Level

Performance Dashboard



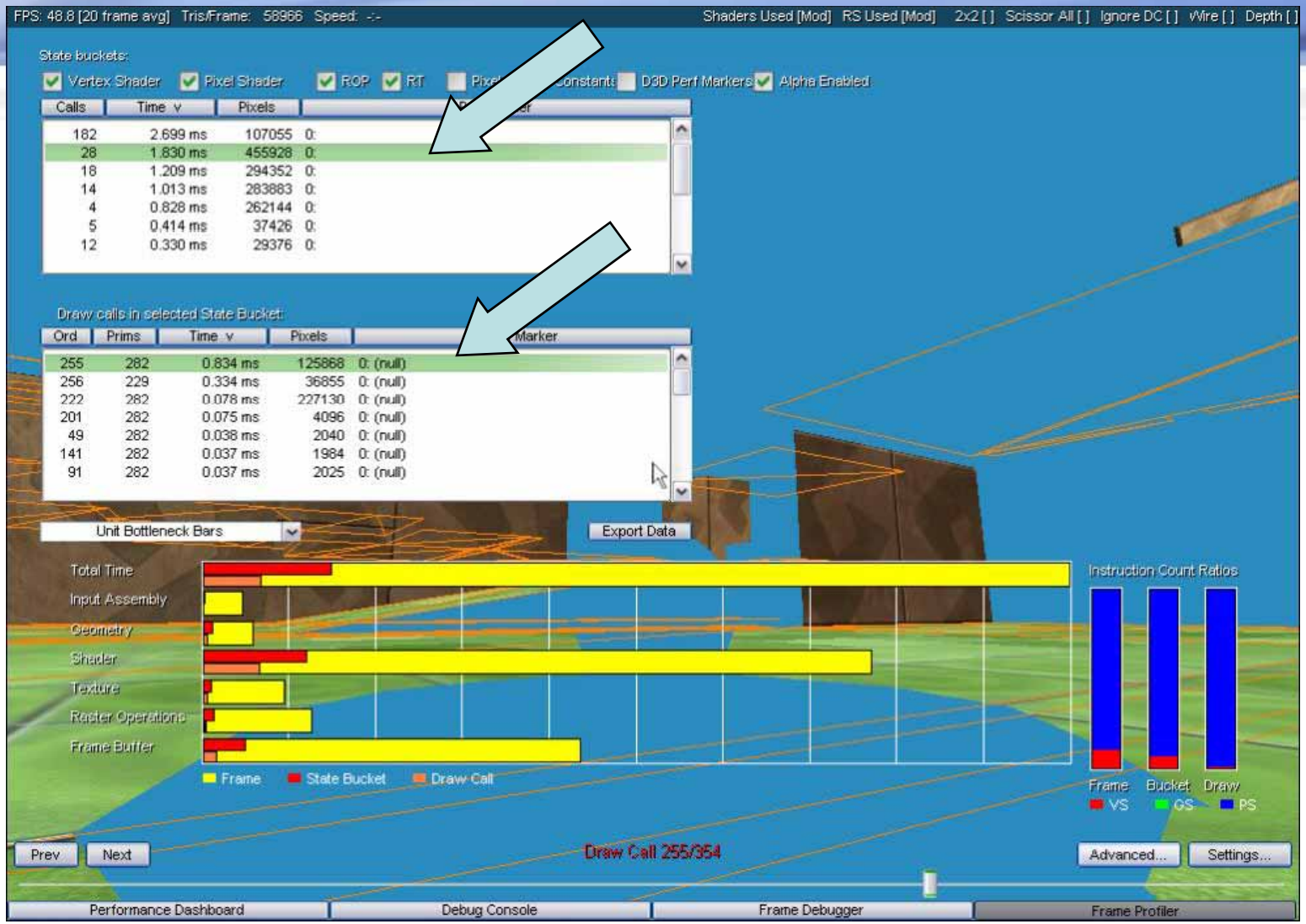
GPU Application Level



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



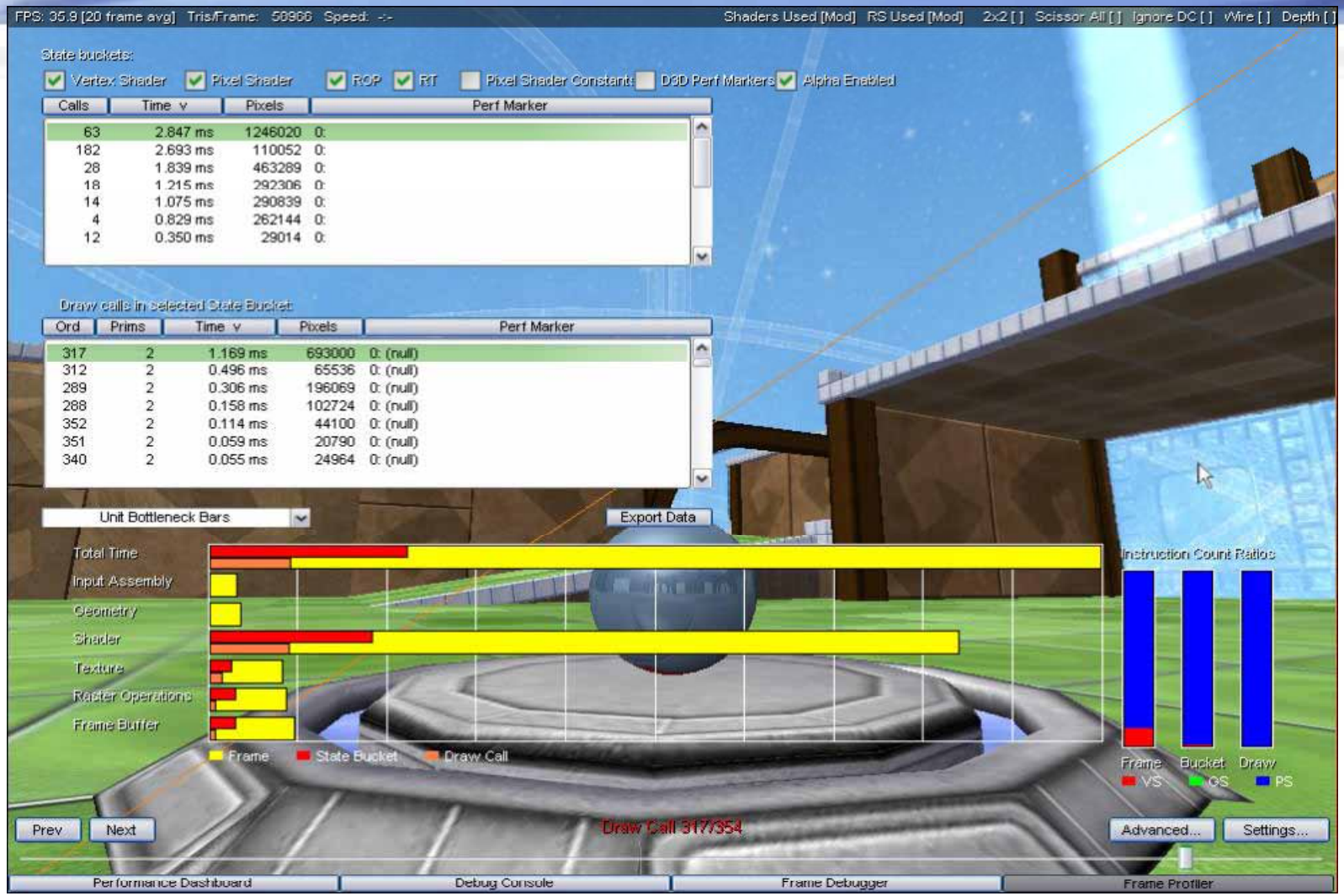
State Bucket = 1.830 ms, 455,928 pixels
Draw Call = 0.834 ms, 125,868 pixels

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.

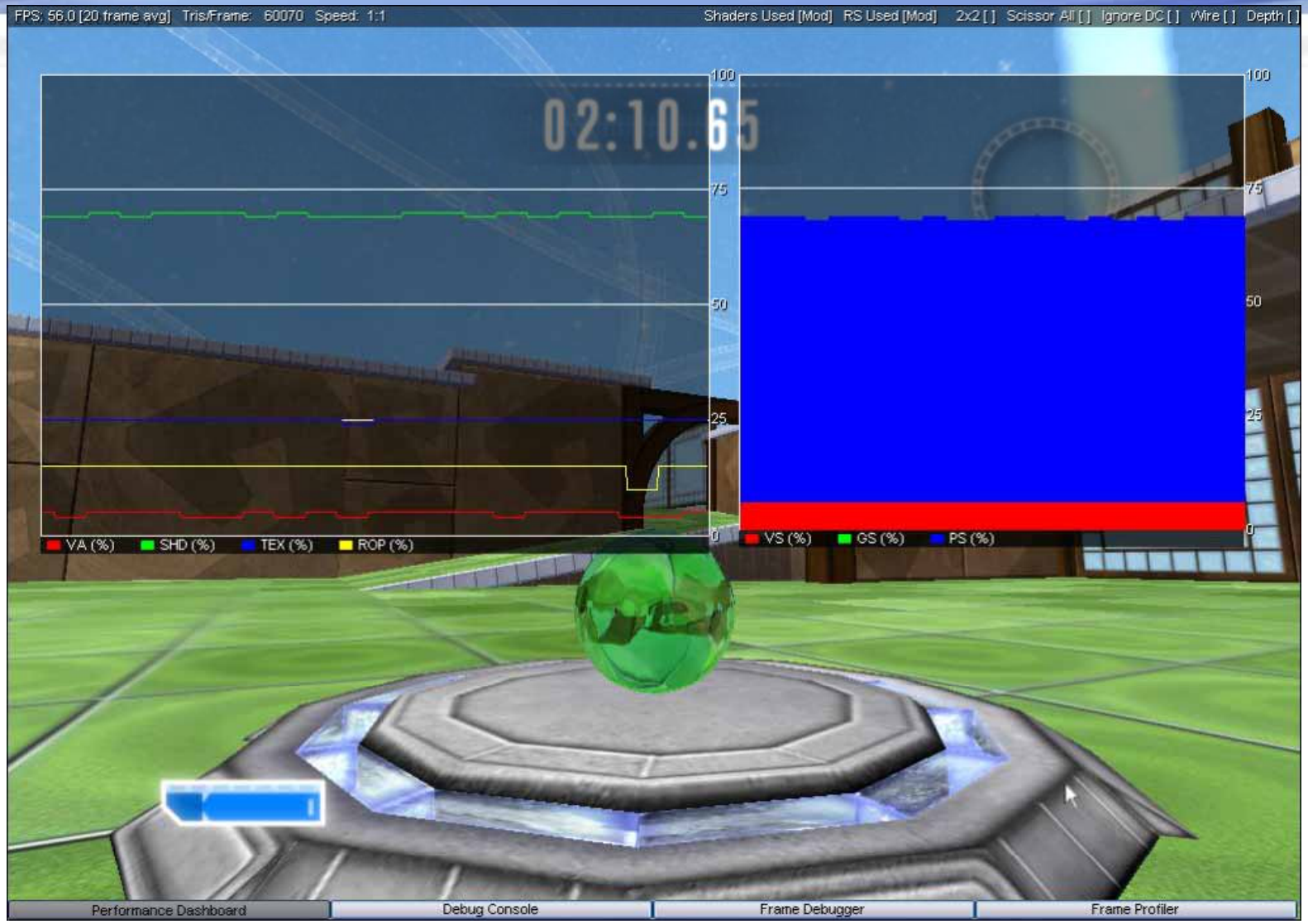
GPU System Level

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…

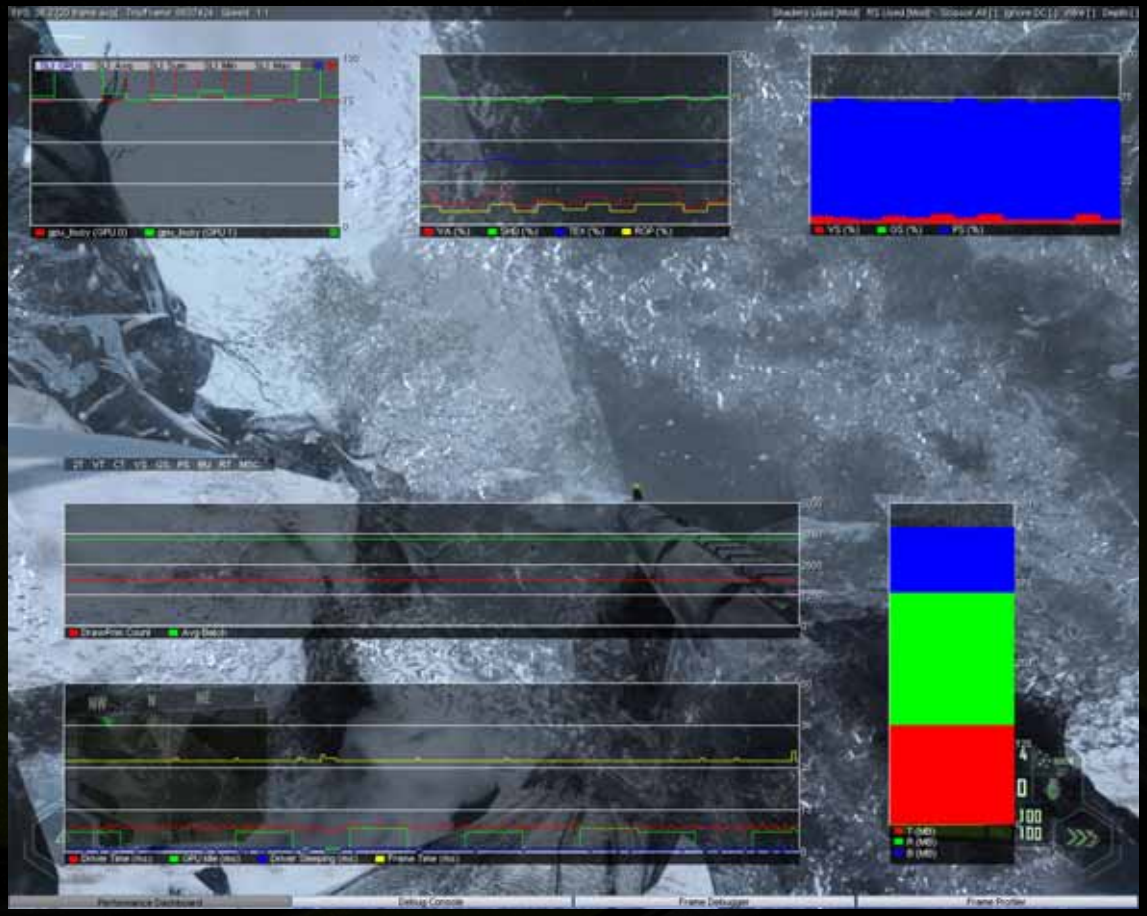
- Investigate more pixel optimizations – Scissor test confirms.
- Compressed texture usage – forcing 2x2 gives ~5 frame improvement.
- SLI Optimizations
- Investigate CPU optimizations, on the 8800 GPU utilization is only 25%

Thanks Eric!!!

Introducing PerfHUD 6.0!

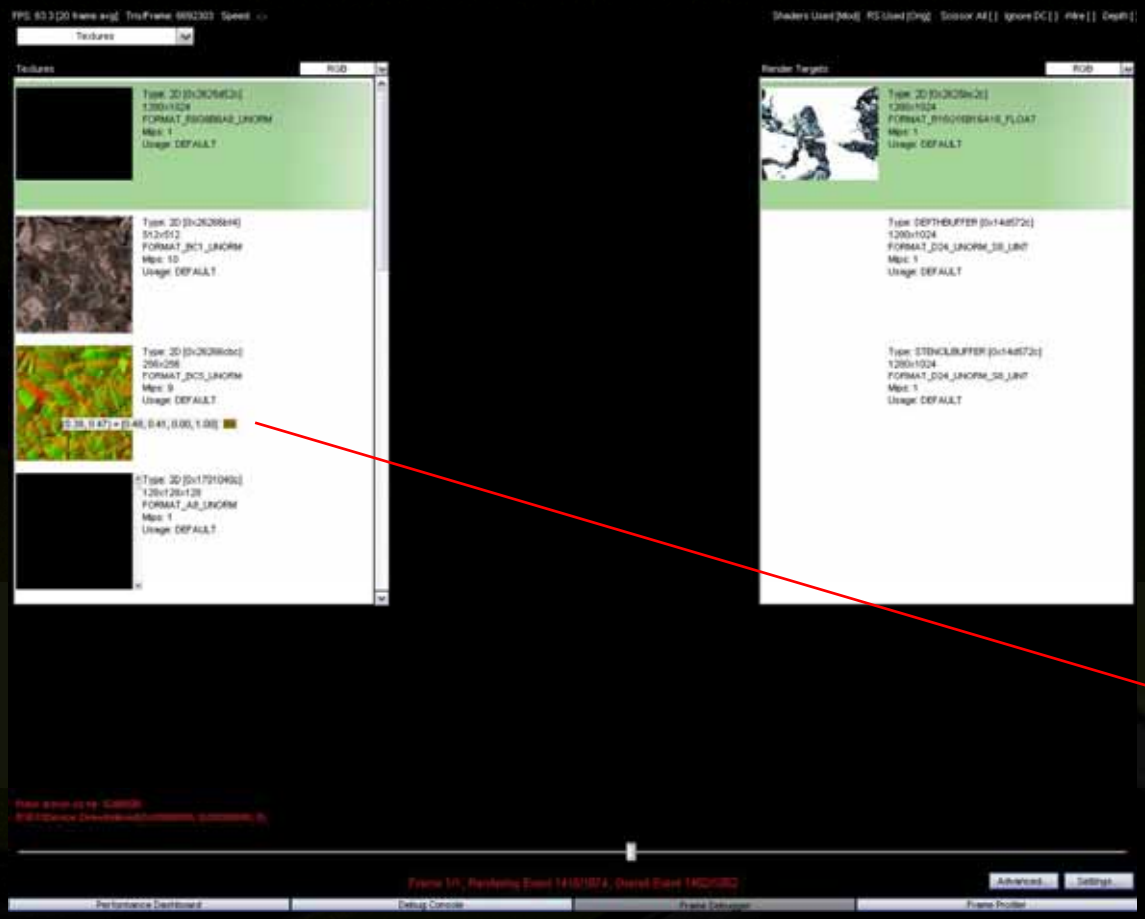
- Unified Driver on Vista: use any release driver!
- Comprehensive SLI Support
  - Graphs for SLI specific data
  - Insight into SLI performance gotchas
- Powerful new debugging features
  - Texture visualization modes
  - API Call data mining and analysis
  - Shader visualization
- Usability Features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

PerfHUD 6: Performance Dashboard



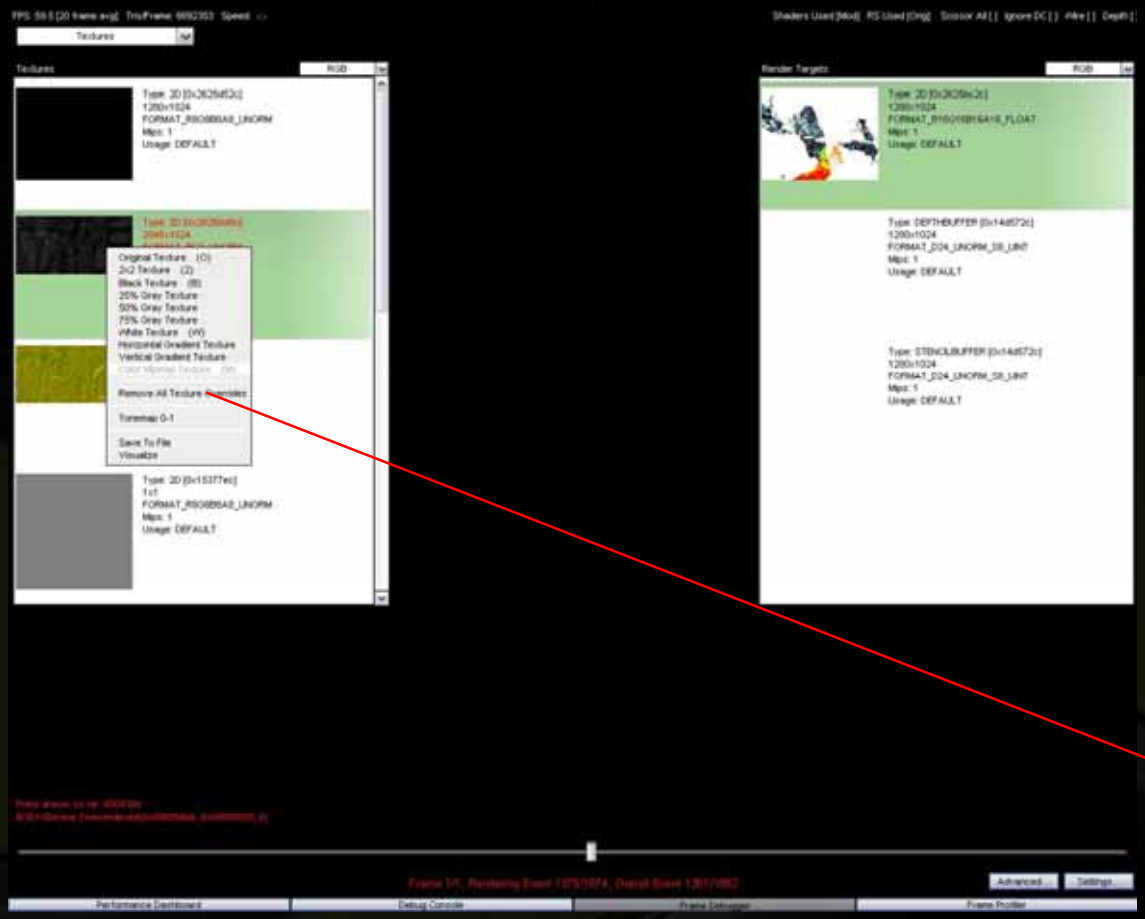
- Graph GPU and driver data
- Edit to suit your needs
- SLI Graph for multi-GPU
- API usage statistics

PerfHUD 6: Frame Debugger



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

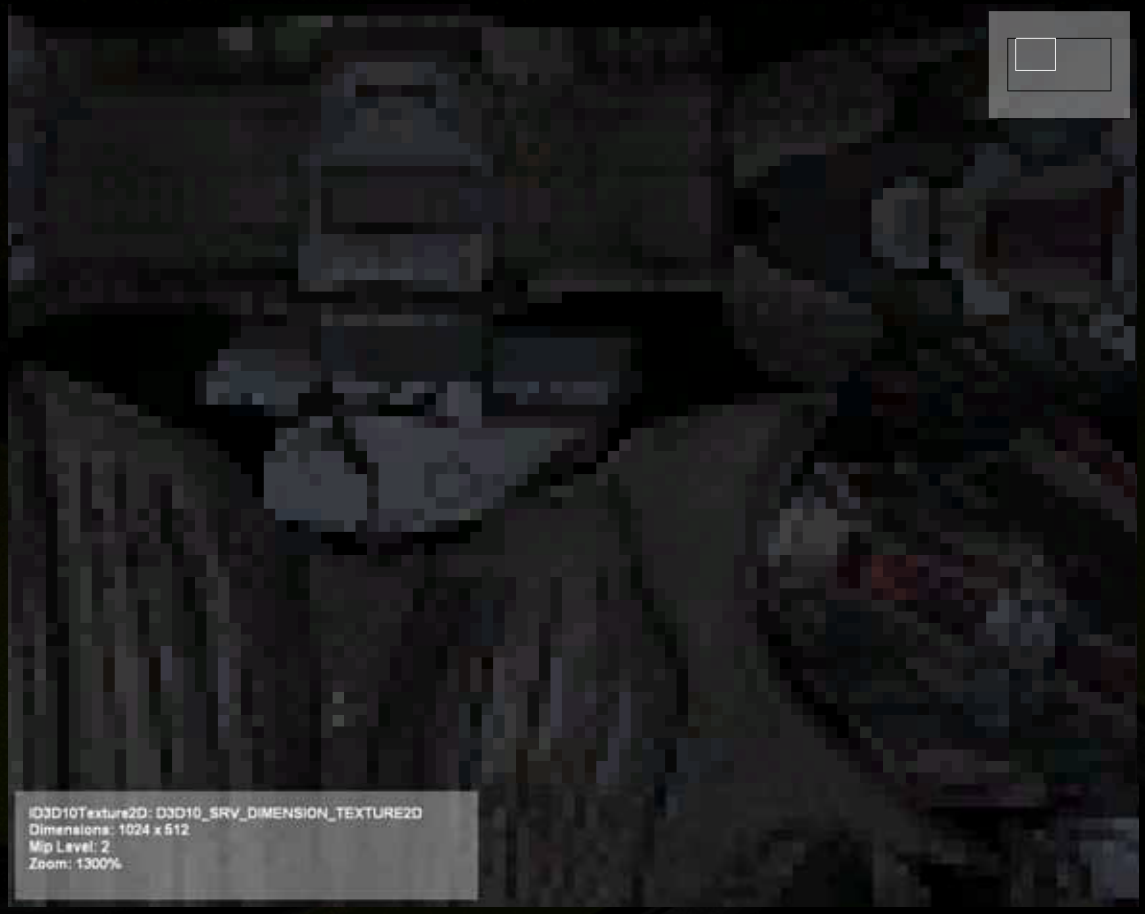
PerfHUD 6: Frame Debugger



Texture analysis: substitute precomputed textures
Controllable via Perf Markers

PerfHUD6: Frame Debugger

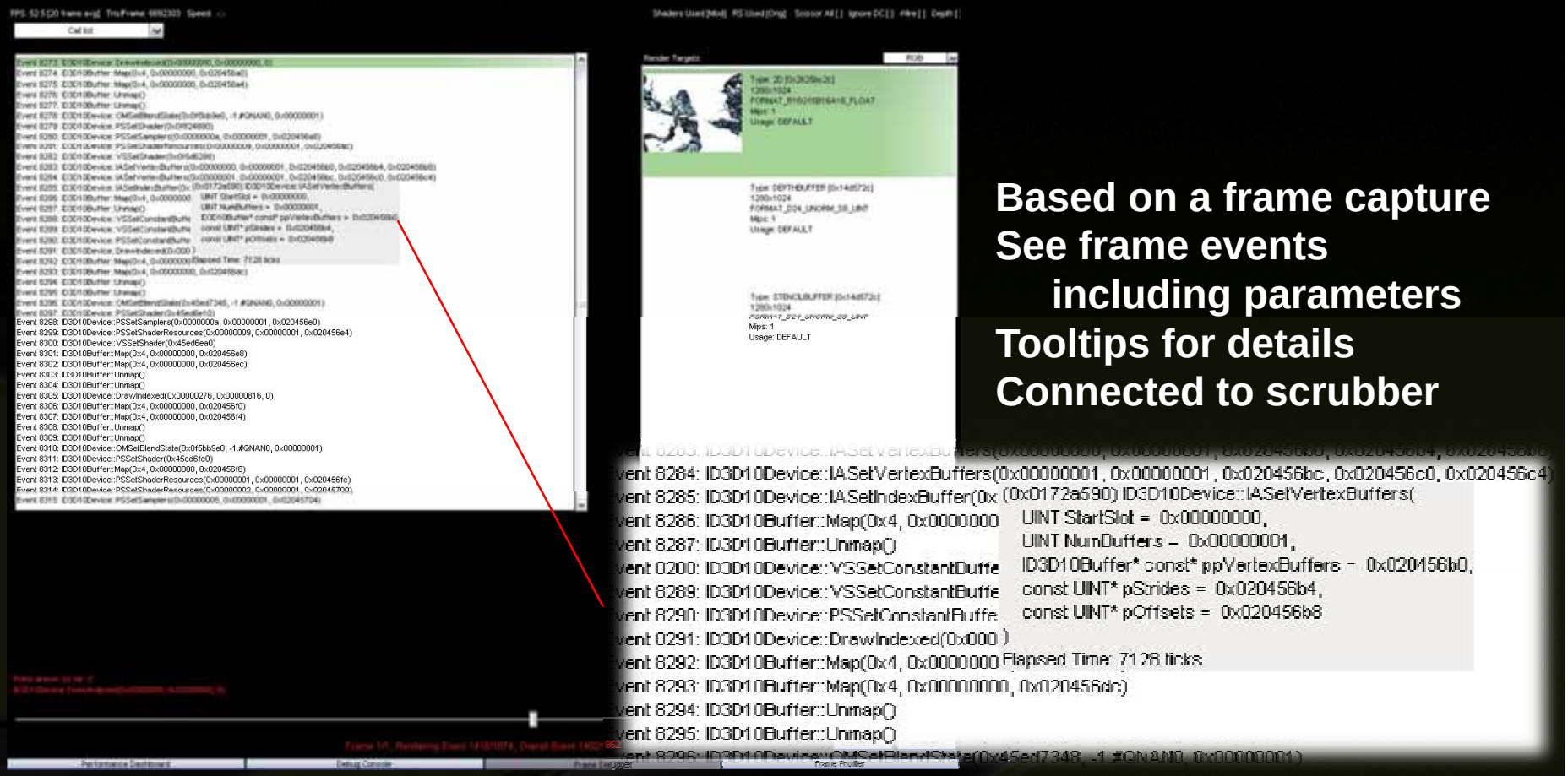
Buffer Visualization



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

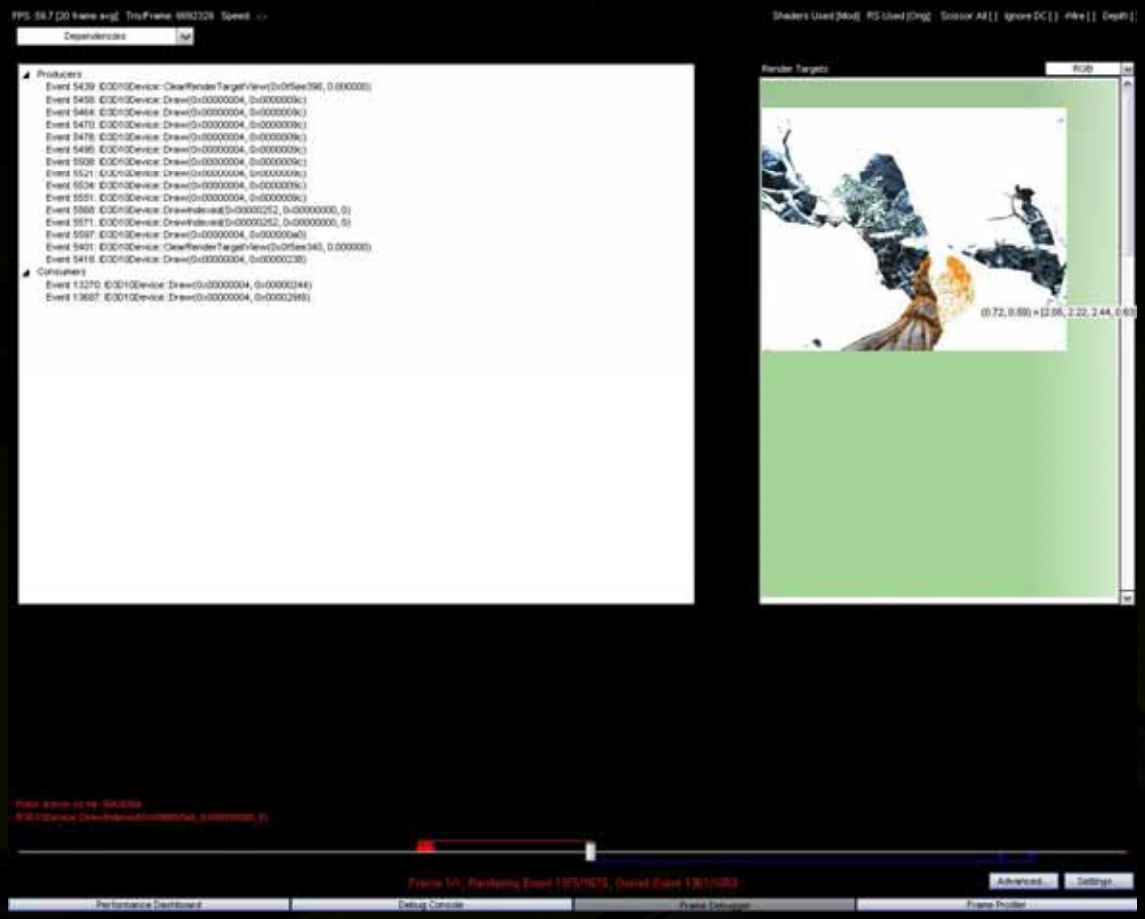
PerfHUD6: Frame Debugger

API Call List



PerfHUD 6: Frame Debugger

Draw Call Dependencies

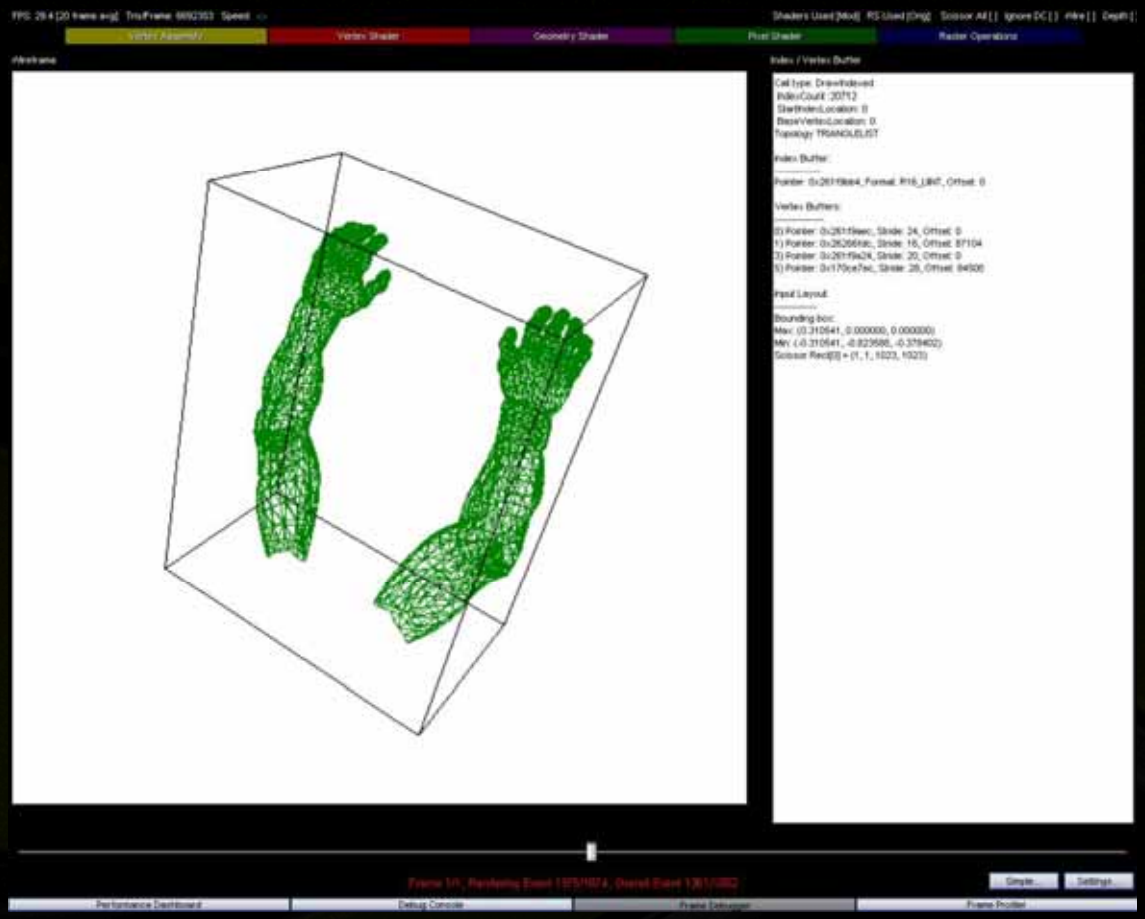


Show producers & consumers dependencies for each call

These can hurt single GPU and SLI performance

PerfHUD 6: Adv Frame Debugger

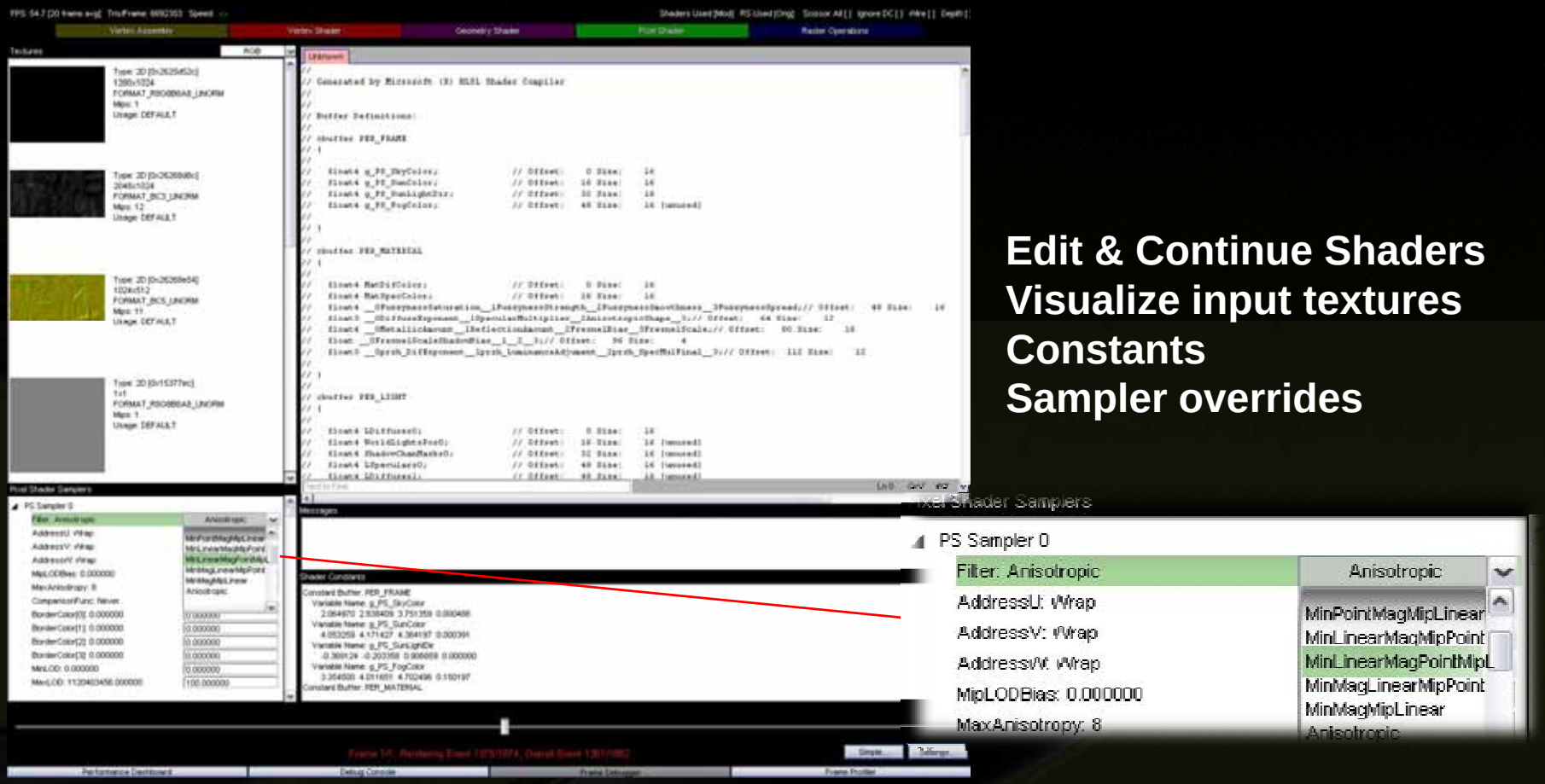
Vertex Assembly



Geometry Preview
Vertex and index buffer setup

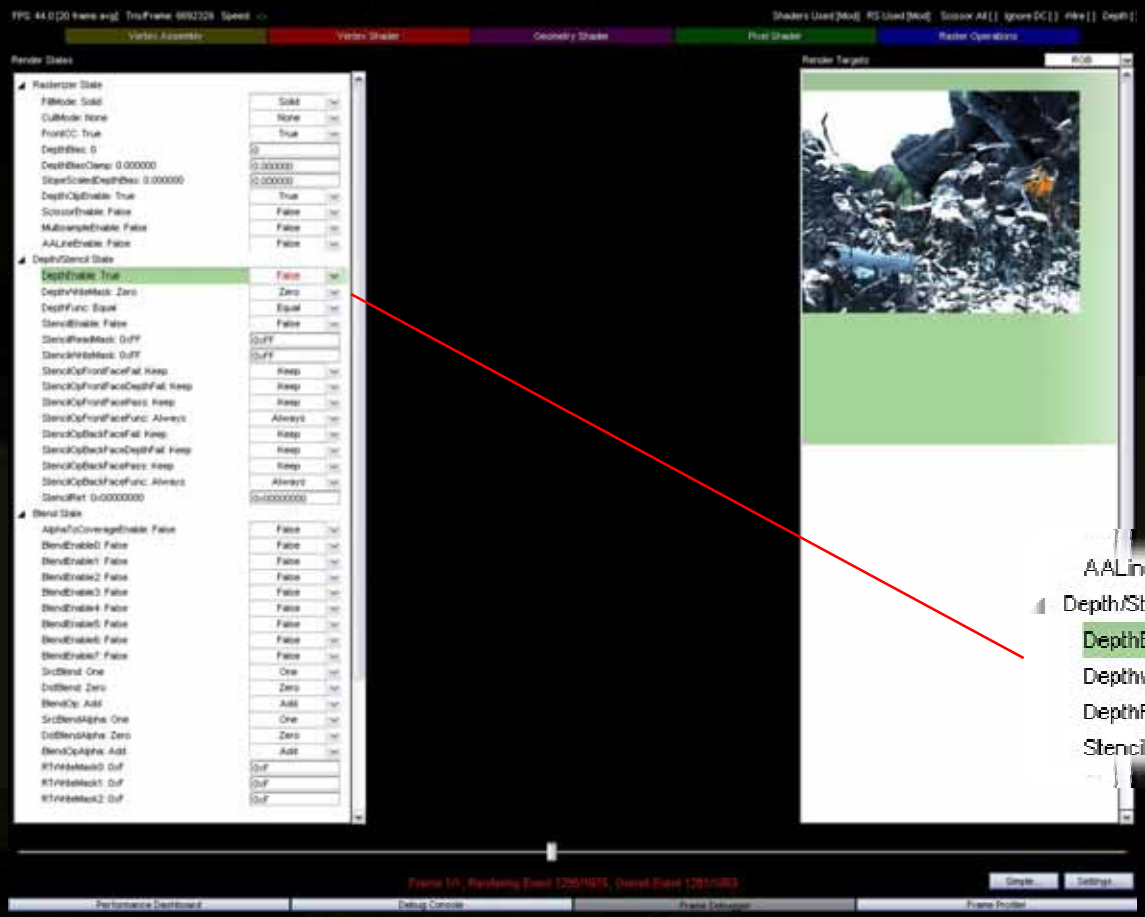
PerfHUD 6: Adv Frame Debugger

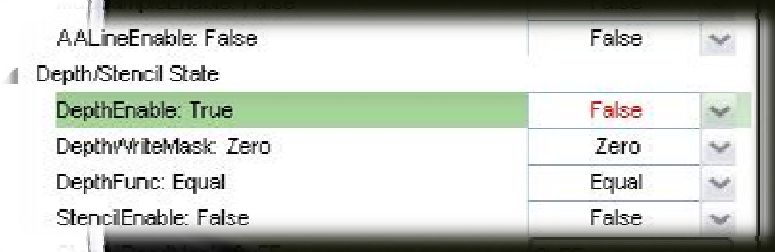
Vertex, Geometry and Pixel Shaders



Edit & Continue Shaders
Visualize input textures
Constants
Sampler overrides

PerfHUD 6: Adv Frame Debugger



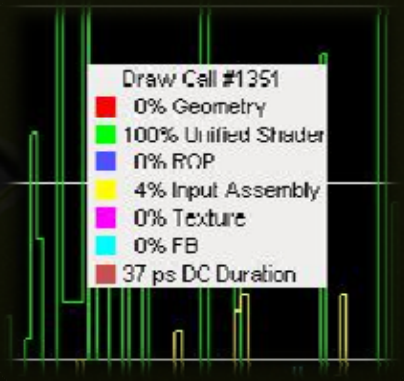


Display and modify all render state settings
Render targets displayed

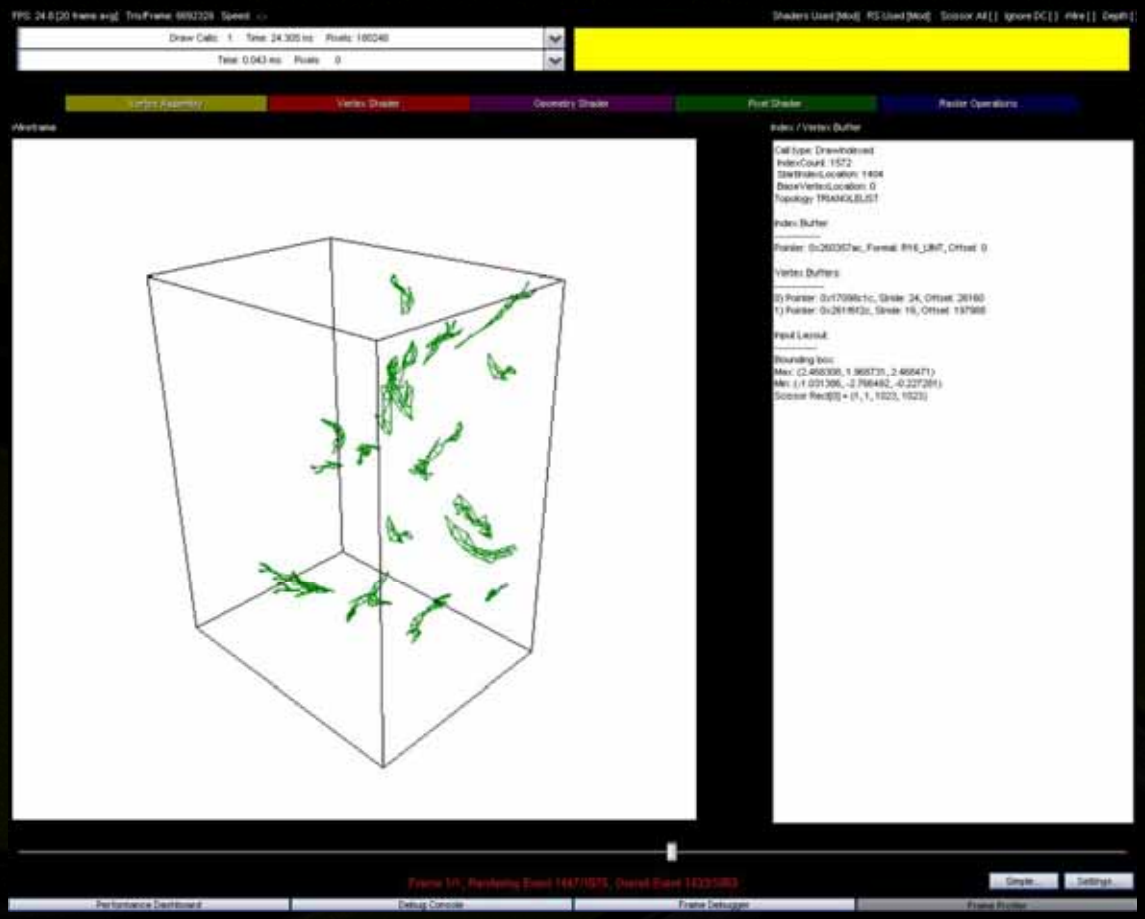
PerfHUD 6: Frame Profiler

One button bottleneck determination

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



PerfHUD 6: Adv Frame Profiler



Same advanced features now in the profiling context

PerfHUD 6

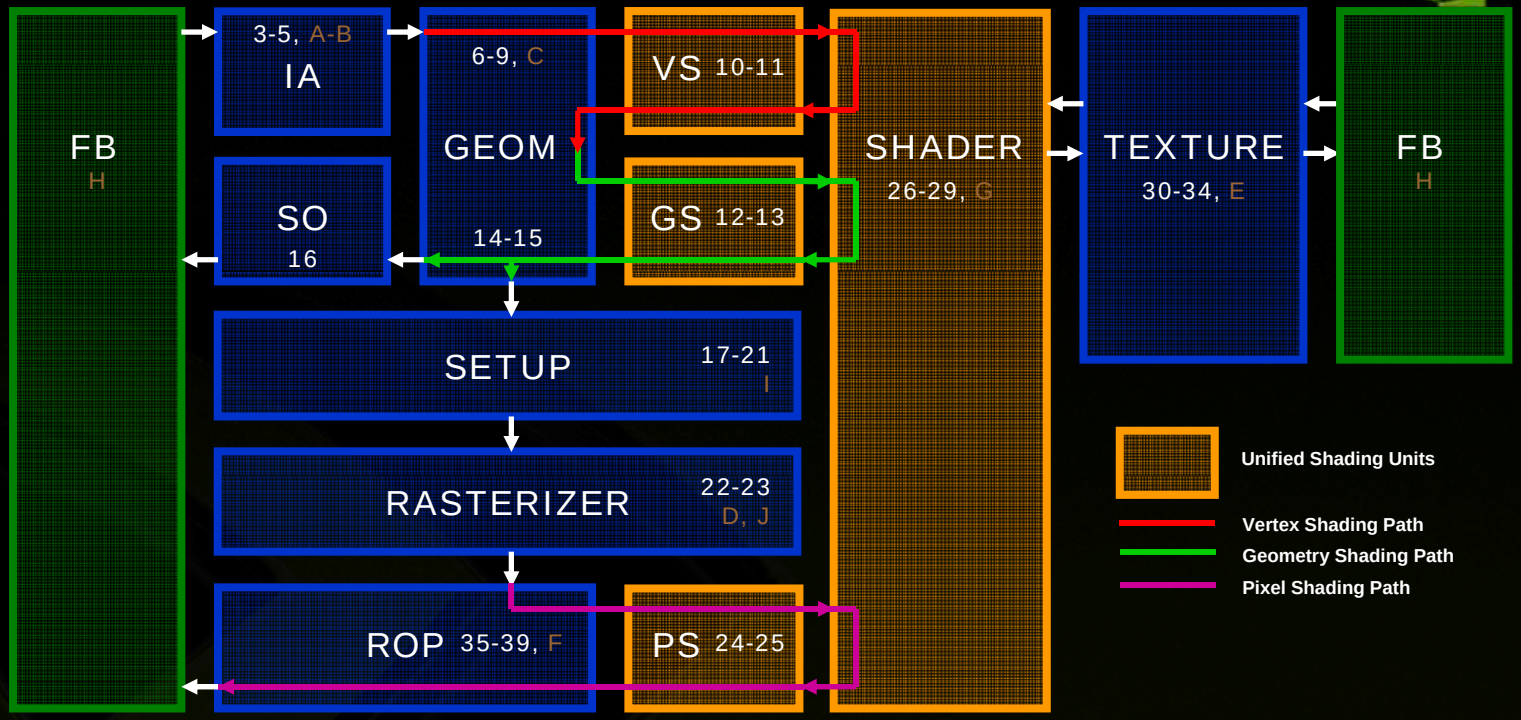
A taste of things to come!

- 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

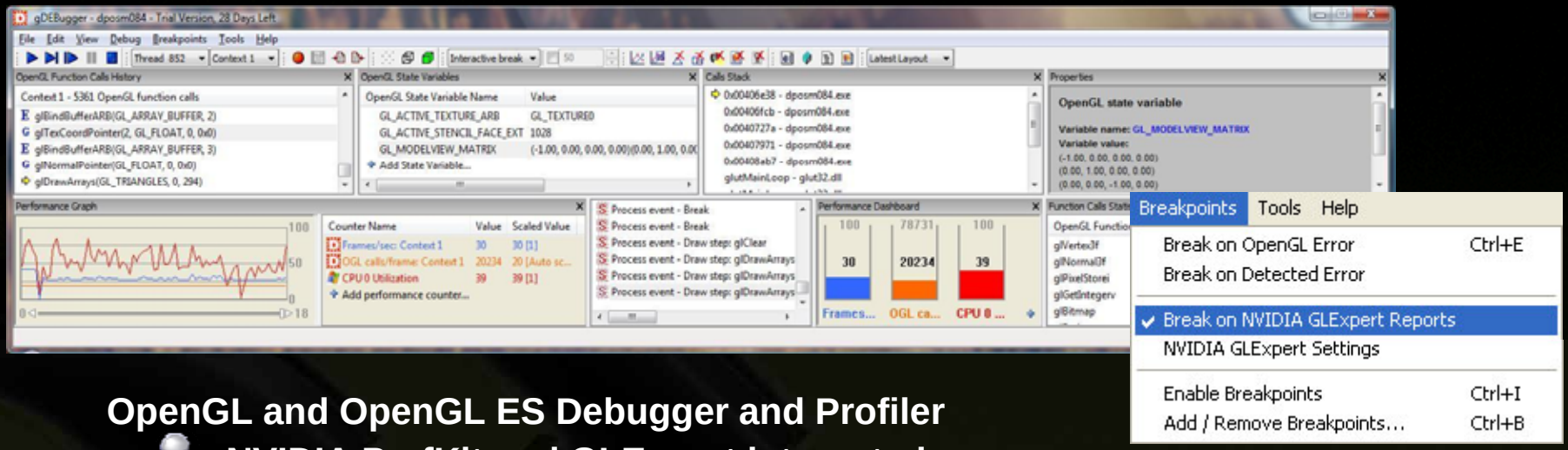
Simplified G80+ Counter Diagram

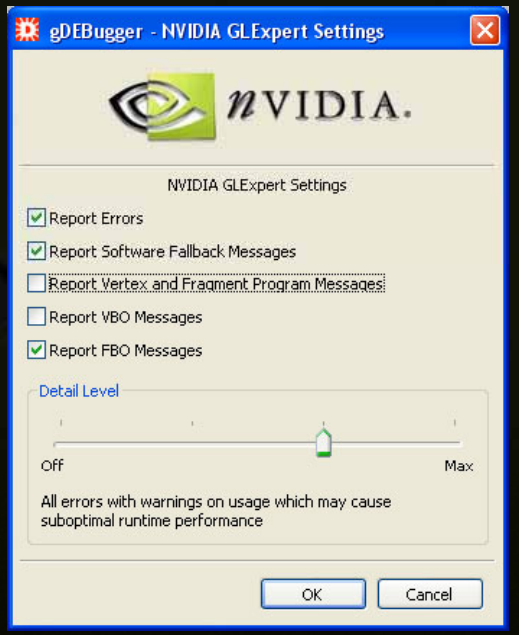


1. gpu_idle (Global)
2. gpu_busy (Global)
3. input_assembler_busy
4. input_assembler_waits_for_fb
5. vertex_attribute_count
6. geom_busy
7. geom_waits_for_shader
8. geom_vertex_in_count
9. geom_primitive_in_count
10. vertex_shader_busy
11. vertex_shader_instruction_rate
12. geometry_shader_busy
13. geometry_shader_instruction_rate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



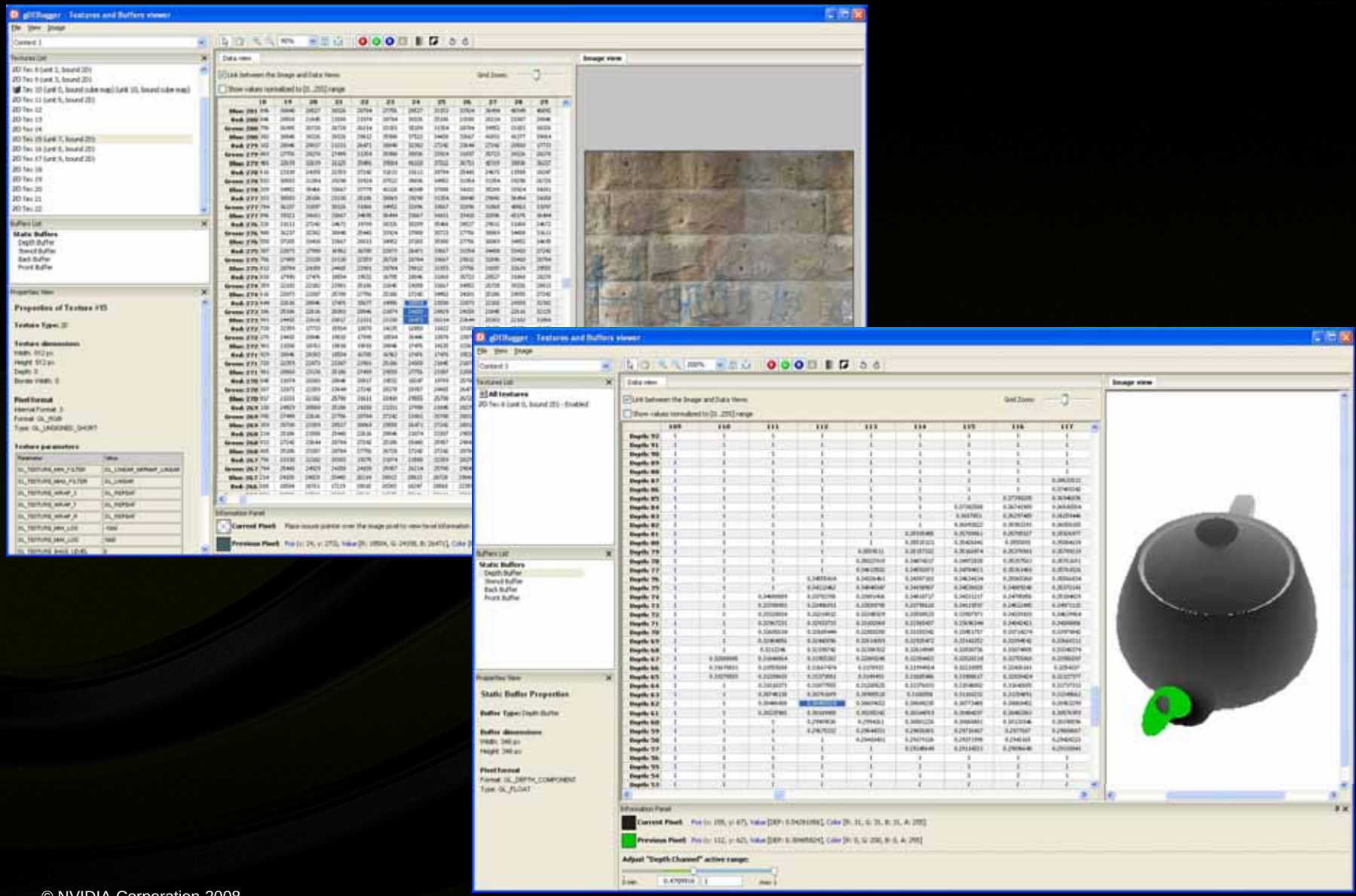


OpenGL and OpenGL ES Debugger and Profiler

- NVIDIA PerfKit and GLExpert 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



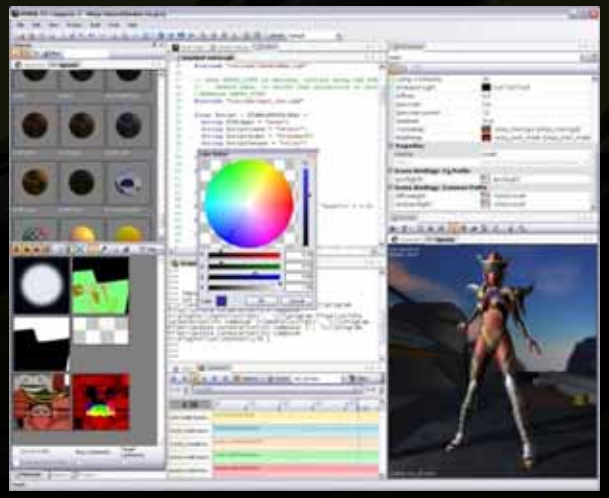
ShaderPerf 2.0

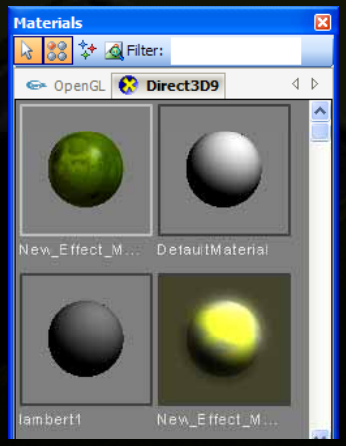
- Determine shader performance
  - Compare with shader cycle budgets
  - Test optimization opportunities
- Automated regression analysis
- Integrated in FX Composer
  - Interactive GUI
  - Artists/TDs code expensive shaders
  - Achieve optimum performance
- GeForce 8 series performance data
- Beta 2.0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

FX Composer

Shader Authoring Made Easy!

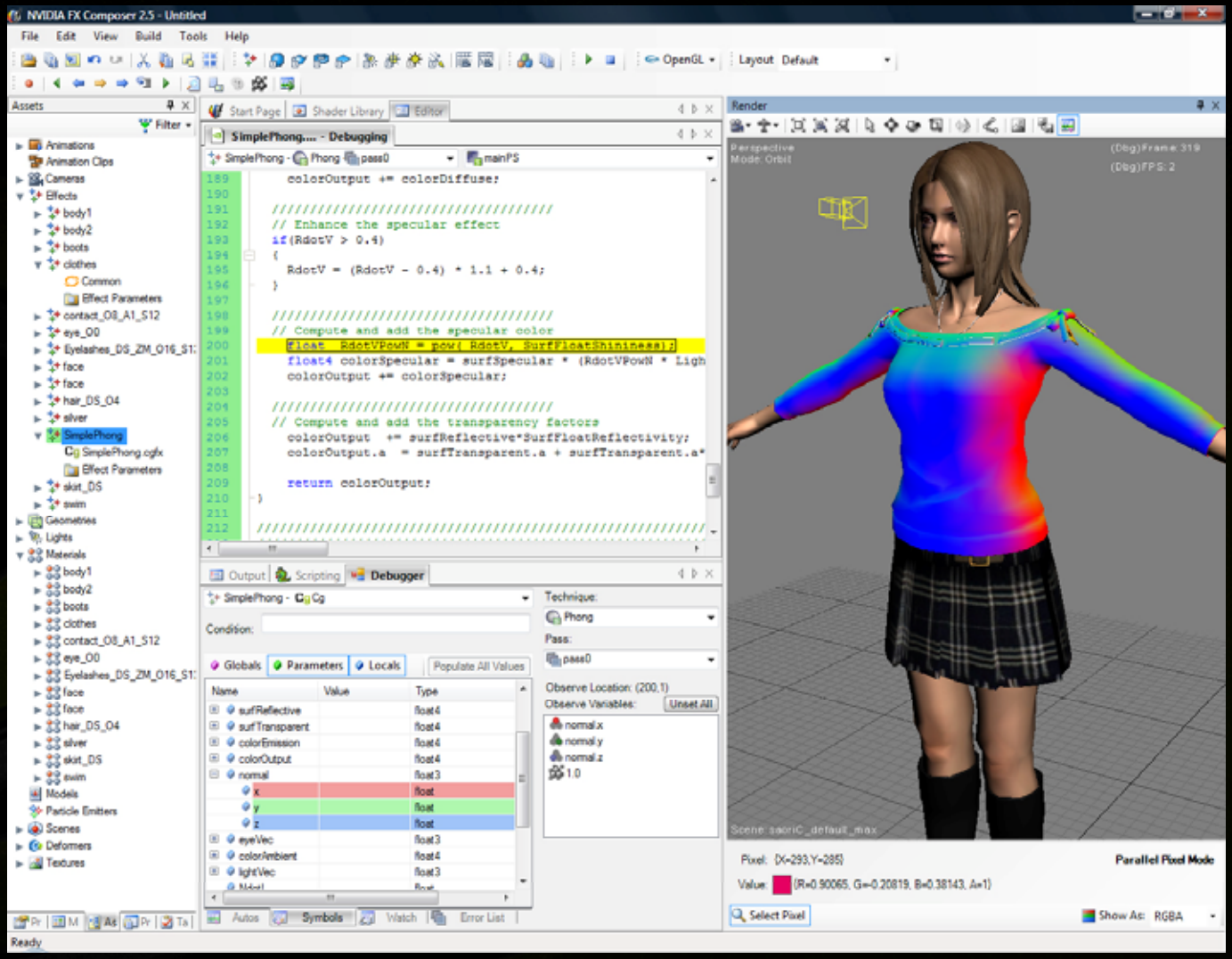
- DirectX 10 backend
- Shader Debugger
- GeForce 8 Series Shader Performance
- Full-featured code editor
- Shader creation wizard with templates
- Integration with online Shader Library
- Materials panel to organize materials





FX Composer

HLSL10 Shader Debugging!



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