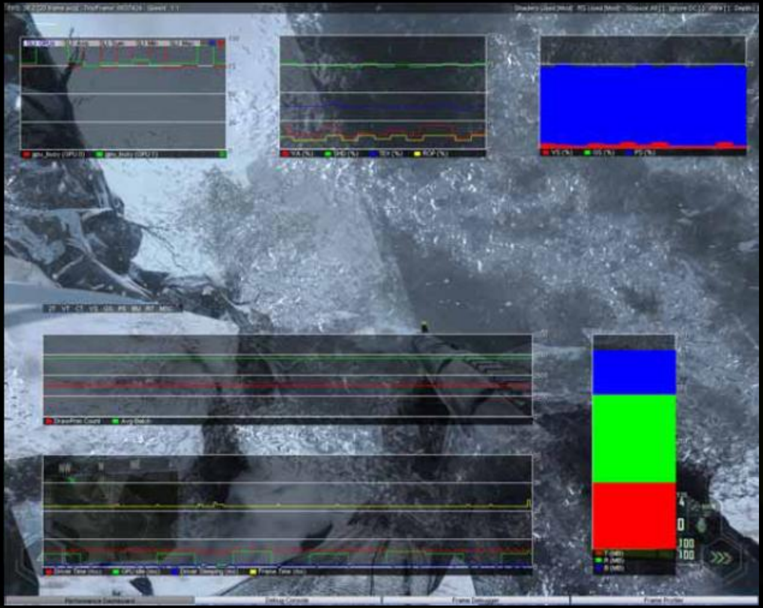
# NVIDIA PerfKit 6

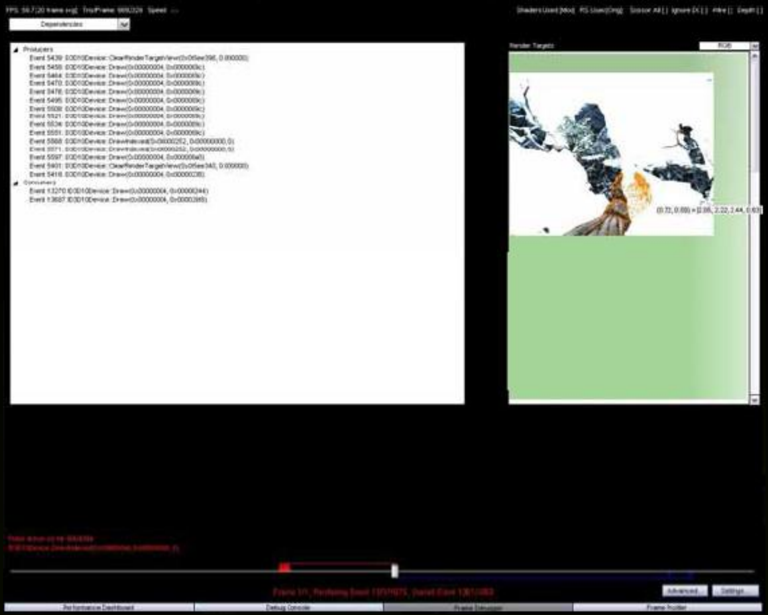
# NVIDIA PerfKit 6: The World’s Most Advanced GPU Performance Suite

- PerfHUD 6
  - *New!* GeForce 8800GT, 9600GT, 9800GX2 support
  - *New!* Use stock NVIDIA drivers with PerfHUD (on Windows Vista)
  - *New!* Multi-GPU Support
  - *New!* More real-time signals, including SLI signals
  - *New!* API Call List
  - *New!* Draw Call Dependency Analysis
  - *New!* Advanced Texture Visualization
  - *New!* Texture Overrides
  - Real-time performance analysis and debugging
  - Automated bottleneck determination
- PerfSDK
  - *New!* GeForce 8800GT, 9600GT, 9800GX2 support
  - API for accessing GPU and driver counters
  - Supports DirectX 9 & 10, as well as OpenGL
- NVIDIA Plug-in for Microsoft PIX for Windows
- GLExpert

# PerfHUD 6



Multi-GPU Support



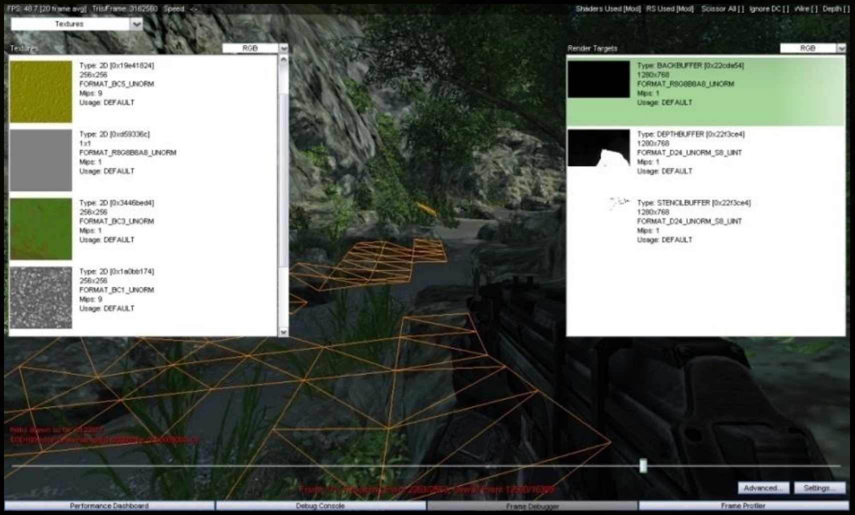
Producers

Event 5439: D3D10Device::ClearRenderTargetView(0x05ee398, 0.000000)
Event 5458: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5464: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5470: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5476: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5495: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5508: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5521: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5534: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5551: D3D10Device::Draw(0x00000004, 0x0000008c)
Event 5568: D3D10Device::DrawIndexed(0x00000252, 0x00000000, 0)
Event 5571: D3D10Device::DrawIndexed(0x00000252, 0x00000000, 0)
Event 5597: D3D10Device::Draw(0x00000004, 0x000006a0)
Event 5401: D3D10Device::ClearRenderTargetView(0x05ee340, 0.000000)
Event 5418: D3D10Device::Draw(0x00000004, 0x00000238)

Consumers

Event 13270 ID3D10Device::Draw(0x00000004, 0x00000244)
Event 13687 ID3D10Device::Draw(0x00000004, 0x00000365)

Advanced Data Mining



FPS: 48.7 [20 frame avg]   Tris/Frame: 3162560   Speed: -:-

Shaders Used [Mod]   RS Used [Mod]   Scissor All [ ]   Ignore DC [ ]   Wire [ ]   Depth [ ]

Textures

Type: 2D [0x19e41824]
256x256
FORMAT_BC5_UNORM
Mips: 9
Usage: DEFAULT

Type: 2D [0xd59336c]
1x1
FORMAT_R8G8B8A8_UNORM
Mips: 1
Usage: DEFAULT

Type: 2D [0x3446bed4]
256x256
FORMAT_BC3_UNORM
Mips: 1
Usage: DEFAULT

Type: 2D [0x1a0bb174]
256x256
FORMAT_BC1_UNORM
Mips: 9
Usage: DEFAULT

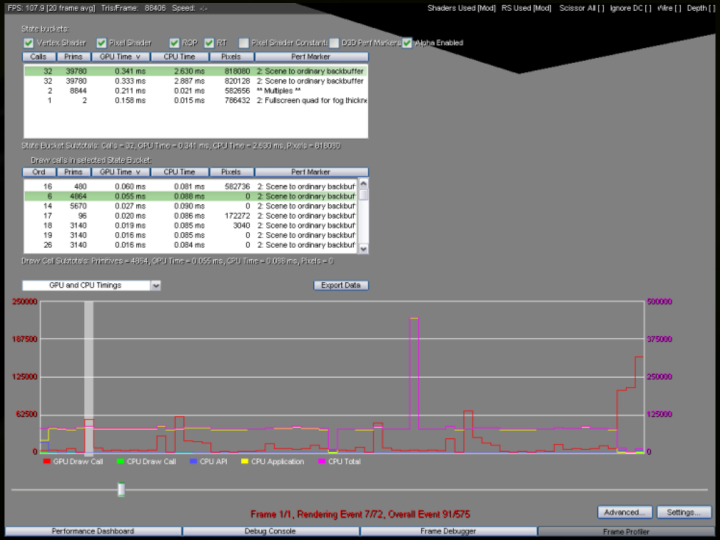
Render Targets

Type: BACKBUFFER [0x22cda54]
1280x768
FORMAT_R8G8B8A8_UNORM
Mips: 1
Usage: DEFAULT

Type: DEPTHBUFFER [0x22f3ce4]
1280x768
FORMAT_D24_UNORM_S8_UINT
Mips: 1
Usage: DEFAULT

Type: STENCILBUFFER [0x22f3ce4]
1280x768
FORMAT_D24_UNORM_S8_UINT
Mips: 1
Usage: DEFAULT

Powerful Texture Visualization



FPS: 107.9 [20 frame avg]   Tris/Frame: 88406   Speed: -:-

Shaders Used [Mod]   RS Used [Mod]   Scissor All [ ]   Ignore DC [ ]   Wire [ ]   Depth [ ]

State buckets:

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

| Calls | Prims | GPU Time v | CPU Time | Pixels | Perf Marker |
|---|---|---|---|---|---|
| 32 | 39780 | 0.341 ms | 2.630 ms | 818080 | 2: Scene to ordinary backbuffer |
| 32 | 39780 | 0.333 ms | 2.887 ms | 820128 | 2: Scene to ordinary backbuffer |
| 2 | 8844 | 0.211 ms | 0.021 ms | 582656 | ^^ Multiples ^^ |
| 1 | 2 | 0.158 ms | 0.015 ms | 786432 | 2: Fullscreen quad for fog thickn |

State Bucket Subtotals: Calls = 32, GPU Time = 0.341 ms, CPU Time = 2.630 ms, Pixels = 818080

Draw calls in selected State Bucket:

| Ord | Prims | GPU Time v | CPU Time | Pixels | Perf Marker |
|---|---|---|---|---|---|
| 16 | 480 | 0.060 ms | 0.081 ms | 582736 | 2: Scene to ordinary backbuf |
| 6 | 4864 | 0.055 ms | 0.088 ms | 0 | 2: Scene to ordinary backbuf |
| 14 | 5670 | 0.027 ms | 0.090 ms | 0 | 2: Scene to ordinary backbuf |
| 17 | 96 | 0.020 ms | 0.086 ms | 172272 | 2: Scene to ordinary backbuf |
| 18 | 3140 | 0.019 ms | 0.085 ms | 3040 | 2: Scene to ordinary backbuf |
| 19 | 3140 | 0.016 ms | 0.085 ms | 0 | 2: Scene to ordinary backbuf |
| 26 | 3140 | 0.016 ms | 0.084 ms | 0 | 2: Scene to ordinary backbuf |

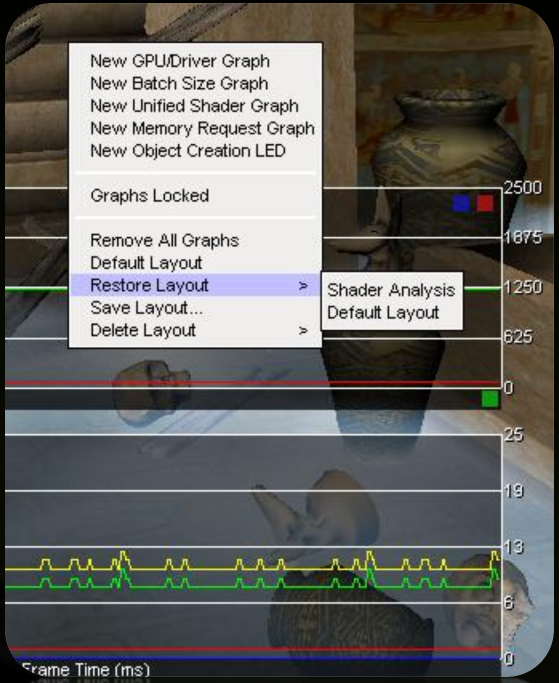
Draw Call Subtotals: Primitives = 4864, GPU Time = 0.055 ms, CPU Time = 0.088 ms, Pixels = 0

Frame 1/1, Rendering Event 7/72, Overall Event 91/575

Combined CPU/GPU Timing Graph

# Tons of other improvements!

- **Multiple, named Performance Dashboard Layouts**
  - Save separate layouts for each of your monitoring scenarios.
- **Sampler State Editing**
  - Edit  and override any property of a sampler in your application
- **New Real-time Experiments**
  - Minimize Geometry (Replaces all geometry with a single primitive)
- **Improved Compatibility and Stability**
  - Rewritten interception layer requires less cooperation from the application.
  - Now compatible with applications that include frame-limiting logic.
  - Now compatible with applications which send non-deterministic graphics workloads.
  - Many bug fixes and performance improvements.
- **Improvements to Frame Profiler**
  - New CPU/GPU Timings Graph lets you directly see and compare utilization in the CPU, the driver, and the GPU.
- **User Interface Improvements**
  - Navigation hotkeys for every major tool
- **Compatibility, stability, and reliability improvements**
  - Extensive testing on a wide range of applications
  - Minor bug fixes



# How to Think of PerfHUD's Main Modes

- **Performance Dashboard**
  Real-time Holistic Analysis

- **Frame Debugger**
  Rendering Debugging

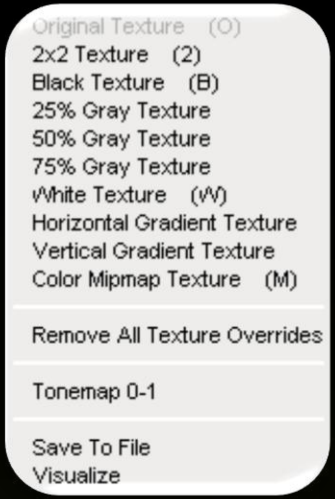
- **Advanced Screens**
  Shader and Render State Inspection and Modification

- **Frame Profiler**
  Automated Bottleneck Detection

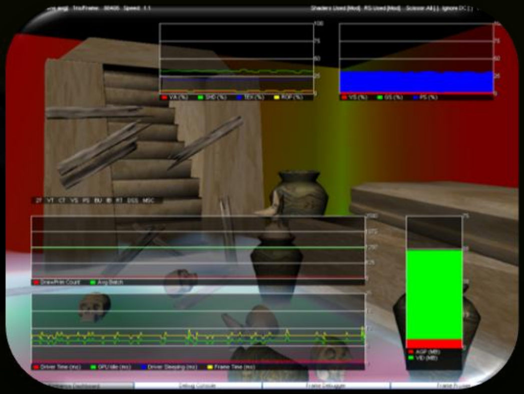
  Per-Draw Call Performance Graphs

# *New!* Advanced Texture Overrides

- Override any texture in your application with a variety of useful texture overrides
- 2x2 Texture: Reduces texture bandwidth usage by using the smallest texture possible.
- Black, 25% Gray, 50% Gray, 75% Gray, White, Horizontal gradient, Vertical Gradient: Each of these can be useful as debug input to your shaders.
- Color Mipmap Texture: Visualize your mip levels quickly and easily
- Use with shader edit and continue to quickly diagnose and correct bugs.



Texture Context Menu



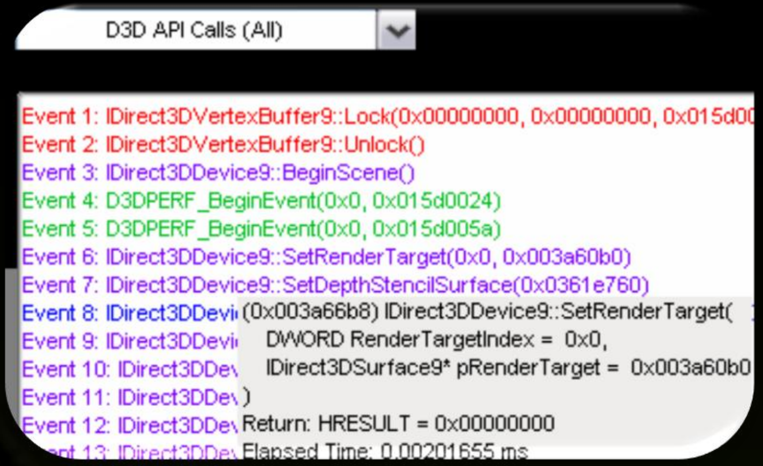
Mipmap Visualization

# *New!* API Call List and Perf Event View

## *New!* API Call List

A full list of all D3D API calls, colorized by call type

One click jumps you to the corresponding draw call.



D3D API Calls (All)

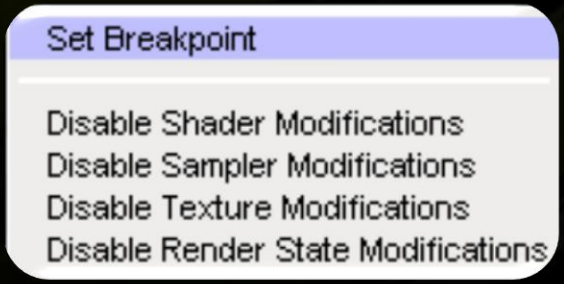
```
Event 1: IDirect3DVertexBuffer9::Lock(0x00000000, 0x00000000, 0x015d00
Event 2: IDirect3DVertexBuffer9::Unlock()
Event 3: IDirect3DDevice9::BeginScene()
Event 4: D3DPERF_BeginEvent(0x0, 0x015d0024)
Event 5: D3DPERF_BeginEvent(0x0, 0x015d005a)
Event 6: IDirect3DDevice9::SetRenderTarget(0x0, 0x003a60b0)
Event 7: IDirect3DDevice9::SetDepthStencilSurface(0x0361e760)
Event 8: IDirect3DDevi (0x003a66b8) IDirect3DDevice9::SetRenderTarget(
Event 9: IDirect3DDevi     DWORD RenderTargetIndex =  0x0,
Event 10: IDirect3DDev     IDirect3DSurface9* pRenderTarget =  0x003a60b0
Event 11: IDirect3DDev )
Event 12: IDirect3DDev Return: HRESULT = 0x00000000
ent 13: IDirect3DDev Elapsed Time: 0.00201655 ms
```

## *New!* Perf Event List

Navigate your scene using D3D Perf Events to annotate sections of the frame.

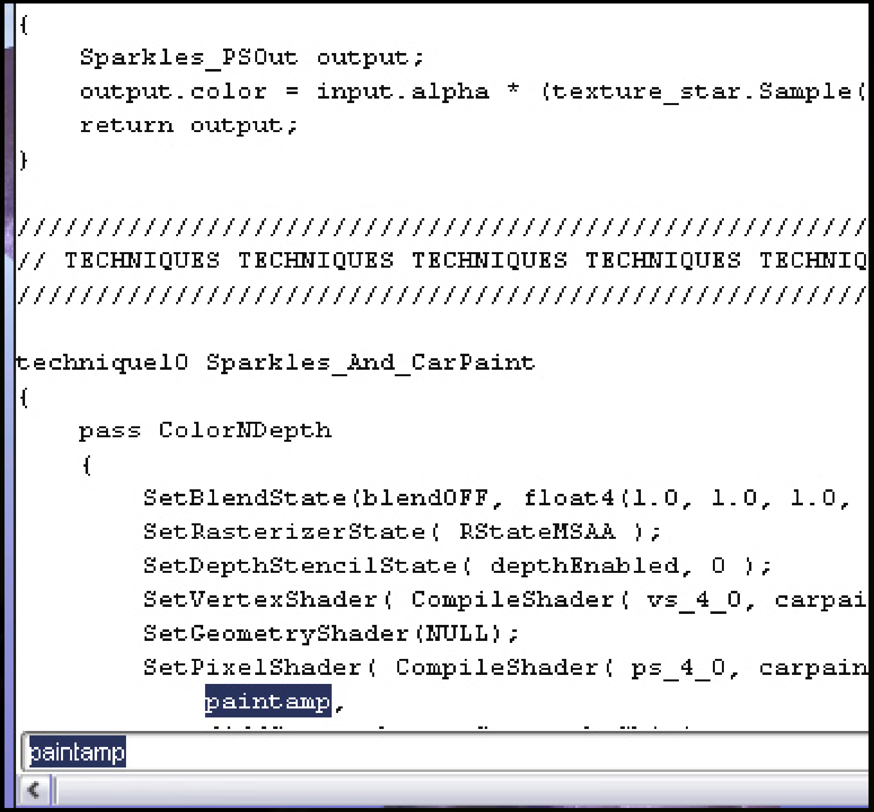
Use Perf Events to selectively disable state overrides for portions of the scene.

Set a debug break (_int 3) to occur on a Perf Event to help debug your CPU code.


Set Breakpoint
Disable Shader Modifications
Disable Sampler Modifications
Disable Texture Modifications
Disable Render State Modifications

# Shader Edit and Continue

- Edit & Continue for:
  - DirectX 9 HLSL and .fx
  - DirectX 10 HLSL and .fx
  - Vertex, geometry, and pixel shaders
- Code editing
  - Standard keyboard and mouse interaction
  - Search bar
- Easily toggle between original and modified shaders
  - Discard Current Edits
  - Restore Original Shader
  - Controlled via hotkey



```
{
    Sparkles_PSOut output;
    output.color = input.alpha * (texture_star.Sample(
    return output;
}

/////////////////////////////////////////////////////////
// TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQ
/////////////////////////////////////////////////////////

technique10 Sparkles_And_CarPaint
{
    pass ColorNDepth
    {
        SetBlendState(blendOFF, float4(1.0, 1.0, 1.0,
        SetRasterizerState( RStateMSAA );
        SetDepthStencilState( depthEnabled, 0 );
        SetVertexShader( CompileShader( vs_4_0, carpai
        SetGeometryShader(NULL);
        SetPixelShader( CompileShader( ps_4_0, carpain
            paintamp,
```

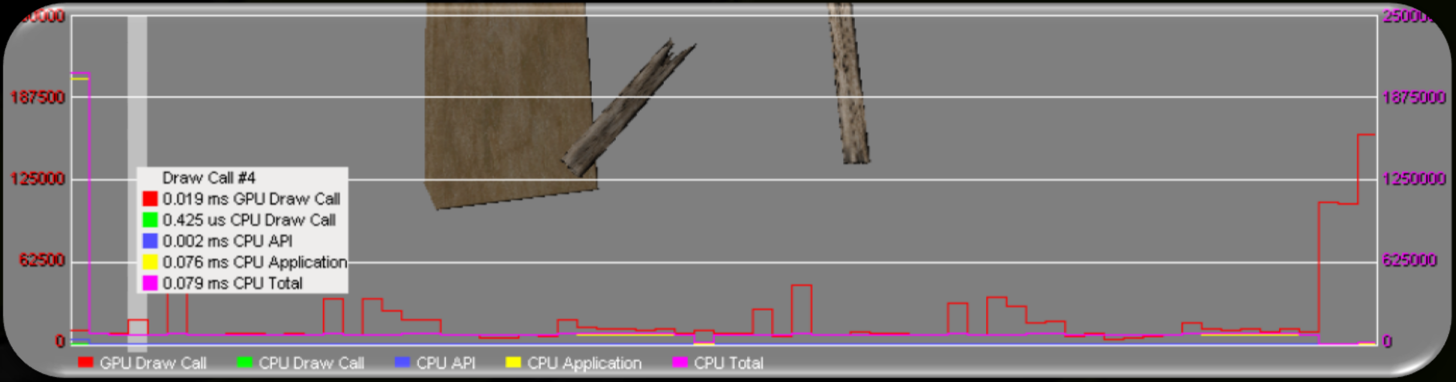
paintamp

Editing Shaders

# The Frame Profiler

- Same powerful automated analysis as in PerfHUD 5

- *New!* CPU/GPU timing graph



Draw Call #4

🟥 0.019 ms GPU Draw Call
🟩 0.425 us CPU Draw Call
🟦 0.002 ms CPU API
🟨 0.076 ms CPU Application
🟪 0.079 ms CPU Total

Detailed CPU/GPU Timings

# How is PerfHUD Different from Other Performance Tools?

- It works in real-time on your application.  Other tools require out of context, offline analysis.  PerfHUD allows you to debug and tune your application in the most natural place: within your application!

- One key press provides a list of draw calls (grouped by bottleneck and sorted by duration) to work on.  Running experiments and collecting data from individual pipeline units is difficult and easy to misjudge.  PerfHUD automatically tells you exactly where your bottlenecks are so you can fix them quickly.

- Real-time frame scrubbing.  Modern engines have thousands of draw calls per frame.  PerfHUD lets you decompose the scene, stepping through each draw call to find any problems.

- Edit-and-continue.  Modifying shader code and render states can be time consuming.  PerfHUD allows you to make changes while your application is running – allowing you to quickly try our ideas and get immediate feedback.

# Trying Out PerfHUD 6

- Read Chapter 1 of the User's Guide
  - This will guide you through the key new features quickly
- Please share your feedback:

  PerfHUD@nvidia.com
  developer.nvidia.com/forums