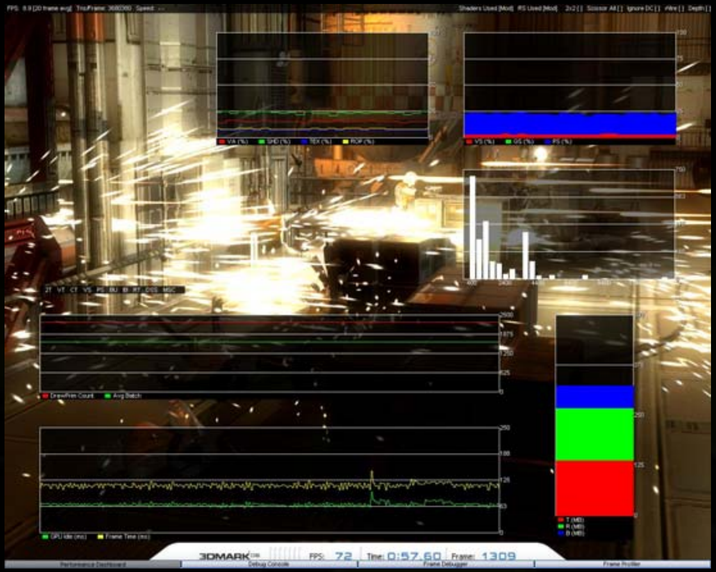
NVIDIA PerfKit 5

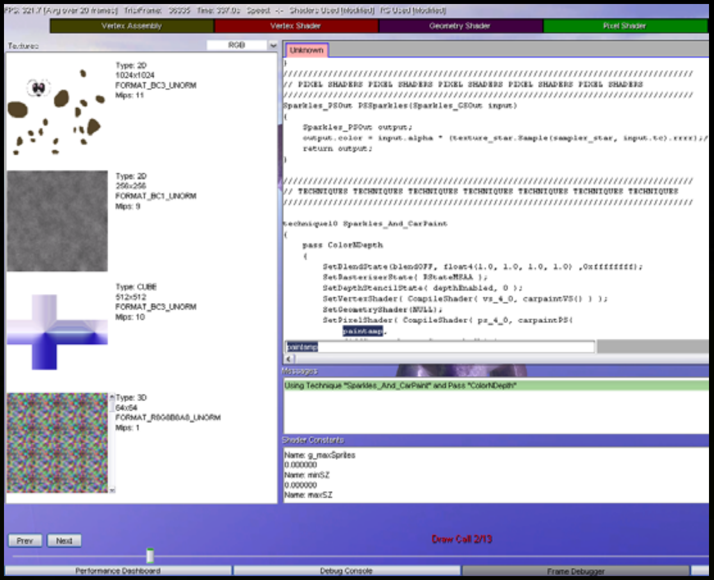
NVIDIA PerfKit 5: The World's Most Advanced GPU Performance Suite

- PerfHUD 5
  - *New!* DirectX 10 Support, GeForce 8800 GTX Support
  - *New!* Microsoft Windows Vista Support (DirectX 9 and 10)
  - *New!* Interactive usage model
    - *New!* Shader edit-and-continue
    - *New!* Render state edit-and-continue
  - *New!* Revamped customizable user interface
  - *New!* Numerous other improvements…
  - Real-time performance analysis and debugging
  - Automated bottleneck determination
- PerfSDK
  - *New!* Microsoft Windows Vista and GeForce 8800 GTX 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
- gDEBugger 30-day trial version
  - Includes PerfSDK and GLExpert integration

PerfHUD 5



Customizable Performance Dashboard



Type: 2D
1024x1024
FORMAT_BC3_UNORM
Mips: 11

Type: 2D
256x256
FORMAT_BC1_UNORM
Mips: 9

Type: CUBE
512x512
FORMAT_BC3_UNORM
Mips: 10

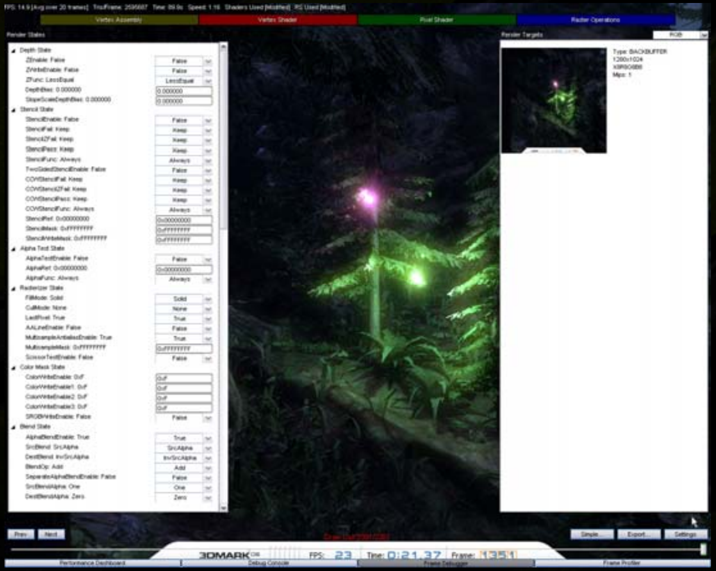
Type: 3D
64x64
FORMAT_R8G8B8A8_UNORM
Mips: 1

```
}
//////////////////////////////////////////////////////////////////////////////
// PIXEL SHADERS PIXEL SHADERS PIXEL SHADERS PIXEL SHADERS PIXEL SHADERS
//////////////////////////////////////////////////////////////////////////////
Sparkles_PSOut PSSparkles(Sparkles_GSOut input)
{
    Sparkles_PSOut output;
    output.color = input.alpha * (texture_star.Sample(sampler_star, input.tc).rrrr);/
    return output;
}

//////////////////////////////////////////////////////////////////////////////
// TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQUES TECHNIQUES
//////////////////////////////////////////////////////////////////////////////

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

Using Technique "Sparkles_And_CarPaint" and Pass "ColorNDepth"

Name: g_maxSprites
0.000000
Name: minSZ
0.000000
Name: maxSZ

Shader Edit-and-Continue

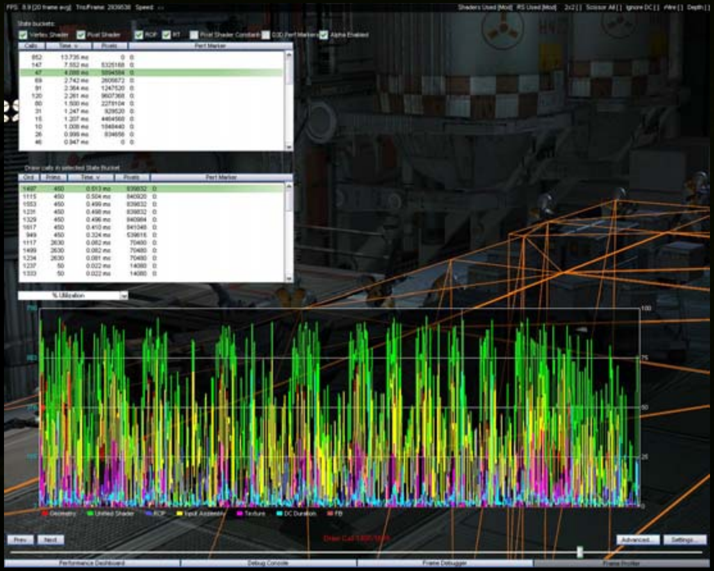


FPS: 14.9 [Avg over 20 frames] TrueFrame: 2595687 Time: 89.9s Speed: 1.16 Shaders Used [Modified] RS Used [Modified]

- Depth State
  - ZEnable: False — False
  - ZWriteEnable: False — False
  - ZFunc: LessEqual — LessEqual
  - DepthBias: 0.000000 — 0.000000
  - SlopeScaleDepthBias: 0.000000 — 0.000000
- Stencil State
  - StencilEnable: False — False
  - StencilFail: Keep — Keep
  - StencilZFail: Keep — Keep
  - StencilPass: Keep — Keep
  - StencilFunc: Always — Always
  - TwoSidedStencilEnable: False — False
  - CCWStencilFail: Keep — Keep
  - CCWStencilZFail: Keep — Keep
  - CCWStencilPass: Keep — Keep
  - CCWStencilFunc: Always — Always
  - StencilRef: 0x00000000 — 0x00000000
  - StencilMask: 0xFFFFFFFF — 0xFFFFFFFF
  - StencilWriteMask: 0xFFFFFFFF — 0xFFFFFFFF
- Alpha Test State
  - AlphaTestEnable: False — False
  - AlphaRef: 0x00000000 — 0x00000000
  - AlphaFunc: Always — Always
- Rasterizer State
  - FillMode: Solid — Solid
  - CullMode: None — None
  - LastPixel: True — True
  - AALineEnable: False — False
  - MultisampleAntialiasEnable: True — True
  - MultisampleMask: 0xFFFFFFFF — 0xFFFFFFFF
  - ScissorTestEnable: False — False
- Color Mask State
  - ColorWriteEnable: 0xF — 0xF
  - ColorWriteEnable1: 0xF — 0xF
  - ColorWriteEnable2: 0xF — 0xF
  - ColorWriteEnable3: 0xF — 0xF
  - SRGBWriteEnable: False — False
- Blend State
  - AlphaBlendEnable: True — True
  - SrcBlend: SrcAlpha — SrcAlpha
  - DestBlend: InvSrcAlpha — InvSrcAlpha
  - BlendOp: Add — Add
  - SeparateAlphaBlendEnable: False — False
  - SrcBlendAlpha: One — One
  - DestBlendAlpha: Zero — Zero

Type: BACKBUFFER
1280x1024
X8R8G8B8
Mips: 1

Render State Edit-and-Continue



FPS: 8.9 [20 frame avg] TrueFrame: 2839538 Speed: --

State buckets:

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

Calls	Time	Pixels	Perf Marker
852	13.735 ms	0	0
147	7.552 ms	5325168	0
47	4.288 ms	5094584	0
69	2.742 ms	2606672	0
91	2.364 ms	1247520	0
120	2.261 ms	9607368	0
60	1.500 ms	2278104	0
31	1.247 ms	929520	0
15	1.207 ms	4464560	0
10	1.008 ms	1048440	0
26	0.998 ms	834658	0
46	0.947 ms	0	0

Draw calls in selected State Bucket

Ord	Prims	Time	Pixels	Perf Marker
1497	450	0.513 ms	839832	0
1115	450	0.504 ms	840920	0
1553	450	0.499 ms	839832	0
1231	450	0.498 ms	839832	0
1329	450	0.496 ms	840984	0
1617	450	0.410 ms	841048	0
949	450	0.324 ms	539616	0
1117	2630	0.082 ms	70400	0
1499	2630	0.082 ms	70400	0
1234	2630	0.081 ms	70400	0
1237	50	0.022 ms	14080	0
1333	50	0.022 ms	14080	0

Enhanced Frame Profiler

“Numerous Other Improvements”? Yes, plenty!

- **G80 support on Windows Vista and XP**
- **DirectX 10 support on Windows Vista**
- **DirectX 9 support on Windows XP and Windows Vista**
- **Edit & Continue for HLSL and .fx vertex, geometry, and pixel shaders**
- **Edit & Continue for Raster Operations state**
- **Customizable Performance Dashboard**
  - Choose up to 4 counters per graph
  - Full set of 40+ PerfSDK Direct3D and GPU counters available
  - Arrange graphs arbitrarily
  - Create and delete graphs
  - Save/load custom layouts
  - Layout stored automatically when exiting PerfHUD
  - Double-clicking color swatch in graph legend toggles display of that channel
  - New Batch Histogram with batch bucket disabling
  - New experiments (minimize geometry, highlight ps_4_0 shaders)
- **Improvements to Frame Debugger**
  - Visualization of 1D, 2D, 3D, rectangle textures, shadow maps, texture arrays, and cube maps
  - Arbitrarily rotate wireframe visualization
  - Show selected draw call only (versus selected and all previous calls)
  - Mouseover on textures and render targets shows texture coordinates and texel color swatch
- **Improvements to Frame Profiler**
  - Instruction Count Ratio graphs
  - Tooltips for graphs with graph values
  - "Alpha Enabled" state bucket criteria
  - Support for Hierarchical Direct3D Performance Markers
- **Improved user interface**
  - Polished look-and-feel with new fonts and widgets
  - Hardware mouse cursor improves responsiveness when frame rate is low
  - Clicking on graphs in Frame Profiler jumps to corresponding draw call
  - Clear graph legends
  - New Help screen with software version, GPU, driver information, and keyboard shortcuts
  - Options screen with numerous choices, including:
    - Clear color buffer when viewing Z-only passes
    - Preserve backbuffer
    - Draw call visualization mode
  - F2 hides/shows PerfHUD UI
- **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
  Holistic Analysis

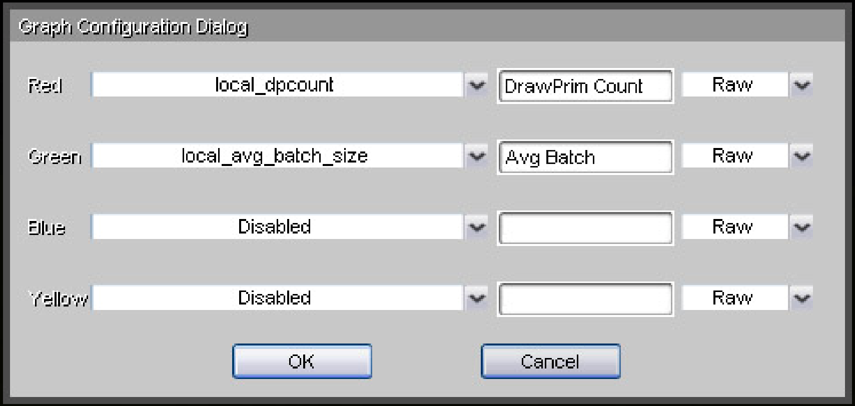
- Frame Debugger
  Rendering Debugging

- Advanced Screens
  Shader and Render State Inspection and Modification

- Frame Profiler
  Automated Bottleneck Determination

New! Revamped Performance Dashboard

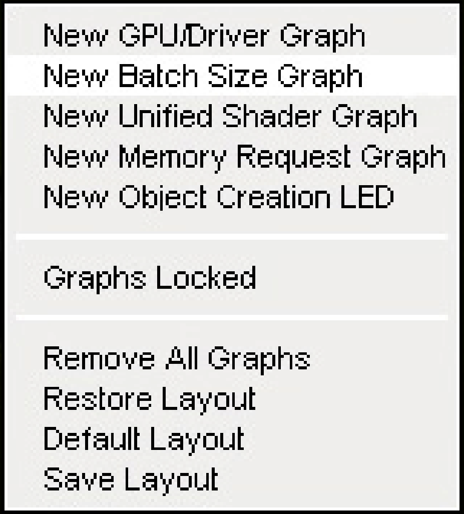
- Choose up to 4 counters per graph
- Pick your own labels
- Full set of 40+ PerfSDK Direct3D and GPU counters available
- Arrange graphs as you choose
- Create and delete graphs
- Save/load custom layouts
- Layout stored automatically when exiting PerfHUD
- Double-clicking color swatch in graph legend toggles display of that channel
- Cleaner user interface
- New Batch Histogram with batch bucket disabling
- New experiments (minimize geometry, highlight ps_4_0 shaders)



Graph Configuration Dialog

Red	local_dpcount	DrawPrim Count	Raw
Green	local_avg_batch_size	Avg Batch	Raw
Blue	Disabled		Raw
Yellow	Disabled		Raw

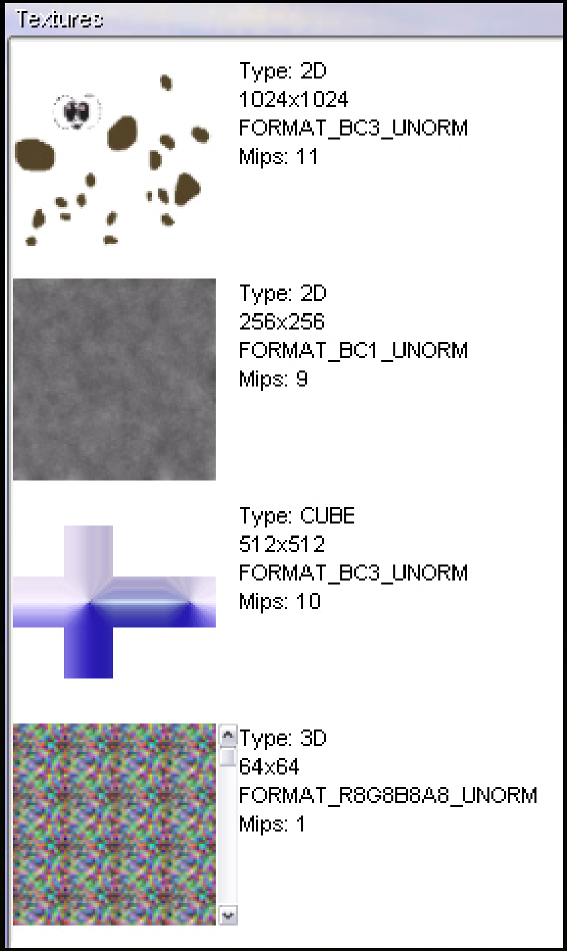
Graph Configuration Dialog



Layout Configuration Menu

New! Improved Frame Debugger

- Visualization of 1D, 2D, 3D, rectangle textures, shadow maps, cube maps, and texture arrays
- User can arbitrarily rotate wireframe visualization
- Show selected draw call only (versus selected and all previous calls)
- Texture and render target tooltips show texture coordinates, texel color, and color swatch
- Clearer draw call display during frame scrubbing


Textures

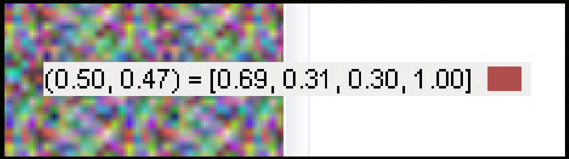
Type: 2D
1024x1024
FORMAT_BC3_UNORM
Mips: 11

Type: 2D
256x256
FORMAT_BC1_UNORM
Mips: 9

Type: CUBE
512x512
FORMAT_BC3_UNORM
Mips: 10

Type: 3D
64x64
FORMAT_R8G8B8A8_UNORM
Mips: 1

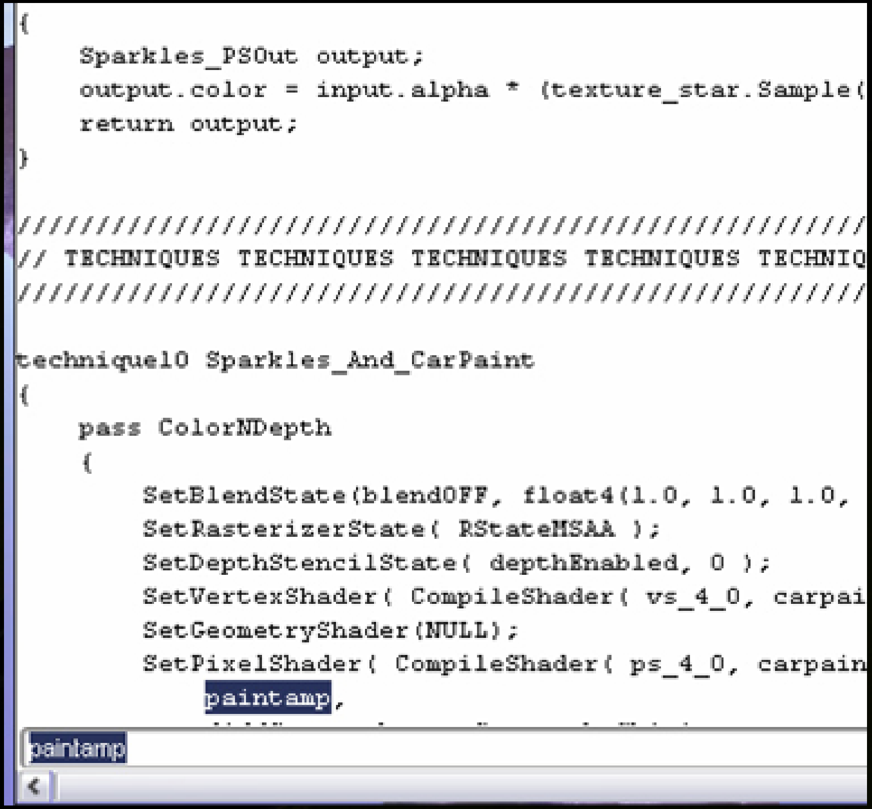
Support for All Texture Formats


(0.50, 0.47) = [0.69, 0.31, 0.30, 1.00] 🟥

Texture and Render Target Tooltips

New! 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 original and modified shaders
  - Discard Current Edits
  - Restore Original Shader
  - Controlled via hotkey



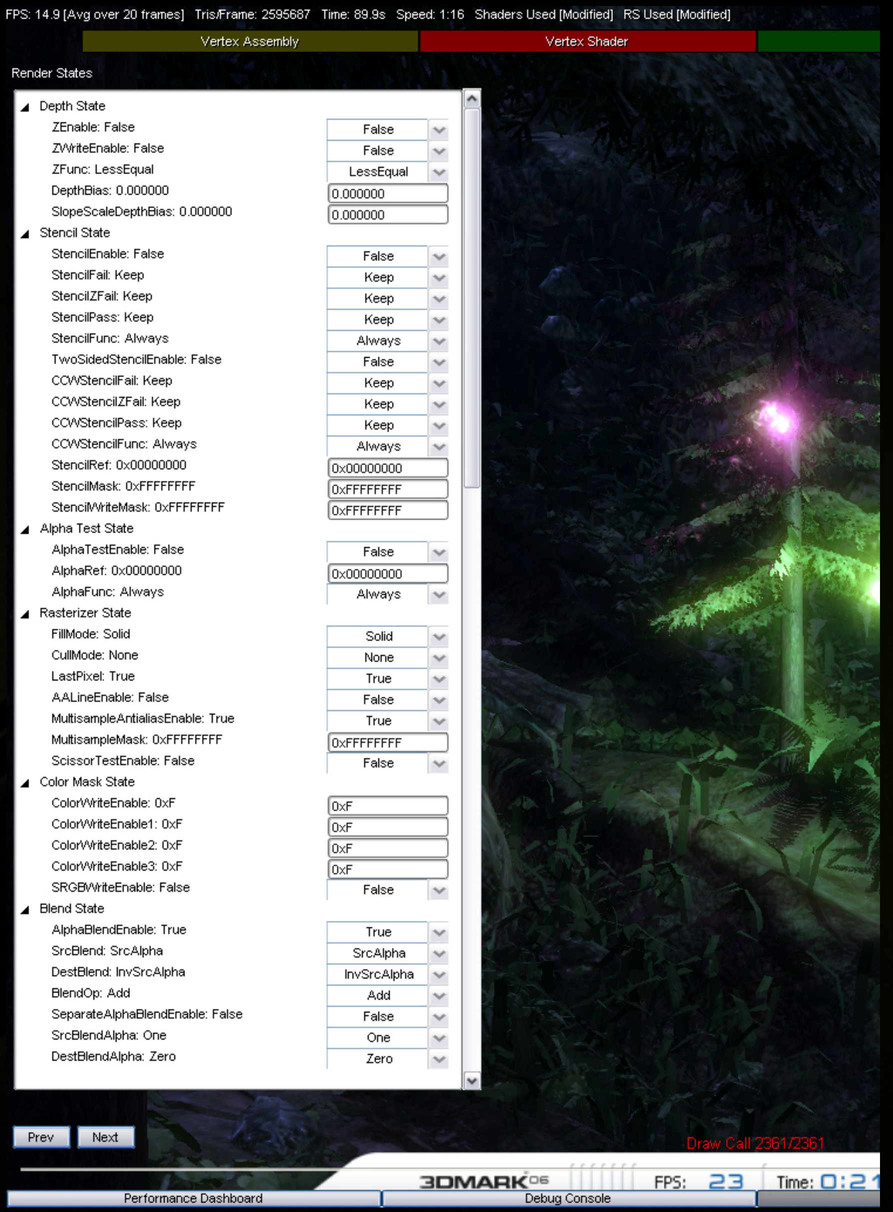
```
{
    Sparkles_PSOut output;
    output.color = input.alpha * (texture_star.Sample(
    return output;
}

////////////////////////////////////////////////////////////
// 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

New! Render State Edit-and-Continue

- View and modify render states
- Easily toggle original and modified states
  - By state group
  - By all states
  - Controlled via hotkeys



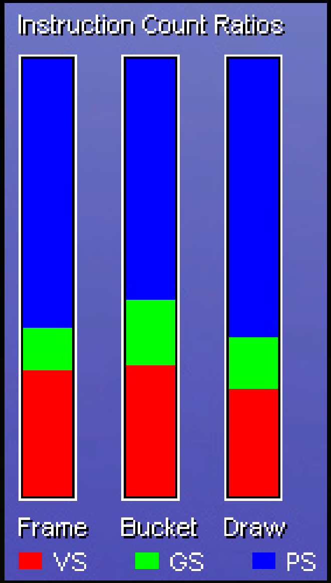
FPS: 14.9 [Avg over 20 frames] Tris/Frame: 2595687 Time: 89.9s Speed: 1:16 Shaders Used [Modified] RS Used [Modified]

Render States

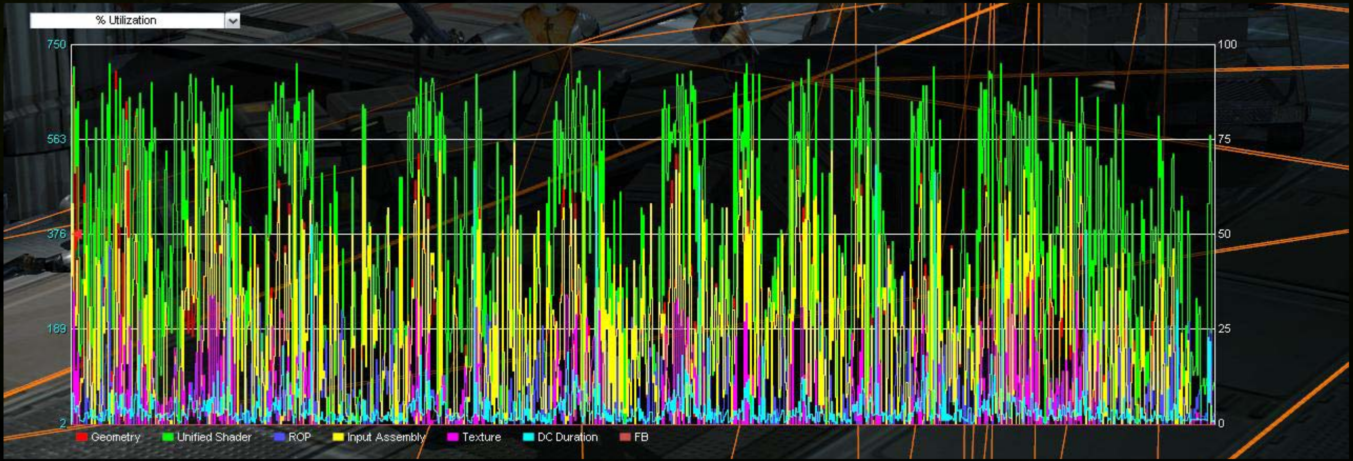
Depth State	
ZEnable: False	False
ZWriteEnable: False	False
ZFunc: LessEqual	LessEqual
DepthBias: 0.000000	0.000000
SlopeScaleDepthBias: 0.000000	0.000000
Stencil State	
StencilEnable: False	False
StencilFail: Keep	Keep
StencilZFail: Keep	Keep
StencilPass: Keep	Keep
StencilFunc: Always	Always
TwoSidedStencilEnable: False	False
CCWStencilFail: Keep	Keep
CCWStencilZFail: Keep	Keep
CCWStencilPass: Keep	Keep
CCWStencilFunc: Always	Always
StencilRef: 0x00000000	0x00000000
StencilMask: 0xFFFFFFFF	0xFFFFFFFF
StencilWriteMask: 0xFFFFFFFF	0xFFFFFFFF
Alpha Test State	
AlphaTestEnable: False	False
AlphaRef: 0x00000000	0x00000000
AlphaFunc: Always	Always
Rasterizer State	
FillMode: Solid	Solid
CullMode: None	None
LastPixel: True	True
AALineEnable: False	False
MultisampleAntialiasEnable: True	True
MultisampleMask: 0xFFFFFFFF	0xFFFFFFFF
ScissorTestEnable: False	False
Color Mask State	
ColorWriteEnable: 0xF	0xF
ColorWriteEnable1: 0xF	0xF
ColorWriteEnable2: 0xF	0xF
ColorWriteEnable3: 0xF	0xF
SRGBWriteEnable: False	False
Blend State	
AlphaBlendEnable: True	True
SrcBlend: SrcAlpha	SrcAlpha
DestBlend: InvSrcAlpha	InvSrcAlpha
BlendOp: Add	Add
SeparateAlphaBlendEnable: False	False
SrcBlendAlpha: One	One
DestBlendAlpha: Zero	Zero

New! Frame Profiler Improvements

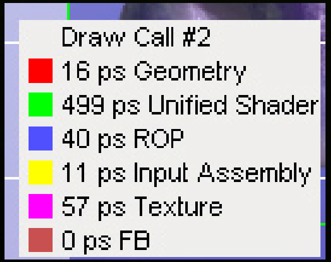
- Instruction Count Ratio graphs
- Tooltips for graphs with graph values
- “Alpha Enabled” state bucket criteria
- Support for Hierarchical Direct3D Performance Markers
- Same informative graphs as before…



Instruction Count Ratios



More Detailed Per-Unit Information


Draw Call #2

🟥 16 ps Geometry
🟩 499 ps Unified Shader
🟦 40 ps ROP
🟨 11 ps Input Assembly
🟪 57 ps Texture
🟥 0 ps FB

Graph Tooltips

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 5

- Read the Quick Tutorial
  - This will guide you through the key new features quickly

- Please share your feedback:
  developer.nvidia.com/forums