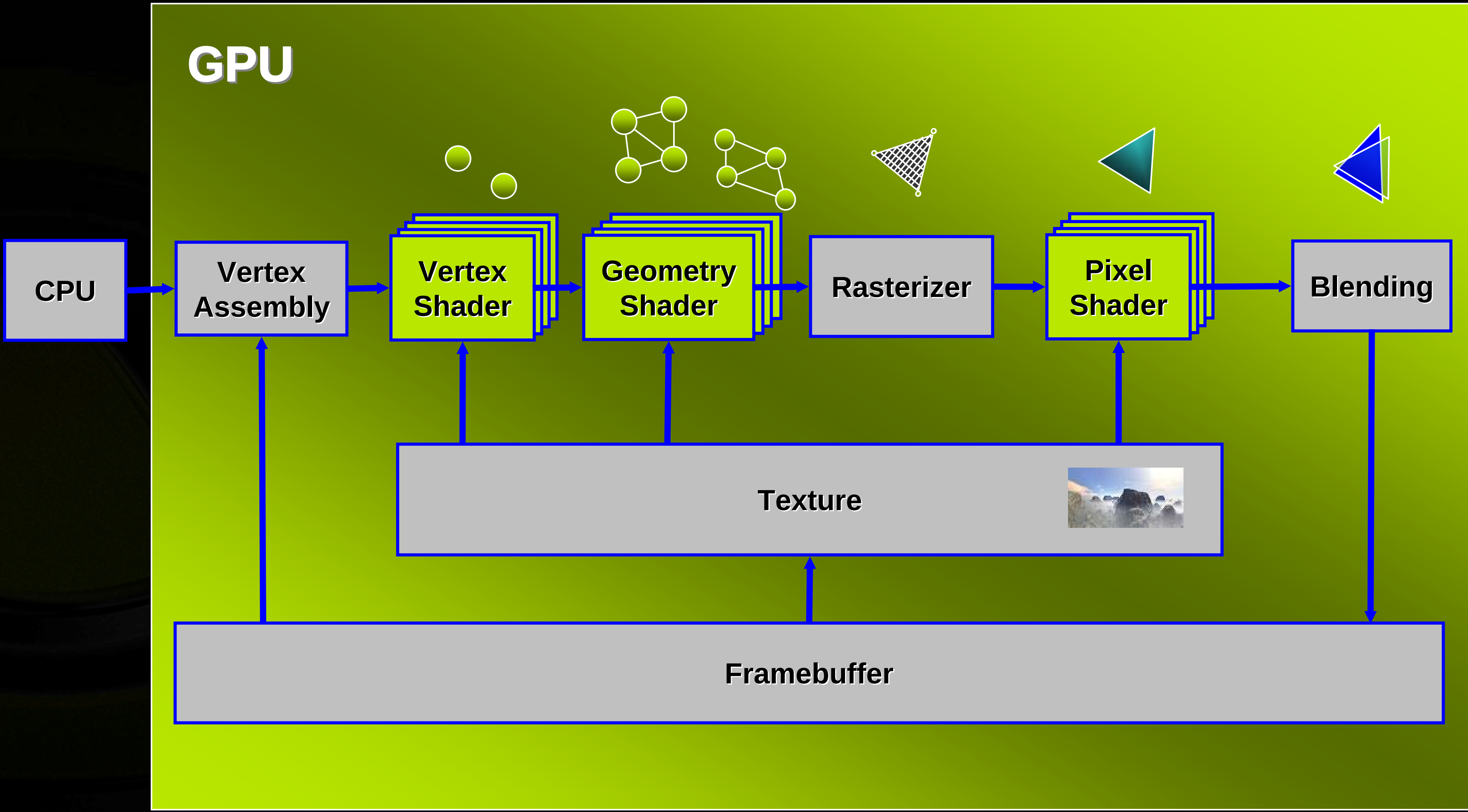
Performance Tools

Jeff Kiel, NVIDIA Corporation

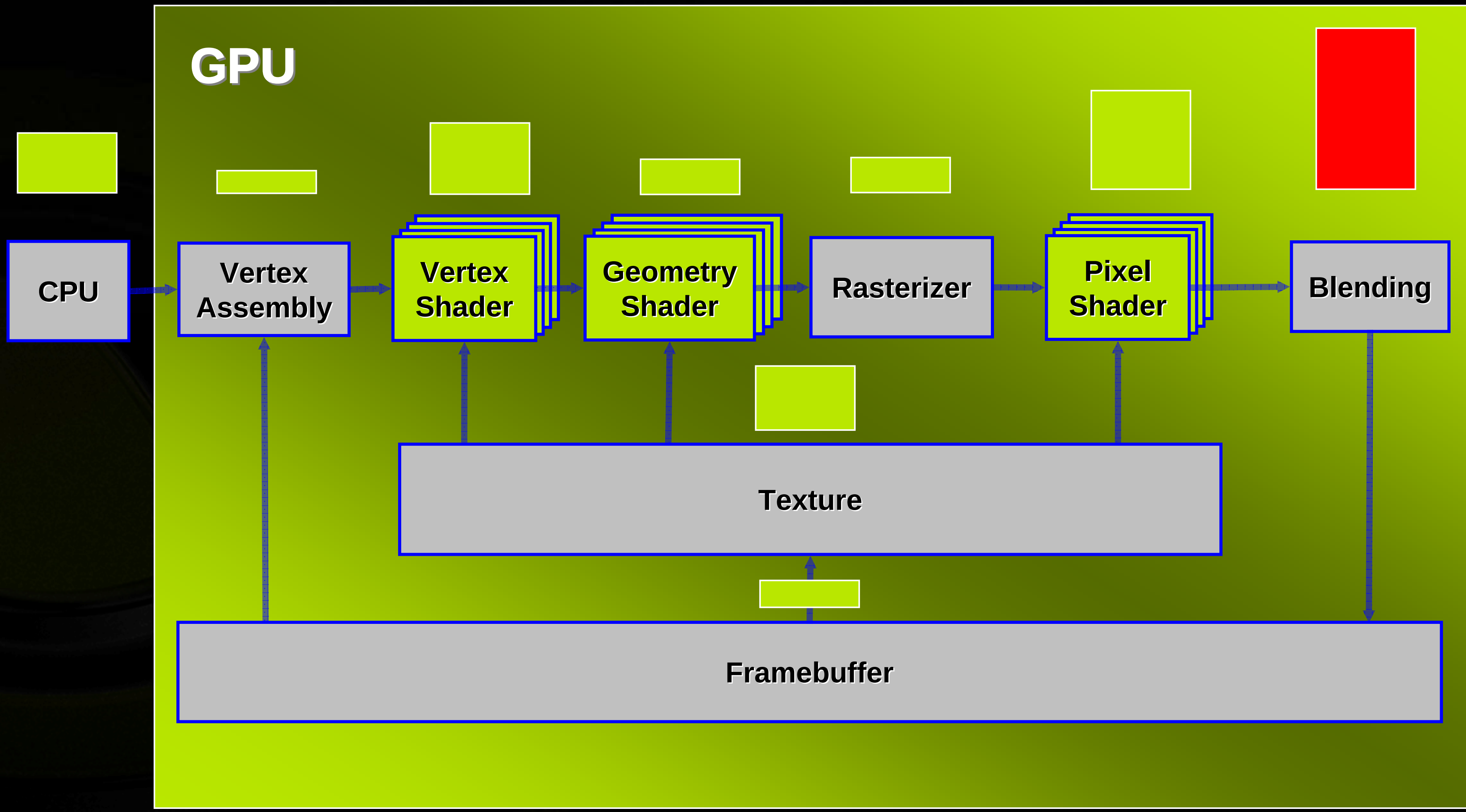
Performance Tools Agenda

- Overview of GPU pipeline and Unified Shader
- NVIDIA PerfKit 5.0: Driver & GPU Performance Data
  - PerfHUD: The GPU Performance Accelerator
  - PerfSDK: Performance data integrated into your application
- ShaderPerf: Shader program performance

GPU Pipelined Architecture (Logical View)



GPU Pipelined Architecture (Logical View)



Common Problems

- New, more complex GPU hardware
  - GPU is a black box
  - Unified shaders changes everything
- Increasing engine and scene complexity
- Artists don’t always understand how rendering engines work
- CPU tuning only gets you so far
- Turn around time for debugging and tuning shaders is too long
- Hard to debug setup issues

PerfHUD solves issues for top developers!

In a recent survey, over 100 PerfHUD 4 users reported an average speedup of 35%, and as much as 400% in some cases!

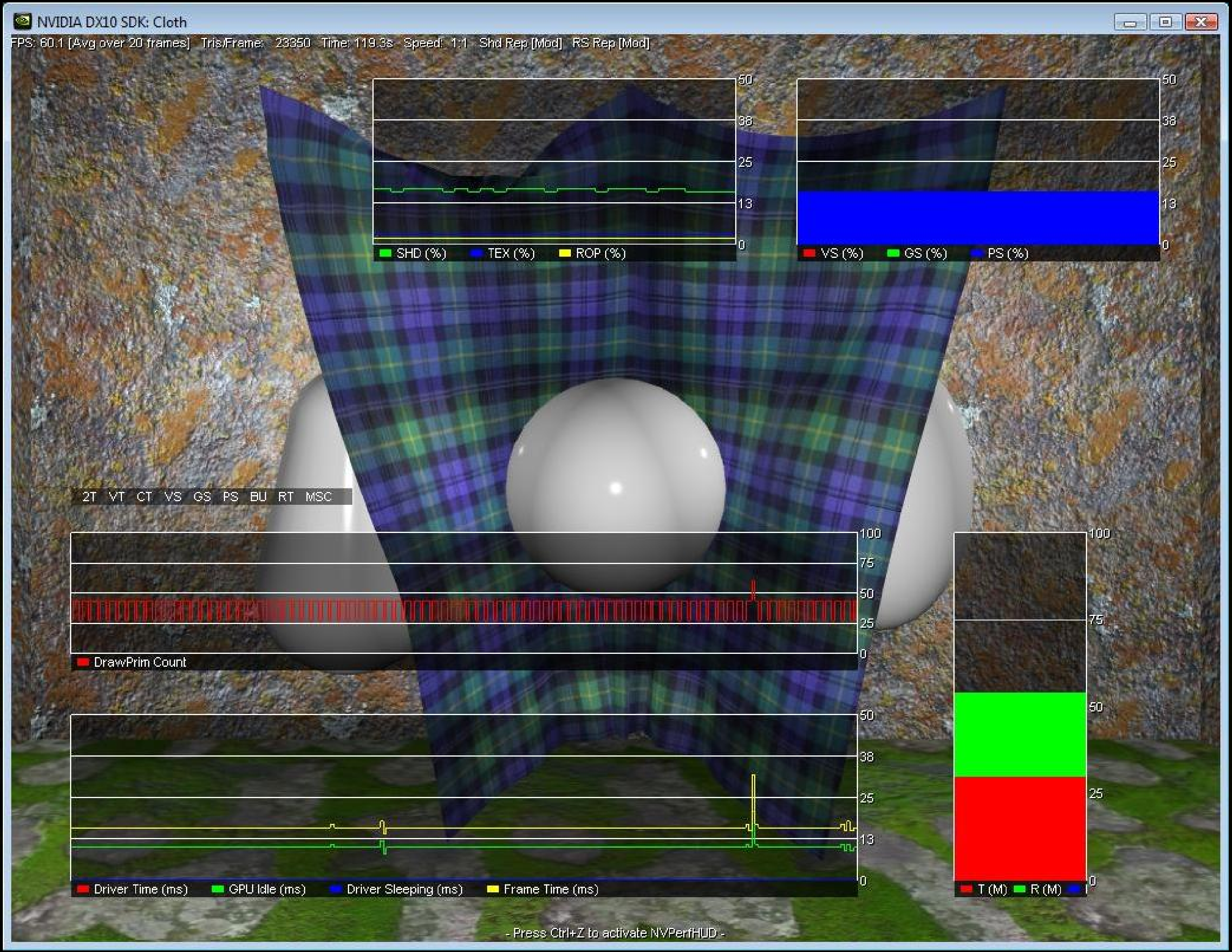
PerfHUD: Direct3D debugging and tuning

- One click bottleneck determination
- Graphs and debugging tools overlaid on your application
- 4 screens for targeted analysis
  - Performance Dashboard
  - Debug Console
  - Frame Debugger
  - Frame Profiler
- Drag and drop application on PerfHUD icon

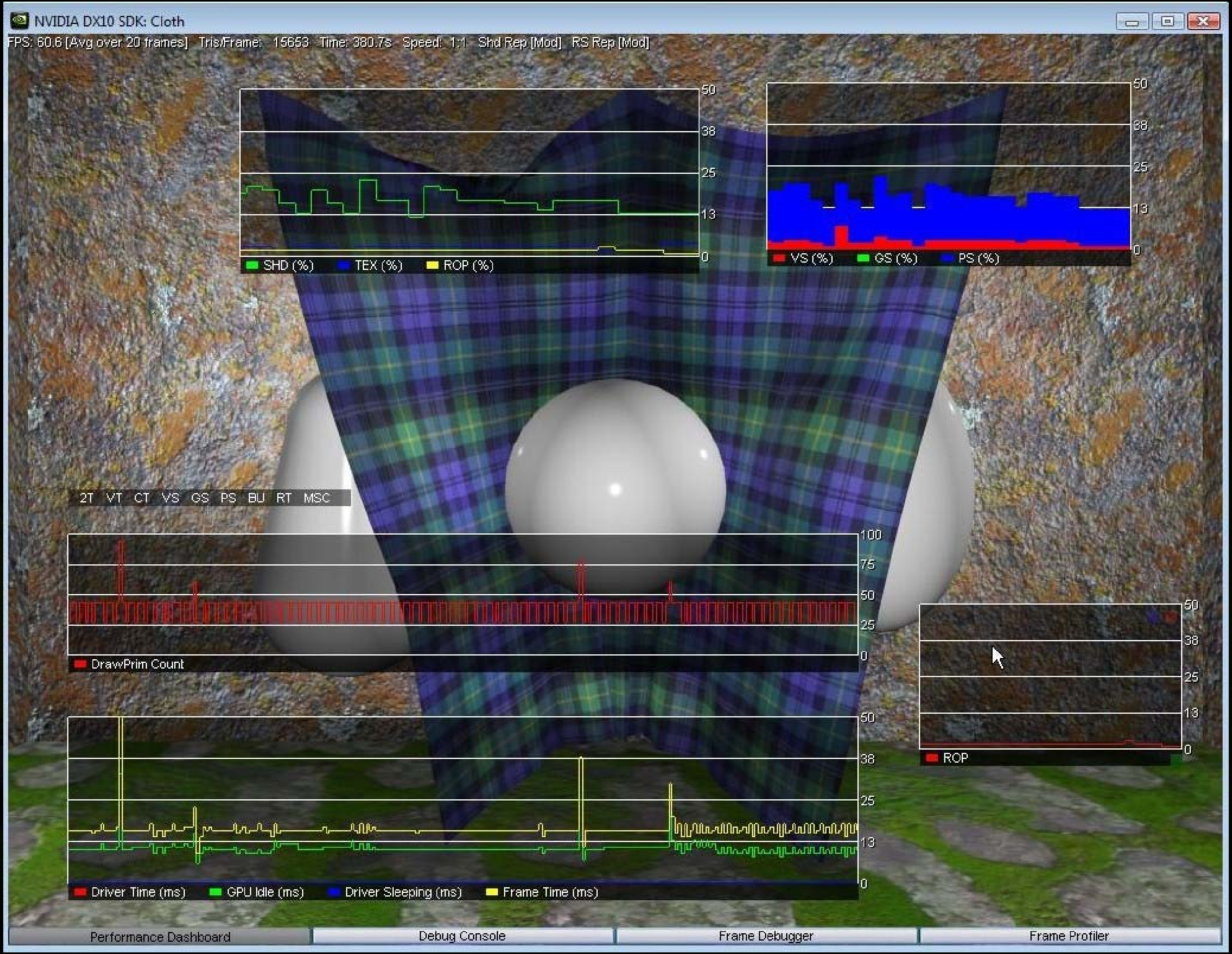
New! PerfHUD 5.0!

- Interactive model
  - Shader Edit and Continue
  - Render state Modification
  - Configurable Graphs
  - Many more features and usability improvements
- New technologies
  - Windows Vista & DirectX 10
  - GeForce 8800 and Unified Shaders

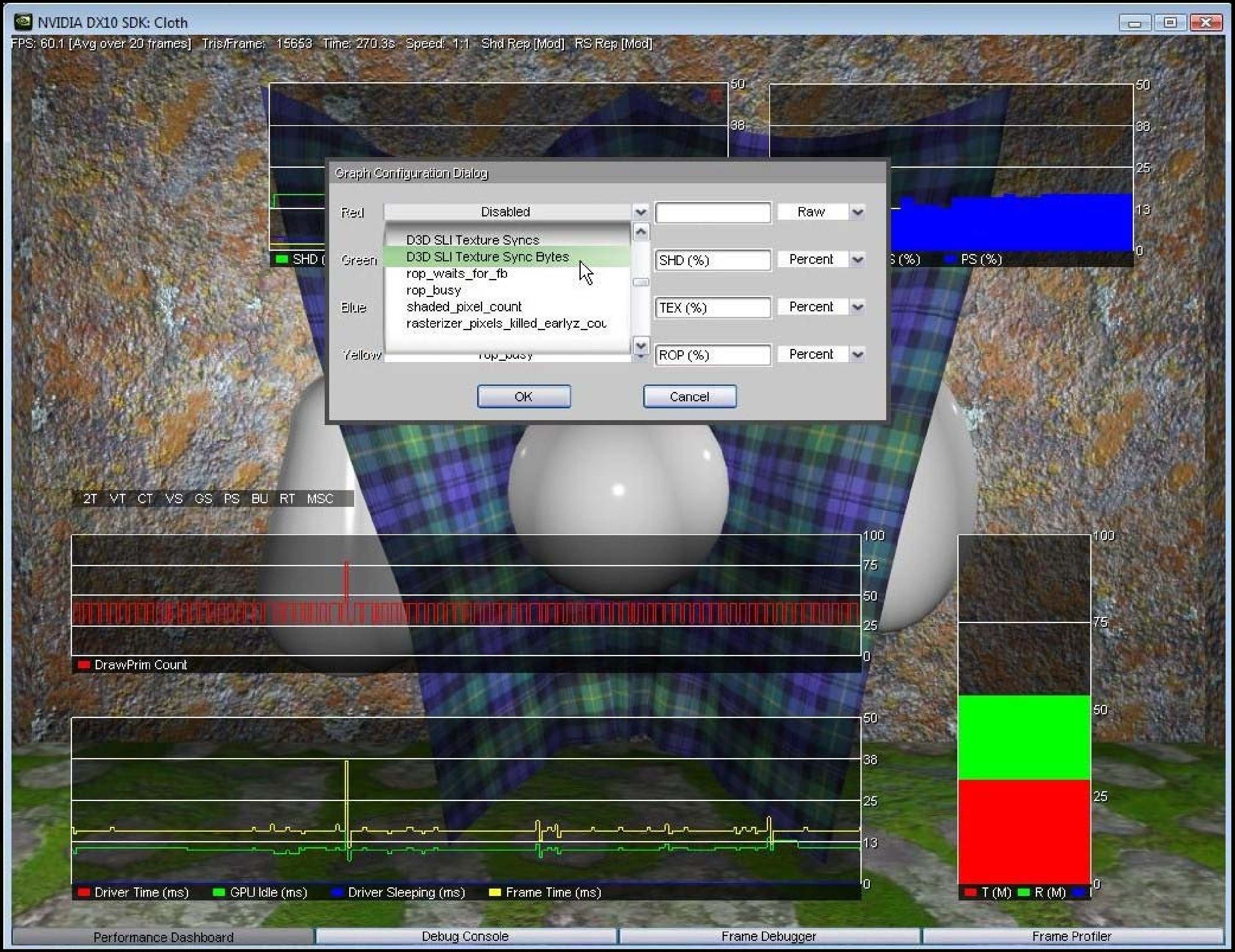
Demo: PerfHUD


NVIDIA DX10 SDK: Cloth
FPS: 60.1 [Avg over 20 frames]   Tris/Frame: 23350   Time: 119.3s   Speed: 1:1   Shd Rep [Mod]   RS Rep [Mod]
SHD (%)   TEX (%)   ROP (%)
VS (%)   GS (%)   PS (%)
2T VT CT VS GS PS BU RT MSC
DrawPrim Count
Driver Time (ms)   GPU Idle (ms)   Driver Sleeping (ms)   Frame Time (ms)
T (M)   R (M)
- Press Ctrl+Z to activate NVPerfHUD -

Demo: Performance Dashboard


NVIDIA DX10 SDK: Cloth
FPS: 60.6 [Avg over 20 frames] Tris/Frame: 15653 Time: 380.7s Speed: 1:1 Shd Rep [Mod] RS Rep [Mod]
50
38
25
13
0
🟩 SHD (%) 🟦 TEX (%) 🟨 ROP (%)
🟥 VS (%) 🟩 GS (%) 🟦 PS (%)
2T VT CT VS GS PS BU RT MSC
100
75
🟥 DrawPrim Count
🟥 Driver Time (ms) 🟩 GPU Idle (ms) 🟦 Driver Sleeping (ms) 🟨 Frame Time (ms)
🟥 ROP
Performance Dashboard
Debug Console
Frame Debugger
Frame Profiler

Demo: Performance Dashboard


NVIDIA DX10 SDK: Cloth

FPS: 60.1 [Avg over 20 frames] Tris/Frame: 15653 Time: 270.3s Speed: 1:1 Shd Rep [Mod] RS Rep [Mod]

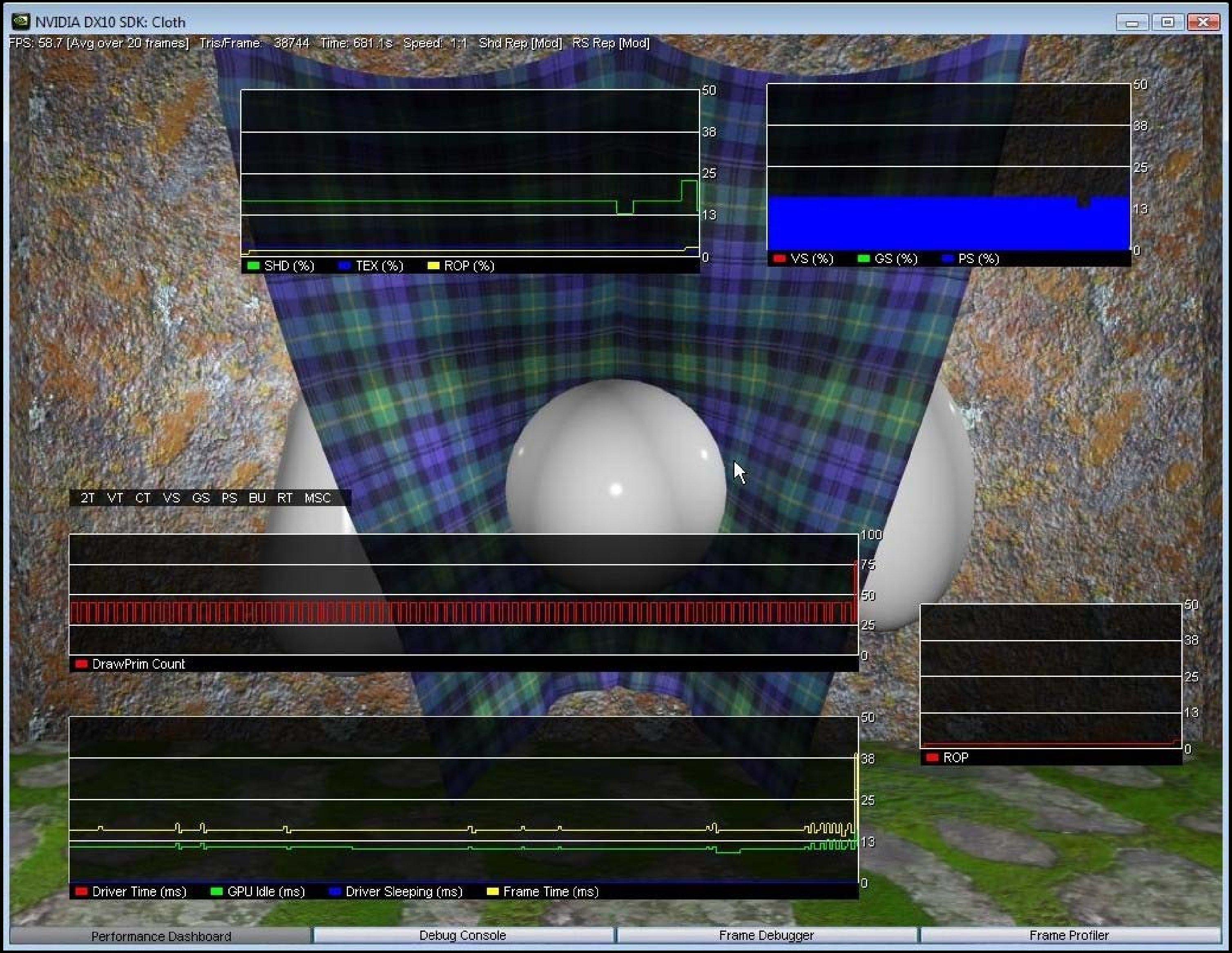
Graph Configuration Dialog

Red	Disabled		Raw
Green		SHD (%)	Percent
Blue		TEX (%)	Percent
Yellow	rop_busy	ROP (%)	Percent

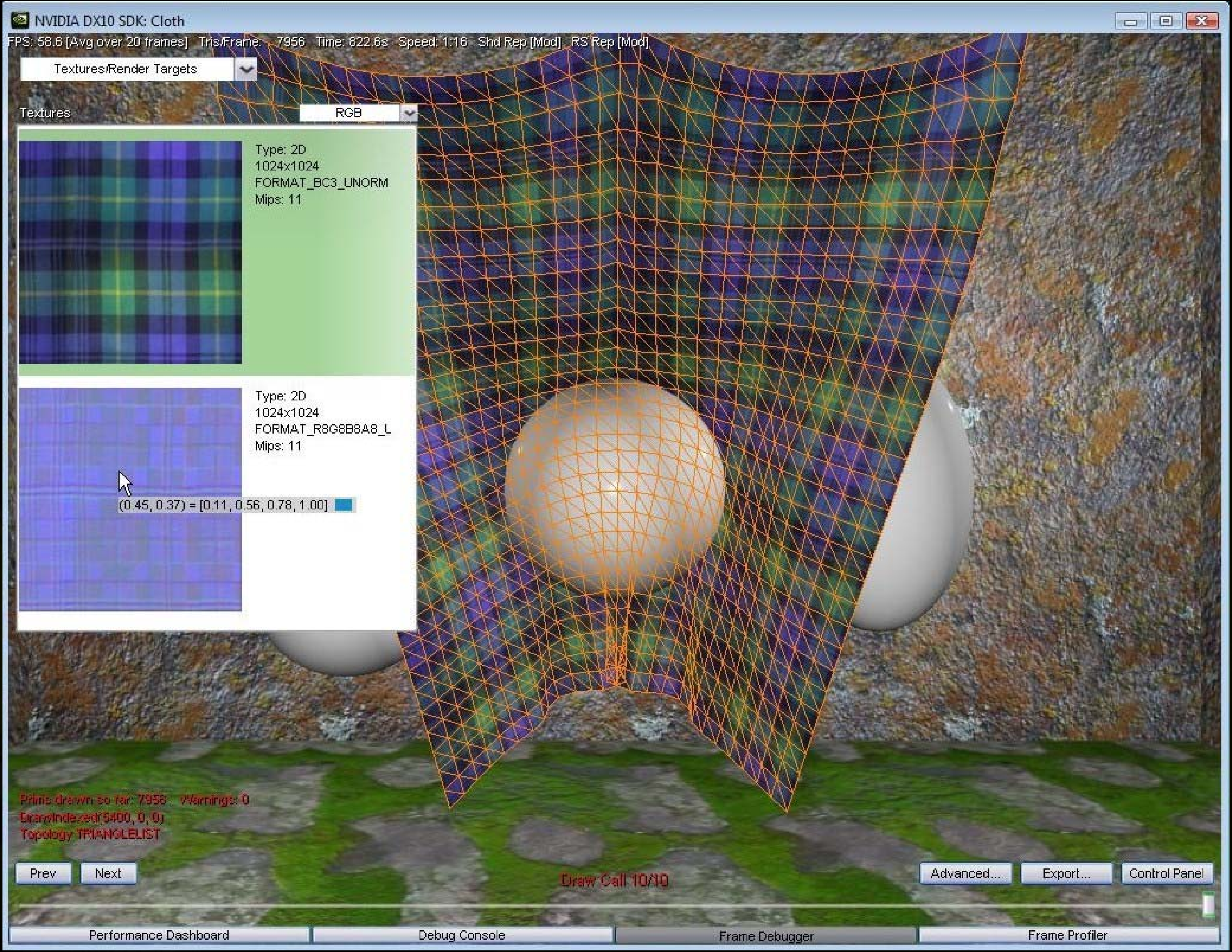
D3D SLI Texture Syncs
D3D SLI Texture Sync Bytes
rop_waits_for_fb
rop_busy
shaded_pixel_count
rasterizer_pixels_killed_earlyz_cou

OK Cancel

Demo: Performance Dashboard


NVIDIA DX10 SDK: Cloth
FPS: 58.7 [Avg over 20 frames]   Tris/Frame: 38744   Time: 681.1 s   Speed: 1:1   Shd Rep [Mod]   RS Rep [Mod]
50 38 25 13 0
🟩 SHD (%) 🟦 TEX (%) 🟨 ROP (%)
50 38 25 13 0
🟥 VS (%) 🟩 GS (%) 🟦 PS (%)
2T VT CT VS GS PS BU RT MSC
100 75 50 25 0
🟥 DrawPrim Count
50 38 25 13 0
🟥 ROP
50 38 25 13 0
🟥 Driver Time (ms) 🟩 GPU Idle (ms) 🟦 Driver Sleeping (ms) 🟨 Frame Time (ms)
Performance Dashboard
Debug Console
Frame Debugger
Frame Profiler

Demo: Frame Debugger



NVIDIA DX10 SDK: Cloth

FPS: 58.6 [Avg over 20 frames]   Tris/Frame: 7956   Time: 622.6s   Speed: 1:16   Shd Rep [Mod]   RS Rep [Mod]

Textures/Render Targets

Textures   RGB

Type: 2D
1024x1024
FORMAT_BC3_UNORM
Mips: 11

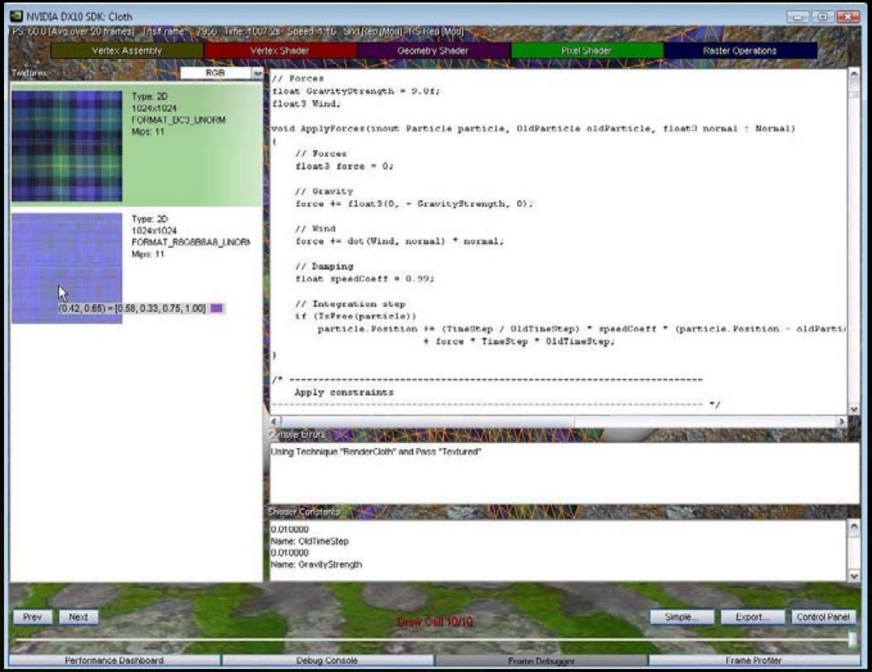
Type: 2D
1024x1024
FORMAT_R8G8B8A8_L
Mips: 11

(0.45, 0.37) = [0.11, 0.56, 0.78, 1.00] 🟦

Prims drawn so far: 7956   Warnings: 0
DrawIndexed(5400, 0, 0)
Topology TRIANGLELIST

Draw Call 10/10

Demo: Advanced Frame Debug



```
Type: 2D
1024x1024
FORMAT_BC3_UNORM
Mips: 11

Type: 2D
1024x1024
FORMAT_R8G8B8A8_UNORM
Mips: 11
(0.42, 0.65) = [0.58, 0.33, 0.75, 1.00]
```

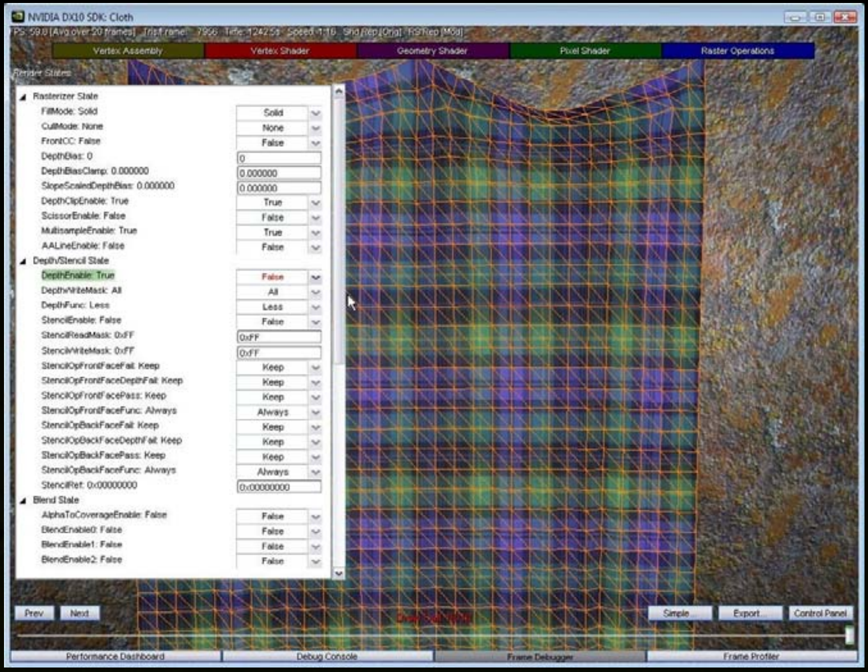
```
// Forces
float GravityStrength = 9.0f;
float3 Wind;

void ApplyForces(inout Particle particle, OldParticle oldParticle, float3 normal : Normal)
{
    // Forces
    float3 force = 0;

    // Gravity
    force += float3(0, - GravityStrength, 0);

    // Wind
    force += dot(Wind, normal) * normal;

    // Damping
    float speedCoeff = 0.99;

    // Integration step
    if (IsFree(particle))
        particle.Position += (TimeStep / OldTimeStep) * speedCoeff * (particle.Position - oldParti
                             + force * TimeStep * OldTimeStep;
}

/* ---------------------------------------------------------------------------
    Apply constraints
--------------------------------------------------------------------------- */
```

Compile Errors

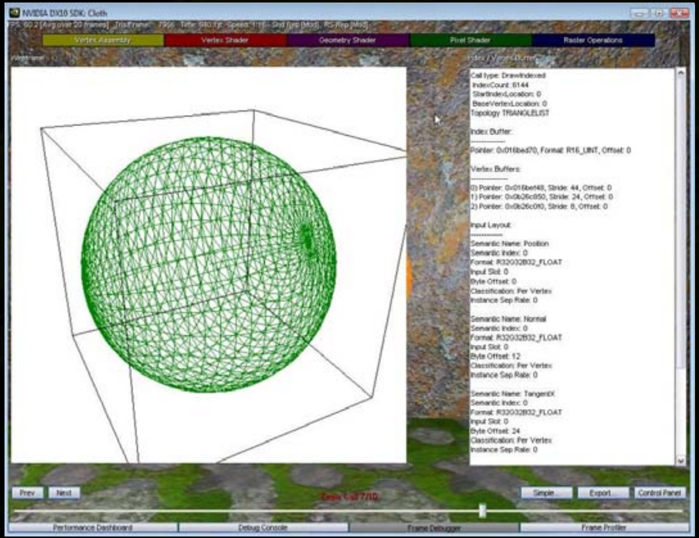
```
Using Technique "RenderCloth" and Pass "Textured"
```

Shader Constants

```
0.010000
Name: OldTimeStep
0.010000
Name: GravityStrength
```



```
Rasterizer State
    FillMode: Solid                          Solid
    CullMode: None                           None
    FrontCC: False                           False
    DepthBias: 0                             0
    DepthBiasClamp: 0.000000                 0.000000
    SlopeScaledDepthBias: 0.000000           0.000000
    DepthClipEnable: True                    True
    ScissorEnable: False                     False
    MultisampleEnable: True                  True
    AALineEnable: False                      False
Depth/Stencil State
    DepthEnable: True                        False
    DepthWriteMask: All                      All
    DepthFunc: Less                          Less
    StencilEnable: False                     False
    StencilReadMask: 0xFF                    0xFF
    StencilWriteMask: 0xFF                   0xFF
    StencilOpFrontFaceFail: Keep             Keep
    StencilOpFrontFaceDepthFail: Keep        Keep
    StencilOpFrontFacePass: Keep             Keep
    StencilOpFrontFaceFunc: Always           Always
    StencilOpBackFaceFail: Keep              Keep
    StencilOpBackFaceDepthFail: Keep         Keep
    StencilOpBackFacePass: Keep              Keep
    StencilOpBackFaceFunc: Always            Always
    StencilRef: 0x00000000                   0x00000000
Blend State
    AlphaToCoverageEnable: False             False
    BlendEnable0: False                      False
    BlendEnable1: False                      False
    BlendEnable2: False                      False
```

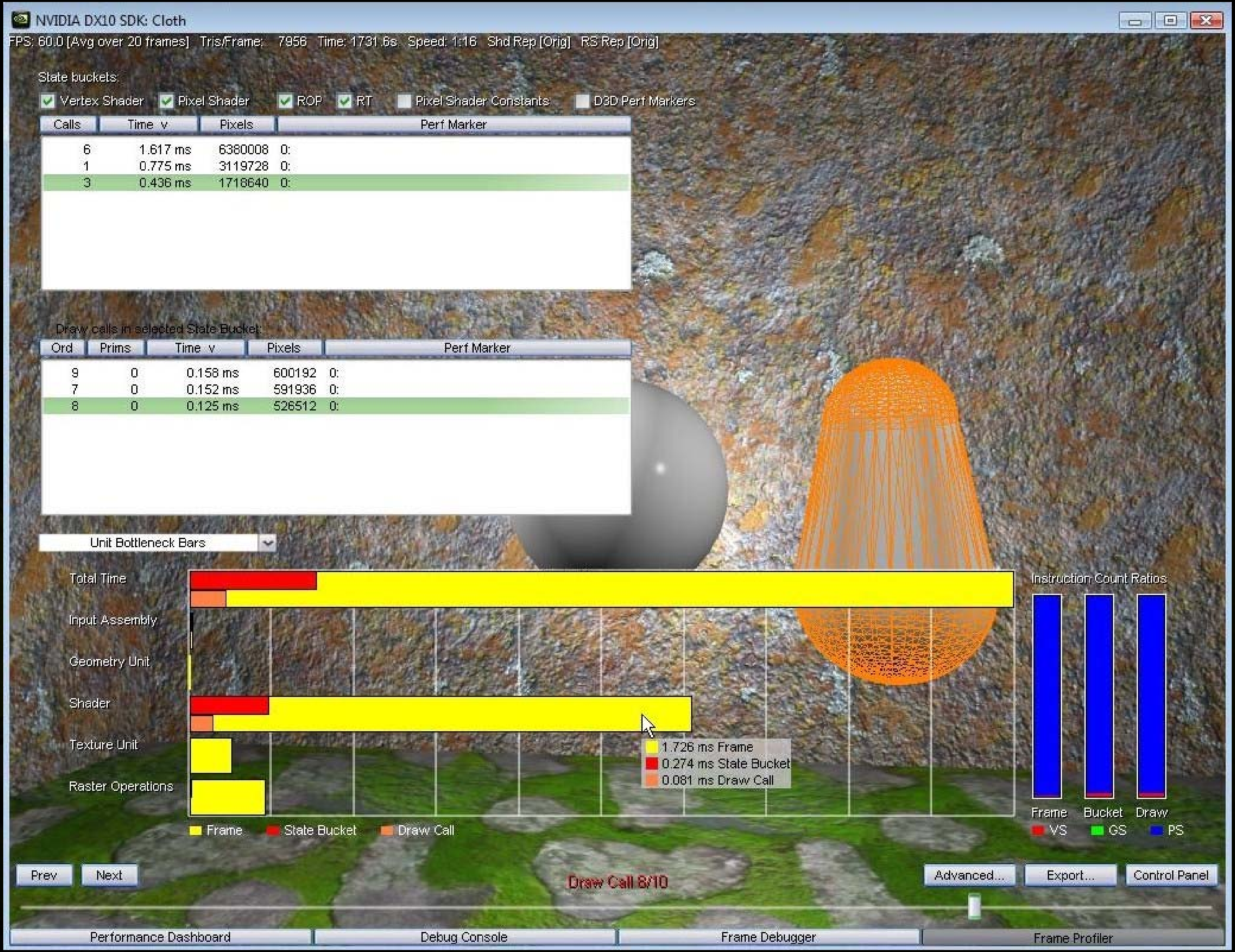


```
Call type: DrawIndexed
 IndexCount: 6144
 StartIndexLocation: 0
 BaseVertexLocation: 0
Topology TRIANGLELIST

Index Buffer:
-----------
Pointer: 0x016bed70, Format: R16_UINT, Offset: 0

Vertex Buffers:
-----------
0) Pointer: 0x016bef48, Stride: 44, Offset: 0
1) Pointer: 0x0b26c050, Stride: 24, Offset: 0
2) Pointer: 0x0b26c0f0, Stride: 8, Offset: 0

Input Layout:
-----------
Semantic Name: Position
Semantic Index: 0
Format: R32G32B32_FLOAT
Input Slot: 0
Byte Offset: 0
Classification: Per Vertex
Instance Sep Rate: 0

Semantic Name: Normal
Semantic Index: 0
Format: R32G32B32_FLOAT
Input Slot: 0
Byte Offset: 12
Classification: Per Vertex
Instance Sep Rate: 0

Semantic Name: TangentX
Semantic Index: 0
Format: R32G32B32_FLOAT
Input Slot: 0
Byte Offset: 24
Classification: Per Vertex
Instance Sep Rate: 0
```

Frame Profiler

- PerfHUD uses PerfKit and SimExp
- Sample ~35 performance counters
- Multiple passes on the scene since they can’t be at the same time
- Need to render THE SAME FRAME until all the counters are read
  - Only possible if application uses time-based animation
  - Intercept: QueryPerformanceCounter(), timeGetTime()
  - NO RDTSC!!

Demo: Frame Profiler



NVIDIA DX10 SDK: Cloth

FPS: 60.0 [Avg over 20 frames]   Tris/Frame: 7956   Time: 1731.6s   Speed: 1:16   Shd Rep [Orig]   RS Rep [Orig]

State buckets:

[✓] Vertex Shader [✓] Pixel Shader [✓] ROP [✓] RT [ ] Pixel Shader Constants [ ] D3D Perf Markers

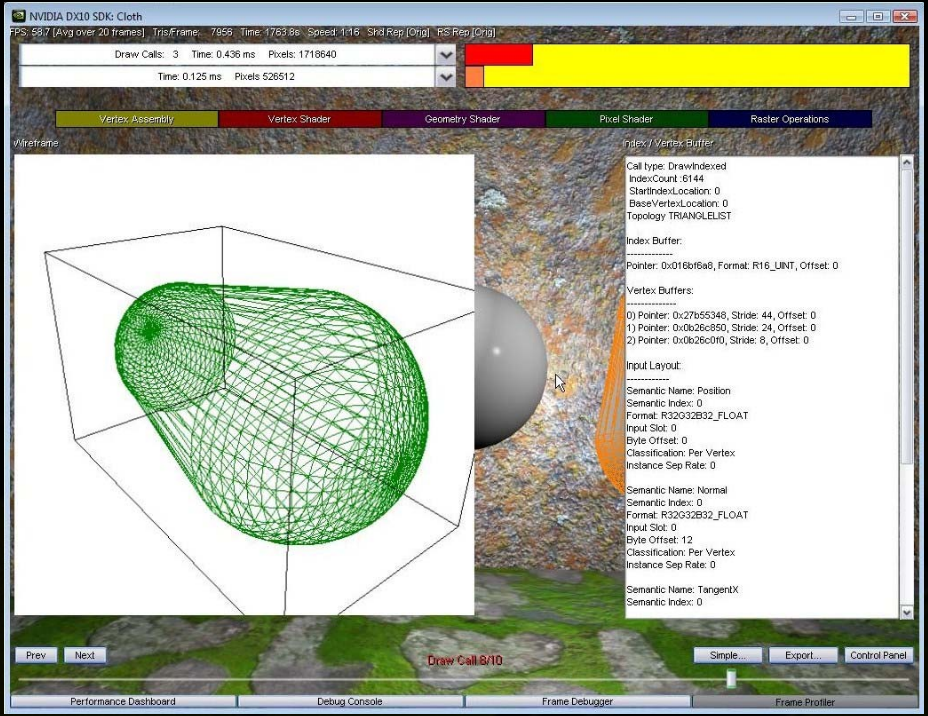
Calls	Time v	Pixels	Perf Marker
6	1.617 ms	6380008	0:
1	0.775 ms	3119728	0:
3	0.436 ms	1718640	0:

Draw calls in selected State Bucket:

Ord	Prims	Time v	Pixels	Perf Marker
9	0	0.158 ms	600192	0:
7	0	0.152 ms	591936	0:
8	0	0.125 ms	526512	0:

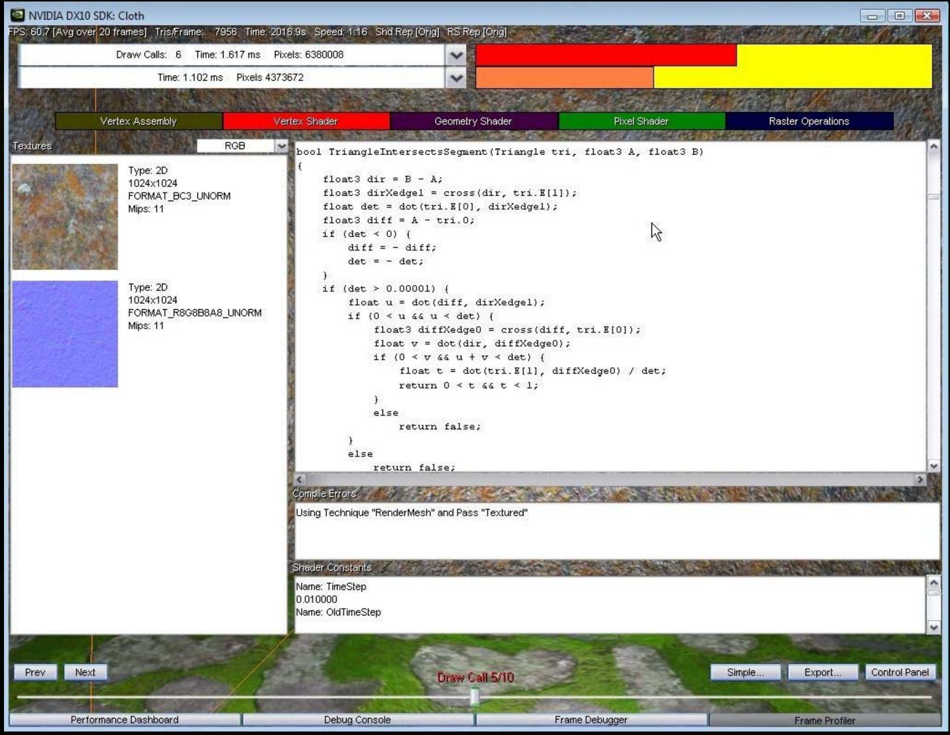
Unit Bottleneck Bars

Demo: Advanced Frame Profiler



```
FPS: 58.7 [Avg over 20 frames]   Tris/Frame:   7956   Time: 1763.8s   Speed: 1:16   Shd Rep [Orig]   RS Rep [Orig]
Draw Calls:  3    Time: 0.436 ms    Pixels: 1718640
Time: 0.125 ms    Pixels 526512
```

```
Call type: DrawIndexed
 IndexCount :6144
 StartIndexLocation: 0
 BaseVertexLocation: 0
Topology TRIANGLELIST

Index Buffer:
-------------
Pointer: 0x016bf6a8, Format: R16_UINT, Offset: 0

Vertex Buffers:
--------------
0) Pointer: 0x27b55348, Stride: 44, Offset: 0
1) Pointer: 0x0b26c850, Stride: 24, Offset: 0
2) Pointer: 0x0b26c0f0, Stride: 8, Offset: 0

Input Layout:
------------
Semantic Name: Position
Semantic Index: 0
Format: R32G32B32_FLOAT
Input Slot: 0
Byte Offset: 0
Classification: Per Vertex
Instance Sep Rate: 0

Semantic Name: Normal
Semantic Index: 0
Format: R32G32B32_FLOAT
Input Slot: 0
Byte Offset: 12
Classification: Per Vertex
Instance Sep Rate: 0

Semantic Name: TangentX
Semantic Index: 0
```



```
FPS: 60.7 [Avg over 20 frames]   Tris/Frame:   7956   Time: 2016.9s   Speed: 1:16   Shd Rep [Orig]   RS Rep [Orig]
Draw Calls:  6    Time: 1.617 ms    Pixels: 6380008
Time: 1.102 ms    Pixels 4373672
```

```
Type: 2D
1024x1024
FORMAT_BC3_UNORM
Mips: 11

Type: 2D
1024x1024
FORMAT_R8G8B8A8_UNORM
Mips: 11
```

```
bool TriangleIntersectsSegment(Triangle tri, float3 A, float3 B)
{
    float3 dir = B - A;
    float3 dirXedge1 = cross(dir, tri.E[1]);
    float det = dot(tri.E[0], dirXedge1);
    float3 diff = A - tri.0;
    if (det < 0) {
        diff = - diff;
        det = - det;
    }
    if (det > 0.00001) {
        float u = dot(diff, dirXedge1);
        if (0 < u && u < det) {
            float3 diffXedge0 = cross(diff, tri.E[0]);
            float v = dot(dir, diffXedge0);
            if (0 < v && u + v < det) {
                float t = dot(tri.E[1], diffXedge0) / det;
                return 0 < t && t < 1;
            }
            else
                return false;
        }
        else
            return false;
```

```
Using Technique "RenderMesh" and Pass "Textured"
```

```
Name: TimeStep
0.010000
Name: OldTimeStep
```

Counter Types

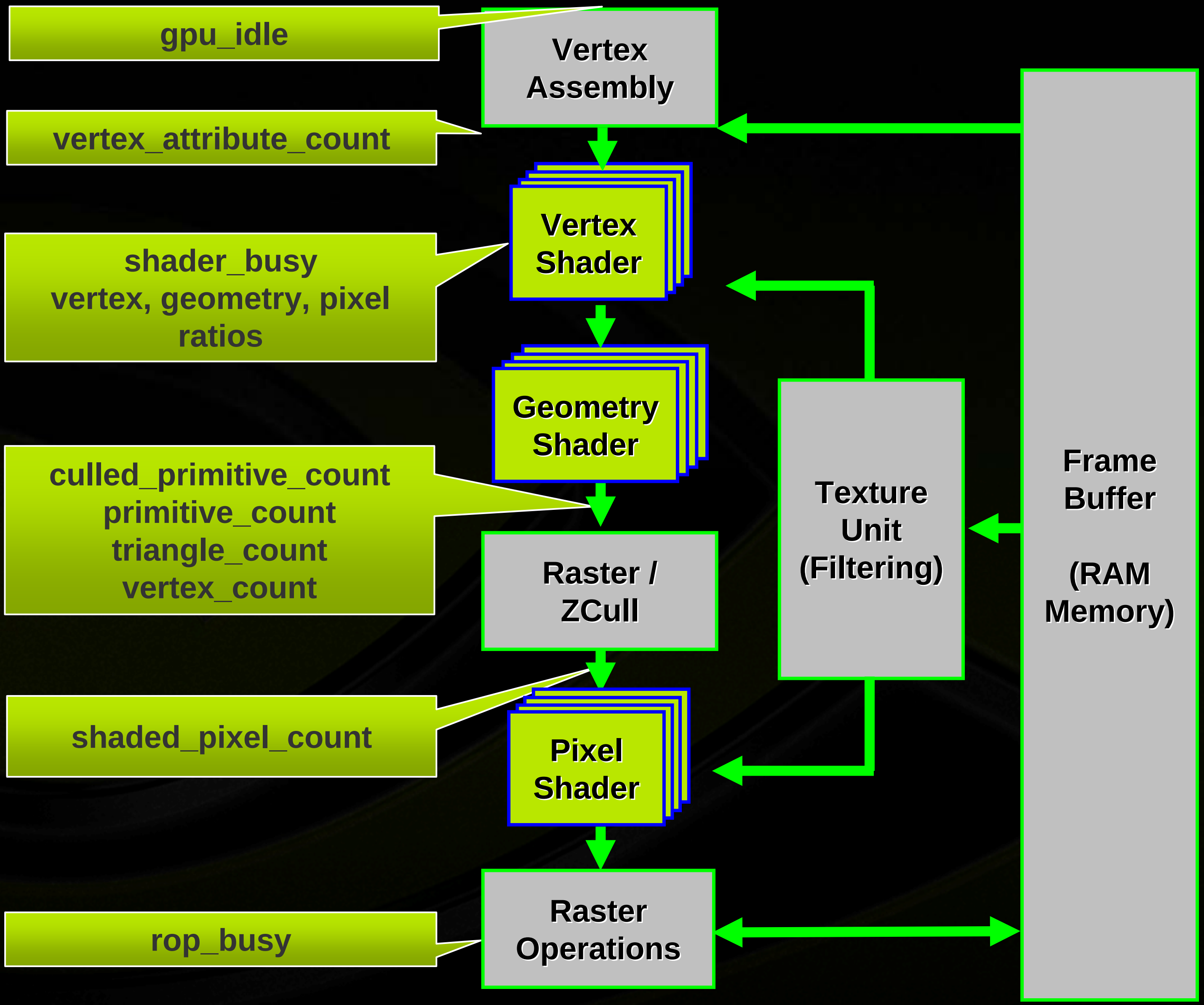
- SW/Driver Counters: PerfAPI, PDH
- Raw GPU Counters: PerfAPI, PDH
- Simplified Experiments: PerfAPI
- Instrumented GPUs

GeForce 8800 Series	GeForce 7800 GTX
GeForce 7950/7900 GTX & GT	GeForce 6800 Ultra & GT
Quadro FX 5500 & 4500	GeForce 6600

Direct3D/OpenGL Driver Counters

- General
  - FPS
  - ms per frame
- Driver
  - Driver frame time (total time spent in driver)
  - Driver sleep time (waiting for GPU)
  - % of the frame time driver is waiting
- Counts
  - Batches, vertices, primitives
  - (Direct3D) Triangles and instanced triangles
  - (Direct3D) Locked render targets
- Memory
  - AGP memory used
  - Video memory used and total

GPU Counters

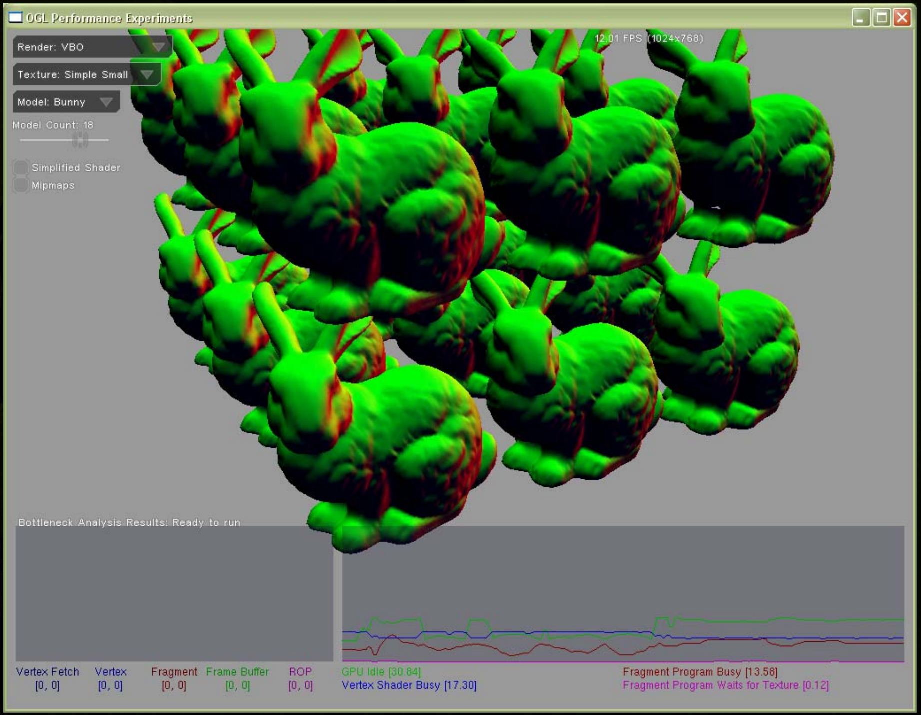


How do I use PerfKit counters?

- PerfAPI: Easy integration of PerfKit
  - Real time performance monitoring using GPU and driver counters, round robin sampling
  - Simplified Experiments for single frame analysis
- PDH: Performance Data Helper for Windows
  - Driver data, GPU counters, and OS information
  - Exposed via Perfmon
  - Good for rapid prototyping
- Sample code and helper classes provided in PerfSDK

PerfAPI: Real Time

```
// Somewhere in setup
NVPMAddCounterByName(“vertex_shader_busy”);
NVPMAddCounterByName (“pixel_shader_busy”);
NVPMAddCounterByName (“shader_waits_for_texture”);
NVPMAddCounterByName (“gpu_idle”);

// In your rendering loop, sample using names
NVPMSample(NULL, &nNumSamples);
NVPMGetCounterValueByName(“vertex_shader_busy”, 0, &nVSEvents, &nVSCycles);
NVPMGetCounterValueByName(“pixel_shader_busy”, 0, &nPSEvents, &nPSCycles);
NVPMGetCounterValueByName(“shader_waits_for_texture”, 0, &nTexEvents, &nTexCycles);
NVPMGetCounterValueByName(“gpu_idle”, 0, &nIdleEvents, &nIdleCycles);
```

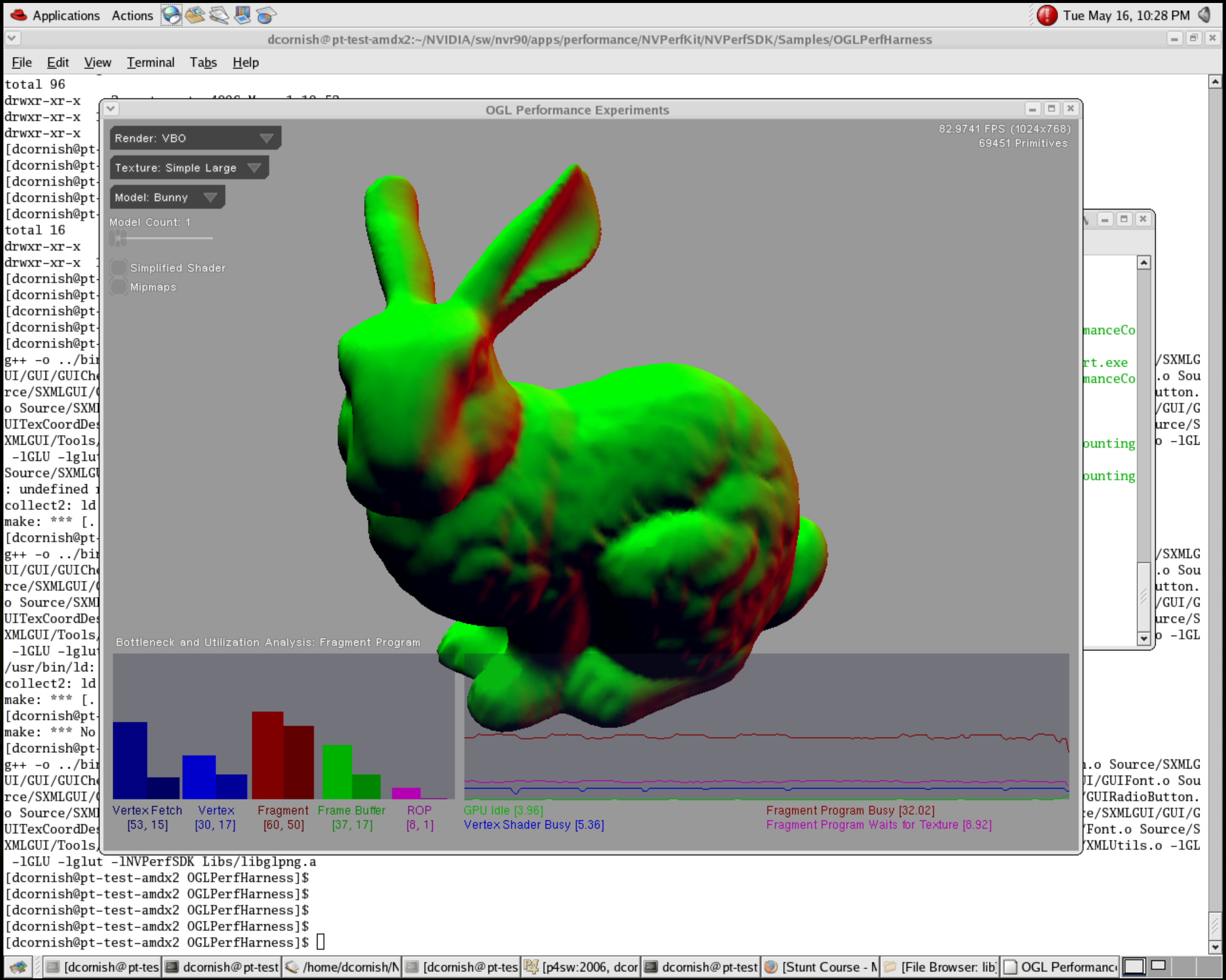


Bottleneck Analysis Results: Ready to run

Vertex Fetch [0, 0]
Vertex [0, 0]
Fragment [0, 0]
Frame Buffer [0, 0]
ROP [0, 0]

GPU Idle [30.84]
Vertex Shader Busy [17.30]
Fragment Program Busy [13.58]
Fragment Program Waits for Texture [0.12]

PerfAPI: SimExp



```
total 96
drwxr-xr-x
drwxr-xr-x
drwxr-xr-x
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
total 16
drwxr-xr-x
drwxr-xr-x
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
[dcornish@pt-
g++ -o ../bi
UI/GUI/GUICh
rce/SXMLGUI/
o Source/SXM
UITexCoordDe
XMLGUI/Tools
 -lGLU -lglu
Source/SXMLGU
: undefined
collect2: ld
make: *** [.
[dcornish@pt-
g++ -o ../bi
UI/GUI/GUICh
rce/SXMLGUI/
o Source/SXM
UITexCoordDe
XMLGUI/Tools
 -lGLU -lglu
/usr/bin/ld:
collect2: ld
make: *** [.
[dcornish@pt-
make: *** No
[dcornish@pt-
g++ -o ../bi
UI/GUI/GUICh
rce/SXMLGUI/
o Source/SXM
UITexCoordDe
XMLGUI/Tools
 -lGLU -lglut -lNVPerfSDK Libs/libglpng.a
[dcornish@pt-test-amdx2 OGLPerfHarness]$
[dcornish@pt-test-amdx2 OGLPerfHarness]$
[dcornish@pt-test-amdx2 OGLPerfHarness]$
[dcornish@pt-test-amdx2 OGLPerfHarness]$
[dcornish@pt-test-amdx2 OGLPerfHarness]$
```

```
manceCo
rt.exe
manceCo
ounting
ounting
```

```
/SXMLG
.o Sou
utton.
/GUI/G
urce/S
o -lGL
/SXMLG
.o Sou
utton.
/GUI/G
urce/S
o -lGL
.o Source/SXMLG
UI/GUIFont.o Sou
/GUIRadioButton.
e/SXMLGUI/GUI/G
Font.o Source/S
XMLUtils.o -lGL
```

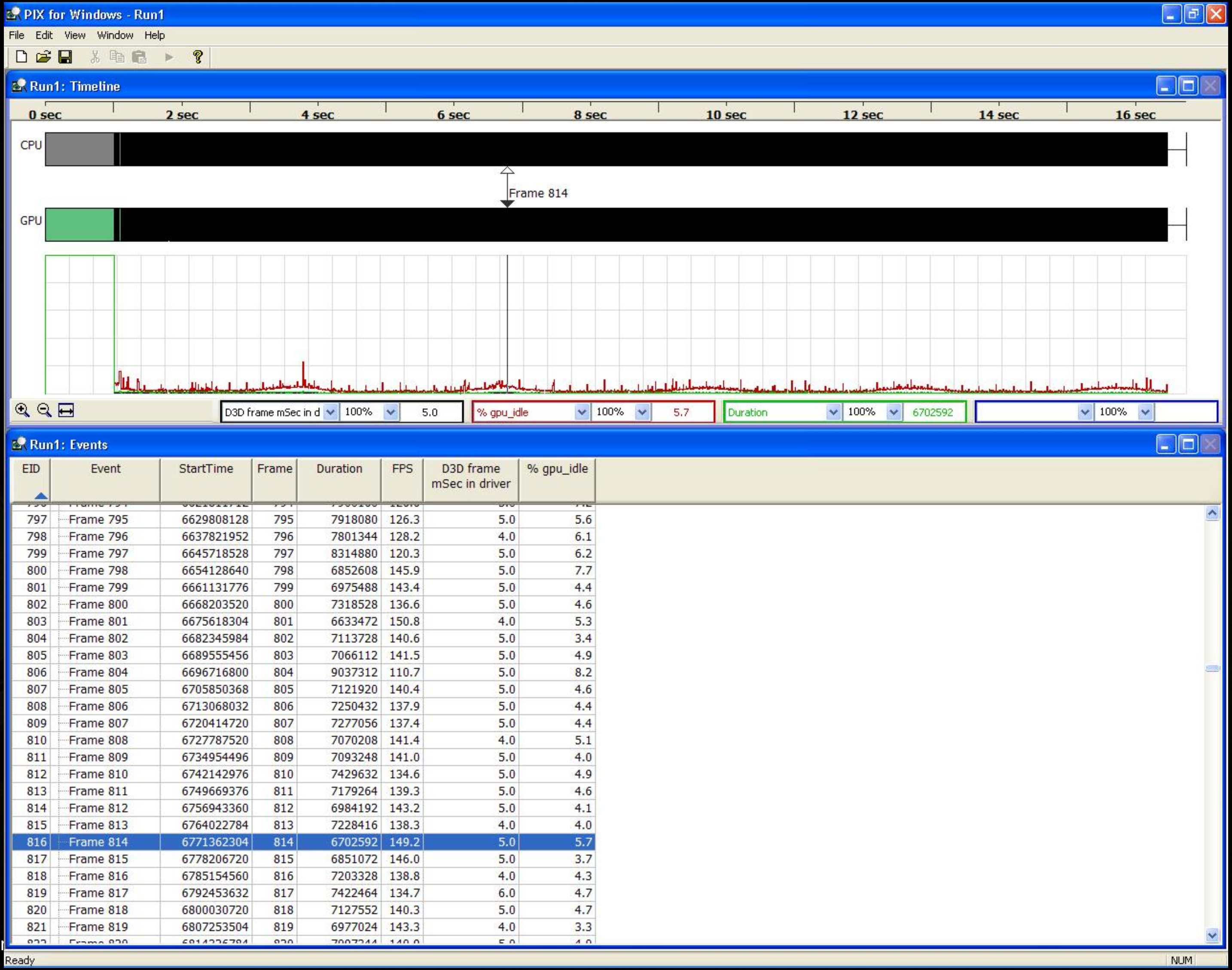
82.9741 FPS (1024x768)
69451 Primitives

Bottleneck and Utilization Analysis: Fragment Program

Vertex Fetch [53, 15]
Vertex [30, 17]
Fragment [60, 50]
Frame Buffer [37, 17]
ROP [8, 1]

GPU Idle [3.96]
Vertex Shader Busy [5.36]
Fragment Program Busy [32.02]
Fragment Program Waits for Texture [8.92]

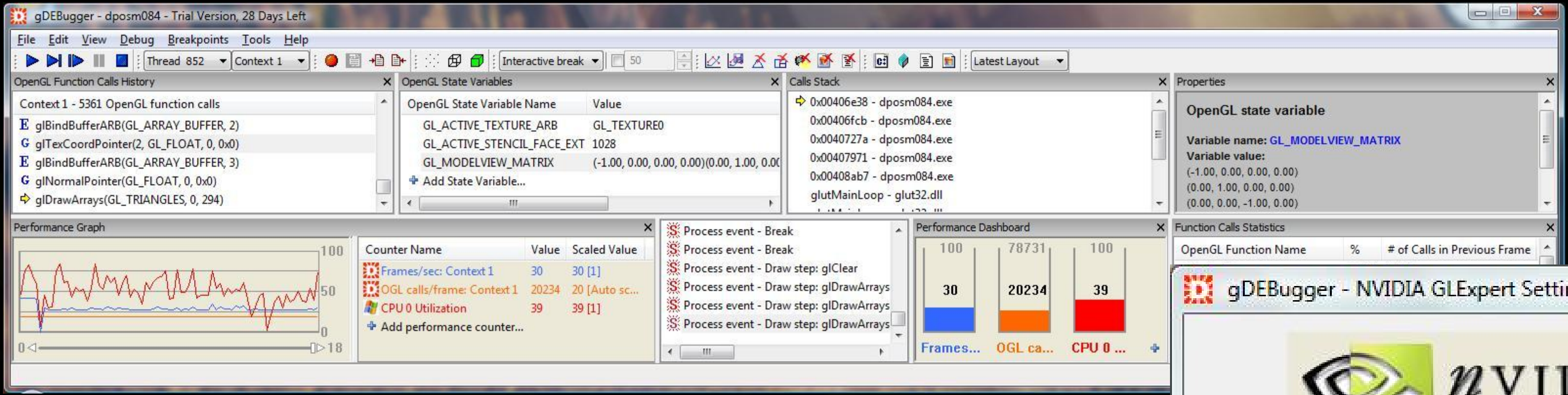
Associated Tools: NVIDIA Plug-In for Microsoft PIX for Windows



D3D frame mSec in d 100% 5.0
% gpu_idle 100% 5.7
Duration 100% 6702592

EID	Event	StartTime	Frame	Duration	FPS	D3D frame mSec in driver	% gpu_idle
797	Frame 795	6629808128	795	7918080	126.3	5.0	5.6
798	Frame 796	6637821952	796	7801344	128.2	4.0	6.1
799	Frame 797	6645718528	797	8314880	120.3	5.0	6.2
800	Frame 798	6654128640	798	6852608	145.9	5.0	7.7
801	Frame 799	6661131776	799	6975488	143.4	5.0	4.4
802	Frame 800	6668203520	800	7318528	136.6	5.0	4.6
803	Frame 801	6675618304	801	6633472	150.8	4.0	5.3
804	Frame 802	6682345984	802	7113728	140.6	5.0	3.4
805	Frame 803	6689555456	803	7066112	141.5	5.0	4.9
806	Frame 804	6696716800	804	9037312	110.7	5.0	8.2
807	Frame 805	6705850368	805	7121920	140.4	5.0	4.6
808	Frame 806	6713068032	806	7250432	137.9	5.0	4.4
809	Frame 807	6720414720	807	7277056	137.4	5.0	4.4
810	Frame 808	6727787520	808	7070208	141.4	4.0	5.1
811	Frame 809	6734954496	809	7093248	141.0	5.0	4.0
812	Frame 810	6742142976	810	7429632	134.6	5.0	4.9
813	Frame 811	6749669376	811	7179264	139.3	5.0	4.6
814	Frame 812	6756943360	812	6984192	143.2	5.0	4.1
815	Frame 813	6764022784	813	7228416	138.3	4.0	4.0
816	Frame 814	6771362304	814	6702592	149.2	5.0	5.7
817	Frame 815	6778206720	815	6851072	146.0	5.0	3.7
818	Frame 816	6785154560	816	7203328	138.8	4.0	4.3
819	Frame 817	6792453632	817	7422464	134.7	6.0	4.7
820	Frame 818	6800030720	818	7127552	140.3	5.0	4.7
821	Frame 819	6807253504	819	6977024	143.3	4.0	3.3

Graphic Remedy's gDEBugger


gDEBugger - dposm084 - Trial Version, 28 Days Left
File Edit View Debug Breakpoints Tools Help
Thread 852
Context 1
Interactive break
Latest Layout

OpenGL Function Calls History

Context 1 - 5361 OpenGL function calls

- E glBindBufferARB(GL_ARRAY_BUFFER, 2)
- G glTexCoordPointer(2, GL_FLOAT, 0, 0x0)
- E glBindBufferARB(GL_ARRAY_BUFFER, 3)
- G glNormalPointer(GL_FLOAT, 0, 0x0)
- glDrawArrays(GL_TRIANGLES, 0, 294)

OpenGL State Variables

OpenGL State Variable Name	Value
GL_ACTIVE_TEXTURE_ARB	GL_TEXTURE0
GL_ACTIVE_STENCIL_FACE_EXT	1028
GL_MODELVIEW_MATRIX	(-1.00, 0.00, 0.00, 0.00)(0.00, 1.00, 0.00
Add State Variable...	

Calls Stack

```
0x00406e38 - dposm084.exe
0x00406fcb - dposm084.exe
0x0040727a - dposm084.exe
0x00407971 - dposm084.exe
0x00408ab7 - dposm084.exe
glutMainLoop - glut32.dll
```

Properties

OpenGL state variable

Variable name: GL_MODELVIEW_MATRIX
Variable value:
(-1.00, 0.00, 0.00, 0.00)
(0.00, 1.00, 0.00, 0.00)
(0.00, 0.00, -1.00, 0.00)

Performance Graph

Counter Name	Value	Scaled Value
Frames/sec: Context 1	30	30 [1]
OGL calls/frame: Context 1	20234	20 [Auto sc...
CPU 0 Utilization	39	39 [1]
Add performance counter...		

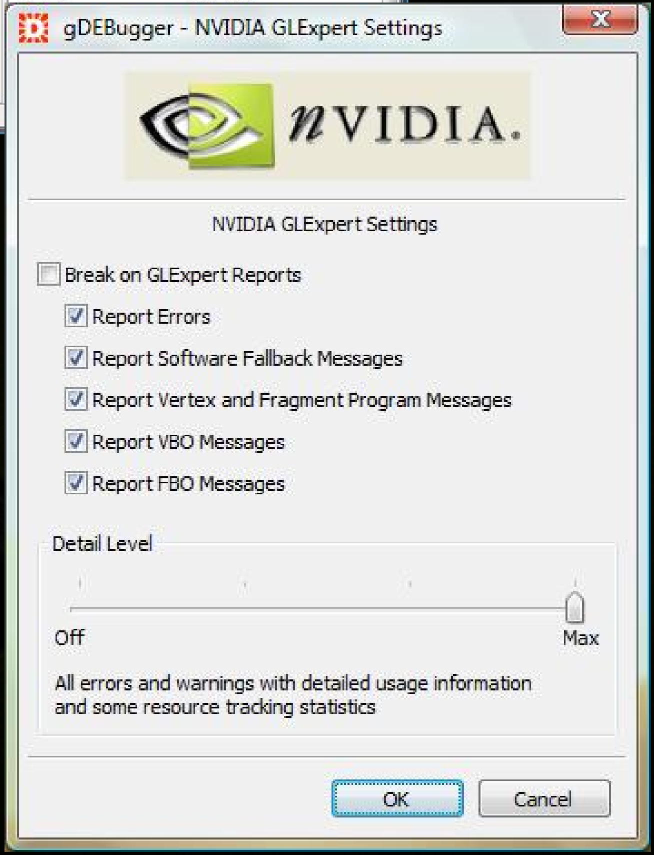
```
Process event - Break
Process event - Break
Process event - Draw step: glClear
Process event - Draw step: glDrawArrays
Process event - Draw step: glDrawArrays
Process event - Draw step: glDrawArrays
```

Performance Dashboard

Frames...	OGL ca...	CPU 0 ...
100	78731	100
30	20234	39

Function Calls Statistics

OpenGL Function Name	%	# of Calls in Previous Frame



NVIDIA GLExpert Settings

- [ ] Break on GLExpert Reports
  - [✓] Report Errors
  - [✓] Report Software Fallback Messages
  - [✓] Report Vertex and Fragment Program Messages
  - [✓] Report VBO Messages
  - [✓] Report FBO Messages

Detail Level: Off — Max [✓ Max]

All errors and warnings with detailed usage information and some resource tracking statistics

- OpenGL Debugging and Performance Tool
  - PerfKit and GLExpert integrated
  - Version 3.0 to support Vista
  - Linux version in Beta, release soon
  - Free academic licenses available from Graphic Remedy and the OpenGL ARB.
  - One year license for full featured version, including all software updates
- Details: http://academic.gremedy.com

Project Status

- PerfKit 4.2 for Windows 32bit available: developer.nvidia.com
- PerfKit 5.0 (Q2 2007)
  - PerfHUD 5.0
  - ForceWare Release 100 Driver
  - GeForce 8800 support
  - Windows 32 and 64 bit
  - Linux 32 and 64 bit
- PerfGraph: www.sourceforge.org\perfgraph
- Instrumented GPUs

GeForce 8800 Series	GeForce 7800 GTX
GeForce 7950/7900 GTX & GT	GeForce 6800 Ultra & GT
Quadro FX 5500 & 4500	GeForce 6600

Feedback and Support: NVPerfKit@nvidia.com

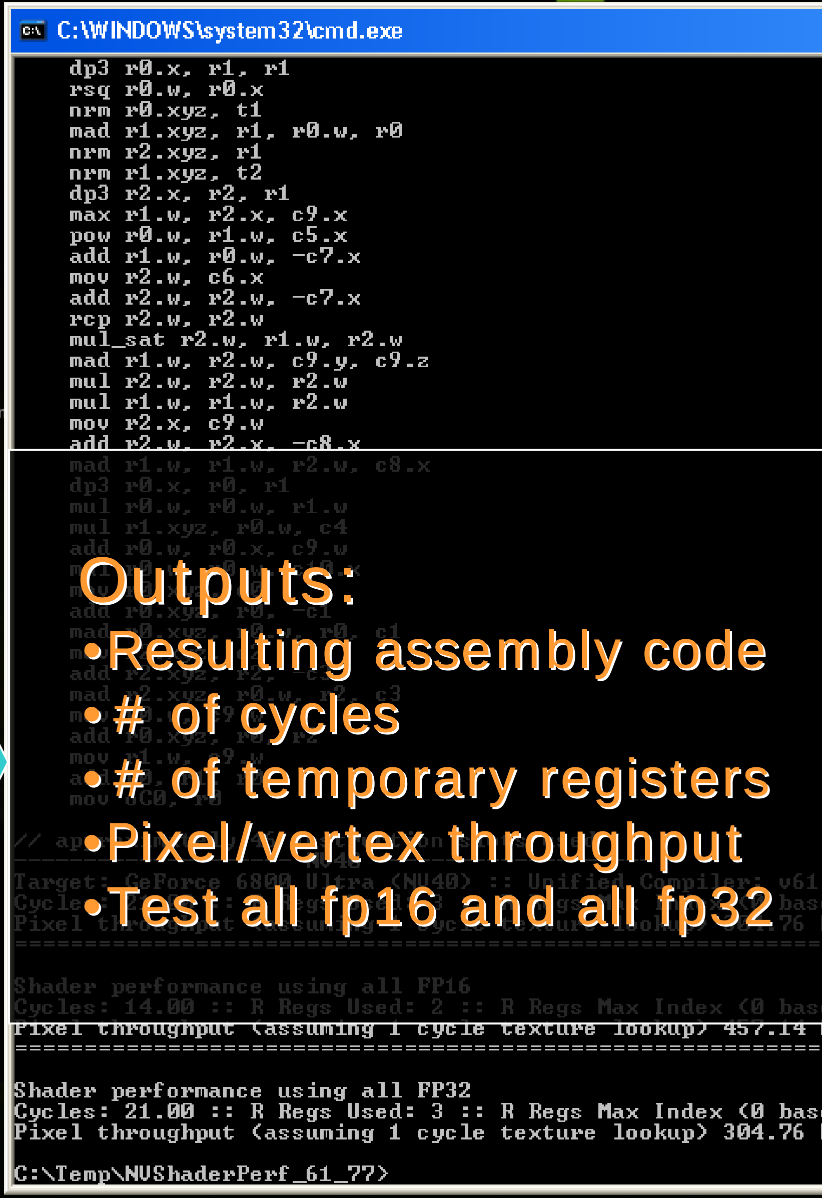
ShaderPerf 2.0

```
v2f BumpReflectVS(a2v IN,
                uniform float4x4 WorldViewProj,
                uniform float4x4 World,
                uniform float4x4 ViewIT)
{
      v2f OUT;
      // Position in screen space.
      OUT.Position = mul(IN.Position, WorldViewProj);
      // pass texture coordinates for fetching the normal map
      OUT.TexCoord.xyz = IN.TexCoord;
      OUT.TexCoord.w = 1.0;
      // compute the 4x4 tranform from tangent space to object space
      float3x3 TangentToObjSpace;
      // first rows are the tangent and binormal scaled by the bump scale
      TangentToObjSpace[0] = float3(IN.Tangent.x, IN.Binormal.x, IN.Normal.x);
      TangentToObjSpace[1] = float3(IN.Tangent.y, IN.Binormal.y, IN.Normal.y);
      TangentToObjSpace[2] = float3(IN.Tangent.z, IN.Binormal.z, IN.Normal.z);
      OUT.TexCoord1.x = dot(World[0].xyz, TangentToObjSpace[0]);
      OUT.TexCoord1.y = dot(World[1].xyz, TangentToObjSpace[0]);
      OUT.TexCoord1.z = dot(World[2].xyz, TangentToObjSpace[0]);
      OUT.TexCoord2.x = dot(World[0].xyz, TangentToObjSpace[1]);
      OUT.TexCoord2.y = dot(World[1].xyz, TangentToObjSpace[1]);
      OUT.TexCoord2.z = dot(World[2].xyz, TangentToObjSpace[1]);
      OUT.TexCoord3.x = dot(World[0].xyz, TangentToObjSpace[2]);
      OUT.TexCoord3.y = dot(World[1].xyz, TangentToObjSpace[2]);
      OUT.TexCoord3.z = dot(World[2].xyz, TangentToObjSpace[2]);
      float4 worldPos = mul(IN.Position, World);
      // compute the eye vector (going from shaded point to eye) in cube space
      float4 eyeVector = worldPos - ViewIT[3]; // view inv. transpose contains eye position in world space in
      OUT.TexCoord1.w = eyeVector.x;
      OUT.TexCoord2.w = eyeVector.y;
      OUT.TexCoord3.w = eyeVector.z;
      return OUT;
}

///////////////// pixel shader //////////////////

float4 BumpReflectPS(v2f IN,
                uniform sampler2D NormalMap,
                uniform samplerCUBE EnvironmentMap,
      uniform float BumpScale) : COLOR
{
      // fetch the bump normal from the normal map
      float3 normal = tex2D(NormalMap, IN.T
      normal = normalize(float3(normal.x *
      // transform the bump normal into cub
      // then use the transformed normal a
      // used to fetch the cube map
      // (we multiply by 2 only to increase
      float3 eyevec = float3(IN.TexCoord1.w
      float3 worldNorm;
      worldNorm.x = dot(IN.TexCoord1.xyz,normal);
      worldNorm.y = dot(IN.TexCoord2.xyz,normal);
      worldNorm.z = dot(IN.TexCoord3.xyz,normal);
      float3 lookup = reflect(eyevec, worldNorm);
      return texCUBE(EnvironmentMap, lookup);
}
```

Inputs:
- GLSL, Cg, HLSL
- PS1.x,PS2.x,PS3.x
- VS1.x,VS2.x, VS3.x
- !!FP1.0
- !!ARBfp1.0

ShaderPerf

GPU Arch:
- GeForce 7X00
- GeForce 6X00
- Geforce FX series
- Quadro FX series


C:\WINDOWS\system32\cmd.exe

```
        dp3 r0.x, r1, r1
        rsq r0.w, r0.x
        nrm r0.xyz, t1
        mad r1.xyz, r1, r0.w, r0
        nrm r2.xyz, r1
        nrm r1.xyz, t2
        dp3 r2.x, r2, r1
        max r1.w, r2.x, c9.x
        pow r0.w, r1.w, c5.x
        add r1.w, r0.w, -c7.x
        mov r2.w, c6.x
        add r2.w, r2.w, -c7.x
        rcp r2.w, r2.w
        mul_sat r2.w, r1.w, r2.w
        mad r1.w, r2.w, c9.y, c9.z
        mul r2.w, r2.w, r2.w
        mul r1.w, r1.w, r2.w
        mov r2.x, c9.w
        add r2.x, r2.x, -c8.x
        mad r1.w, r1.w, r2.w, c8.x
        dp3 r0.x, r0, r1
        mul r0.w, r0.w, r1.w
        mul r1.xyz, r0.w, c4
        add r0.w, r0.x, c9.w
        add r0.xyz, r0, -c1
        mad r0.xyz,      r0, c1
        add r2.xyz, r2, -c3
        mad r2.xyz, r0.w, r2, c3
        add r0.xyz, r0, r2
        mov r1.w, c9.w
        mov oC0, r0

// ap
Target: GeForce 6800 Ultra (NV40) :: Unified Compiler: v61
Cycle
Pixel throughput (assuming 1 cycle texture lookup)              76
================================================================

Shader performance using all FP16
Cycles: 14.00 :: R Regs Used: 2 :: R Regs Max Index (0 bas
Pixel throughput (assuming 1 cycle texture lookup) 457.14
================================================================

Shader performance using all FP32
Cycles: 21.00 :: R Regs Used: 3 :: R Regs Max Index (0 bas
Pixel throughput (assuming 1 cycle texture lookup) 304.76

C:\Temp\NVShaderPerf_61_77>
```

Outputs:
- Resulting assembly code
- # of cycles
- # of temporary registers
- Pixel/vertex throughput
- Test all fp16 and all fp32

ShaderPerf: In your pipeline

- Test current performance
  - Compare with shader cycle budgets
  - Test optimization opportunities
  - Not just Tex/ALU balance: cycles & throughput
- Automated regression analysis
- Integrated in FX Composer 2.0
  - Artists/TDs code expensive shaders
  - Achieve optimum performance

ShaderPerf 2.0 Alpha

- Supports Direct3D/HLSL
- GeForce 7XXX, 6XXX, and FX GPUs
- ForceWare Release 100 Unified Compiler
- Improved vertex performance simulation and throughput calculation
- Multiple drivers from one ShaderPerf
- Smaller footprint
- New programmatic interface

ShaderPerf 2.0: Beta

- Full support for Cg and GLSL, vertex and fragment programs
- Support for GeForce 8XXX series GPUs
- Geometry shaders and geometry throughput

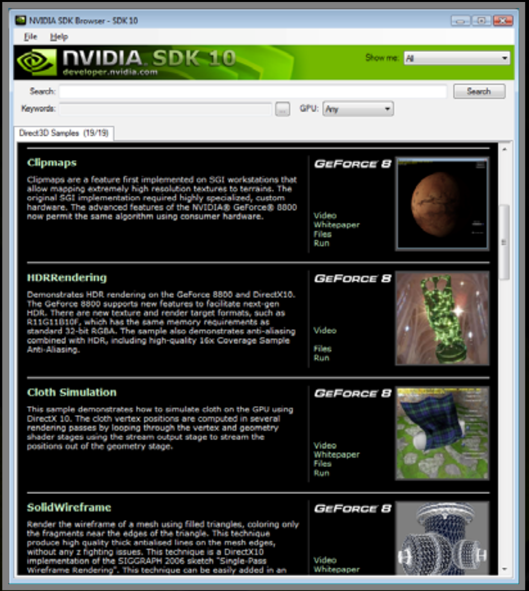
Questions?

- Developer tools DVDs available at our booth
  - PerfKit 2.2
  - PerfHUD 4 Overview Video
  - PerfHUD 4 Quick Reference Card
  - ShaderPerf 2.0 Alpha
  - User Guides
- Online:
  - http://developer.nvidia.com/NVPerfKit
  - http://developer.nvidia.com/NVPerfHUD
  - http://developer.nvidia.com/NVShaderPerf
- Feedback and Support:

  NVPerfKit@nvidia.com
  NVPerfHUD@nvidia.com
  NVShaderPerf@nvidia.com
  FXComposer@nvidia.com

Six All-New NVIDIA Developer Tools!

Check it out at booth #5134!



Clipmaps

Clipmaps are a feature first implemented on SGI workstations that allow mapping extremely high resolution textures to terrains. The original SGI implementation required highly specialized, custom hardware. The advanced features of the NVIDIA® GeForce® 8800 now permit the same algorithm using consumer hardware.

GeForce 8 — Video, Whitepaper, Files, Run

HDRRendering

Demonstrates HDR rendering on the GeForce 8800 and DirectX10. The GeForce 8800 supports new features to facilitate next-gen HDR. There are new texture and render target formats, such as R11G11B10F, which has the same memory requirements as standard 32-bit RGBA. The sample also demonstrates anti-aliasing combined with HDR, including high-quality 16x Coverage Sample Anti-Aliasing.

GeForce 8 — Video, Files, Run

Cloth Simulation

This sample demonstrates how to simulate cloth on the GPU using DirectX 10. The cloth vertex positions are computed in several rendering passes by looping through the vertex and geometry shader stages using the stream output stage to stream the positions out of the geometry stage.

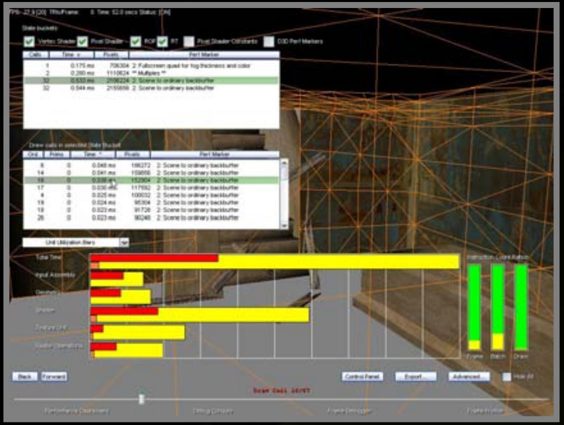
GeForce 8 — Video, Whitepaper, Files, Run

SolidWireframe

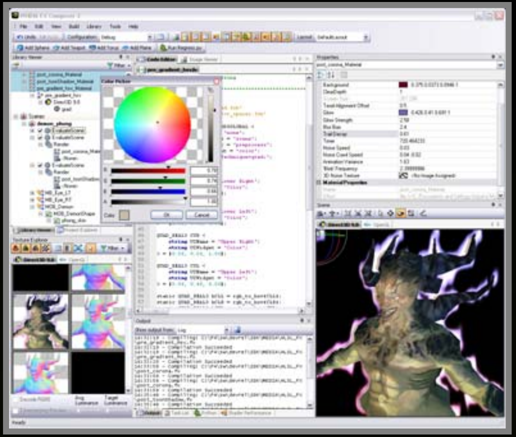
Render the wireframe of a mesh using filled triangles, coloring only the fragments near the edges of the triangle. This technique produce high quality thick antialised lines on the mesh edges, without any z fighting issues. This technique is a DirectX10 implementation of the SIGGRAPH 2006 sketch "Single-Pass Wireframe Rendering". This technique can be easily added in an

GeForce 8 — Video, Whitepaper

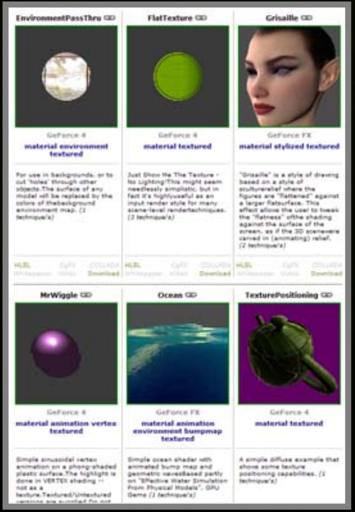
SDK 10



PerfKit 5



FX Composer 2



EnvironmentPassThru

GeForce 4

material environment textured

For use in backgrounds, or to cut 'holes' through other objects.The surface of any model will be replaced by the colors of thebackground environment map. (1 technique's)

FlatTexture

GeForce 4

material textured

Just Show Me The Texture - No Lighting!This might seem needlessly simplistic, but in fact it's highlyuseful as an input render style for many scene-level rendertechniques. (1 technique's)

Grisaille

GeForce FX

material stylized textured

"Grisaille" is a style of drawing based on a style of sculturerelief where the figures are "flattened" against a larger flatsurface. This effect allows the user to tweak the "flatness" ofthe shading against the surface of the screen, as if the 3D scenewere carved in (animating) relief. (2 technique's)

MrWiggle

GeForce 4

material animation vertex textured

Simple sinusoidal vertex animation on a phong-shaded plastic surface.The highlight is done in VERTEX shading -- not as a texture.Textured/Untextured versions are supplied fo

Ocean

GeForce FX

material animation environment bumpmap textured

Simple ocean shader with animated bump map and geometric wavesBased partly on "Effective Water Simulation From Physical Models", GPU Gems (1 technique's)

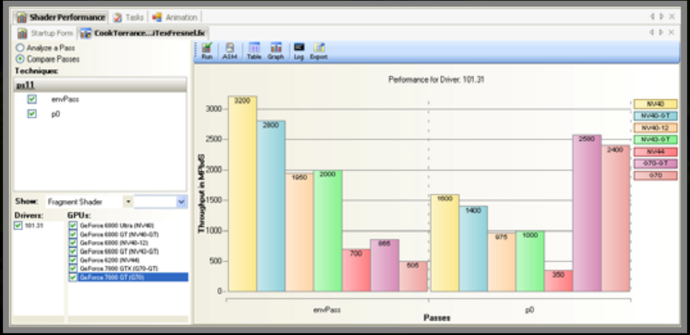
TexturePositioning

GeForce 4

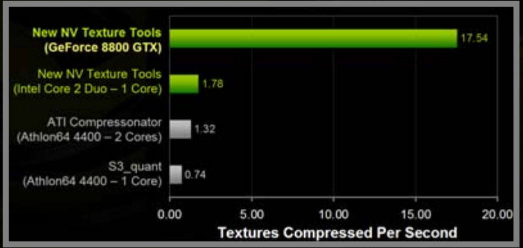
material textured

A simple diffuse example that shows some texture positioning capabilities. (1 technique's)

Shader Library



ShaderPerf 2



GPU-Accelerated Texture Tools