



NVIDIA®

Performance Tools

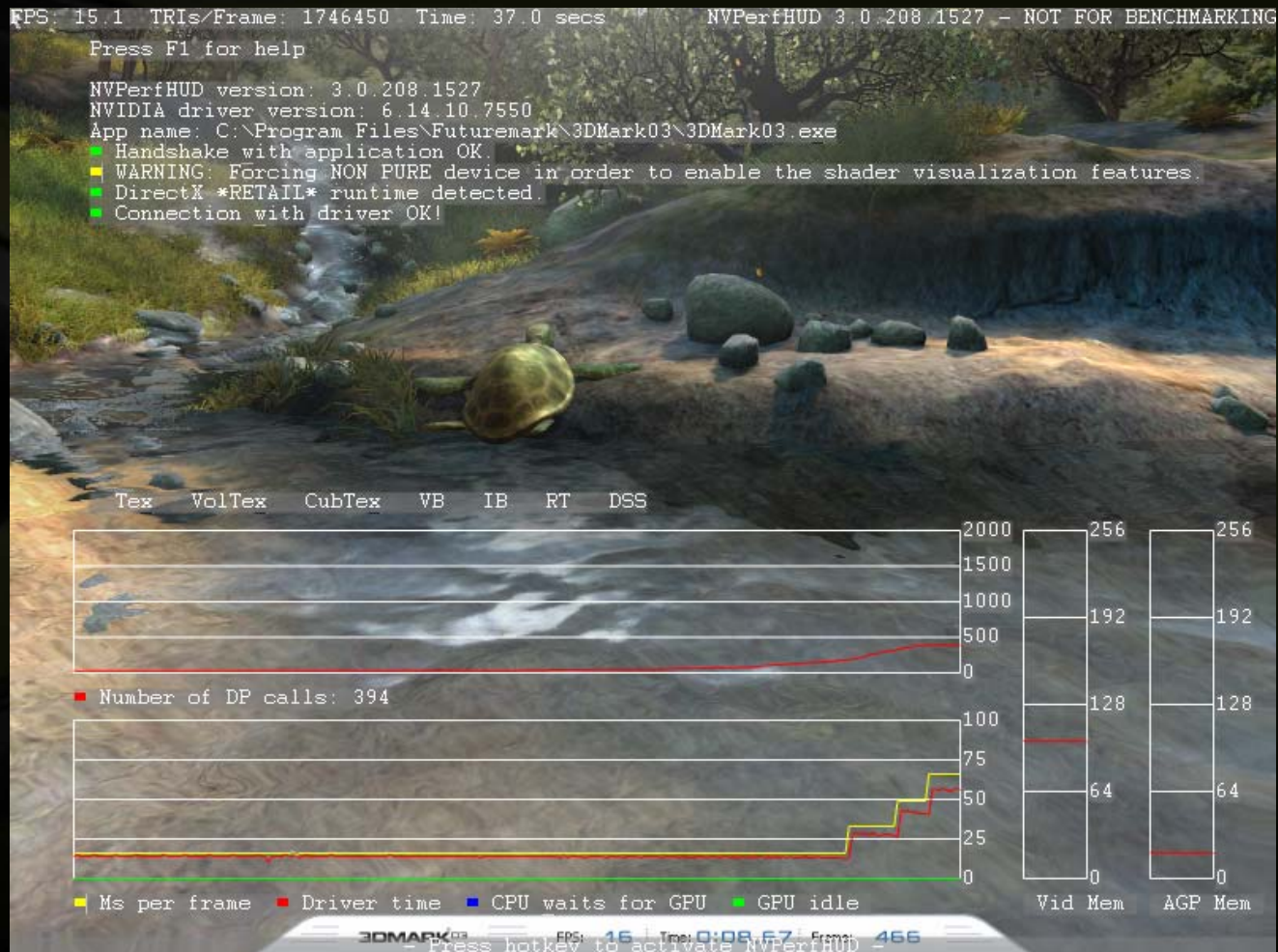
Raul Aguaviva, Sim Dietrich, and Sébastien Dominé

Agenda

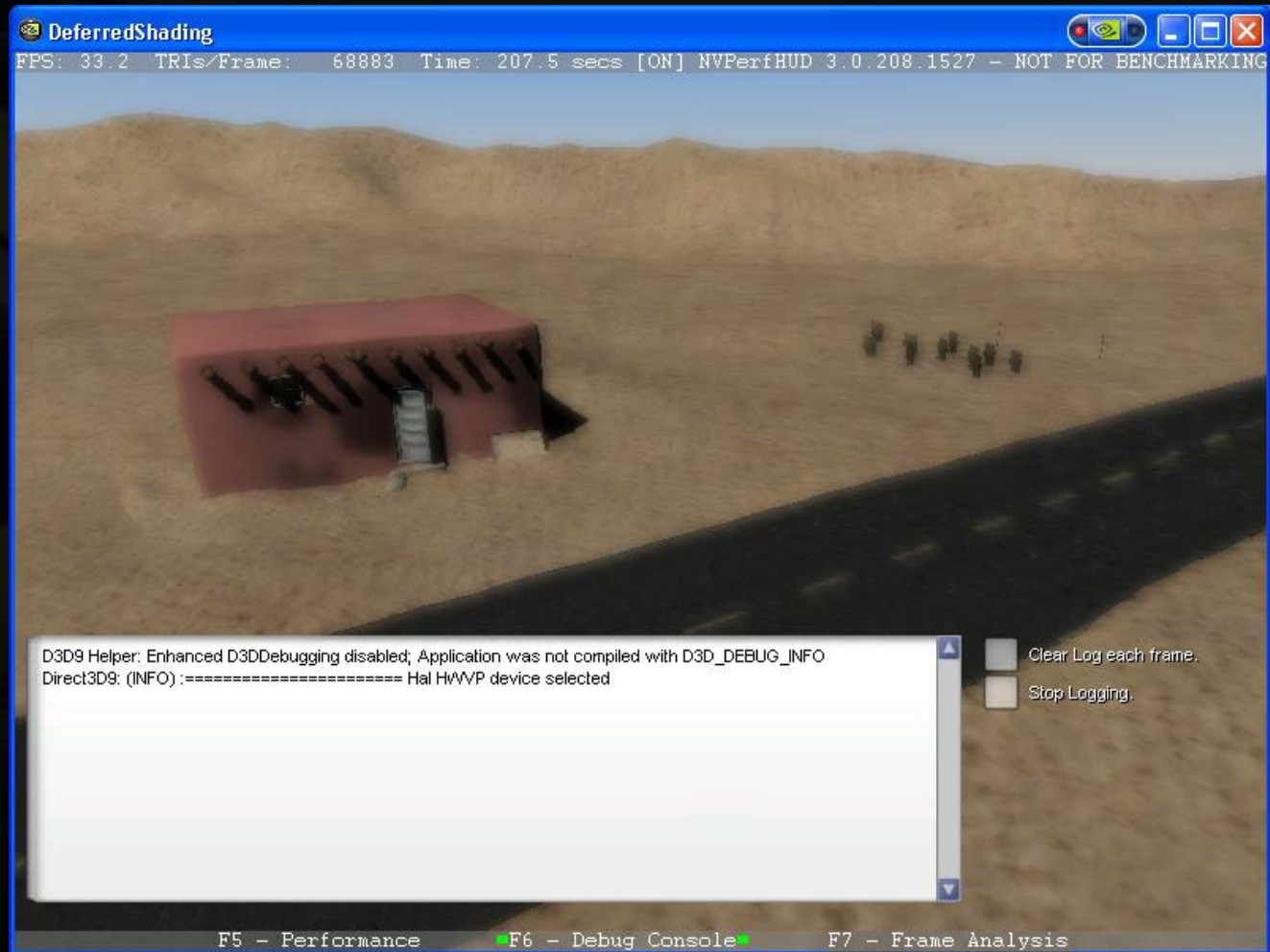


- **NVPerfHUD 3.0**
 - The tool
 - In real life
- **NVPerfKIT 1.0**
- **NVShaderPerf 71.84**
- **FX Composer 1.6**
- **Conclusion**
- **Q&A**

NVPerfHUD 3.0



Debug Console



Frame Analysis



DeferredShading FPS: 53.9 TRIs/Frame: 68883 Time: 732.2 secs [ON] NVPerfHUD 3.0.208.1527 - NOT FOR BENCHMARKING

Top 10 Warning List

Texture Name	Dimensions	Format	MIPs	Mag	Min	MIP
D3DRTYPE_TEXTURE	256x256	D3DFMT_DXT3	9	LINEAR	LINEAR	LINEAR
Not used						
RD3DRTYPE_TEXTURE	791x565	D3DFMT_A8R8G8B8	1			
RD3DRTYPE_TEXTURE	791x565	D3DFMT_R32F	1			
Not used						
RD3DRTYPE_TEXTURE	791x565	D3DFMT_A8R8G8B8	1			
Not used						

Prims Drawn: 68219
DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 211, 0, 380)
RT: 0x00aed730 BB: 0x001565a0
DP Pointer 220 / 247

Step Back Step Forward Advanced

Jump to Warnings
Force Clear
Wireframe
Depth Complexity

F5 - Performance F6 - Debug Console F7 - Frame Analysis

Advanced - Index Unit

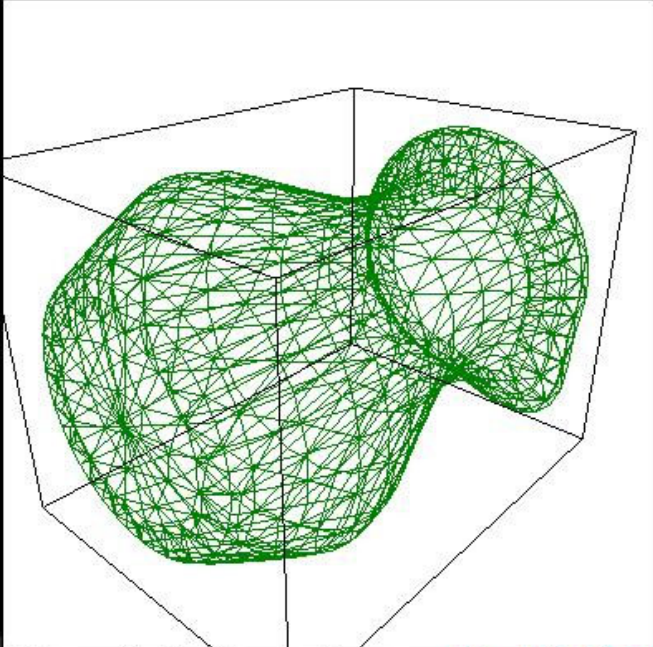


DeferredShading

FPS: 57.2 TRIs/Frame: 68883 Time: 938.3 secs [ON] NVPerfHUD 3.0.208.1527 - NOT FOR BENCHMARKING

Index Unit Vertex Shader Pixel Shader Raster Operation

Wireframe



Index / Vertex Buffer

```
** DP Info **
DrawIndexedPrimitive:
Type: D3DPT_TRIANGLELIST
BaseVertexIndex: 0
MinVertexIndex: 0
NumVertices: 783
startIndex: 0
primCount: 1484
HRESULT: 0x00000000
Msg: S_OK
Desc: The function completed successful

** Index Buffer Description **
Format: D3DFMT_INDEX16
Pool: D3DPool_MANAGED
Usage: D3DUSAGE_WRITEONLY
Length: 8904 bytes

** VB Declaration **
Total vertex size: 72
0 - POSITION FLOAT3 DEFAULT
0 - NORMAL FLOAT3 DEFAULT
0 - COLOR D3DCOLOR DEFAULT
0 - TEXCOORD FLOAT2 DEFAULT
0 - TEXCOORD FLOAT3 DEFAULT
0 - TEXCOORD FLOAT3 DEFAULT
```

Step Back Step Forward Simple DP Pointer 176 / 247

F5 - Performance F6 - Debug Console F7 - Frame Analysis

Advanced – Vertex Shader



DeferredShading

FPS: 57.7 TRIs/Frame: 68883 Time: 1054.4 secs [ON]NVPerfHUD 3.0.208.1527 - NOT FOR BENCHMARKING

Index Unit Vertex Shader Pixel Shader Raster Operation

Vertex Shader

```
// c2 = { 0, 0, 0, 0 };  
// c3 = { 0, 0, 0, 0 };  
//  
// WorldViewIT  
// c4 = { 0, 0, 0, 0 };  
// c5 = { 0, 0, 0, 0 };  
// c6 = { 0, 0, 0, 0 };  
//  
  
vs_1_1  
dcl_position v0  
dcl_normal v1  
dcl_texcoord v2
```

Textures

Sampler: s0
D3DRTYPE_TEXTURE 256x256
D3DFMT_DXT3
MIPs:8
Mag:LINEAR Min:LINEAR MIP:LINEAR

Vertex Shader Constants

Floating Point Constants:

```
C[ 0]: -1.130298 0.503030 0.0000  
C[ 1]: 0.063986 0.143776 1.7248  
C[ 2]: 0.404976 0.909972 -0.0906  
C[ 3]: 0.404912 0.909830 -0.0906  
C[ 4]: -0.913609 0.406594 0.0000  
C[ 5]: 0.036943 0.083009 0.9958  
C[ 6]: 0.404912 0.909830 -0.0906
```

Step Back Step Forward Simple DP Pointer 161 / 247

F5 - Performance F6 - Debug Console F7 - Frame Analysis

Advanced – Pixel Shader



DeferredShading

FPS: 48.7 TRIs/Frame: 68883 Time: 1338.9 secs [ON]NVPerfHUD 3.0.208.1527 – NOT FOR BENCHMARKING

Index Unit Vertex Shader **Pixel Shader** Raster Operation

Pixel Shader

```
// Registers:  
//  
// Name      Reg  Size  
// -----  
// DiffuseMapSampler s0  1  
//  
  
ps_2_0  
def c1, 1, 0.5, 0, 0  
dcl t0.xy  
dcl_2d s0  
texdd_pp r0, t0, s0  
mov r0.w, c1.x
```

Pixel Shader Constants

Floating Point Constants:
C[0]: 0.000000 0.000000 0.000000
C[1]: 0.000000 0.000000 0.000000

Integer Constants:
Boolean Constants:

Textures

Sampler: s0
D3DRTYPE_TEXTURE 1024x512
D3DFMT_DXT3
MIPs: 11
Mag: LINEAR Min: LINEAR MIP: LINEAR

Step Back Step Forward Simple DP Pointer 164 / 247

F5 – Performance F6 – Debug Console F7 – Frame Analysis

Advanced – Raster Operation



DeferredShading

FPS: 33.5 TRIs/Frame: 68883 Time: 1640.2 secs [ON]NVPerfHUD 3.0.208.1527 - NOT FOR BENCHMARKING

Index Unit Vertex Shader Pixel Shader Raster Operation

Render Targets and Render States

**** Back Buffer Information ****
Back Buffer size: 791x565
BackBufferFormat: D3DFMT_X8R8G8B8
Back Buffer count: 2
MultiSampleType: D3DMULTISAMPLE_NONE
MultiSampleQuality: 0
AutoDepthStencilFormat: D3DFMT_D24X8
FullScreen_RefreshRateInHz: 0
vWindowed: 1

**** RenderStates Dump ****
ZENABLE = D3DZB_FALSE
FILLMODE = D3DFILL_SOLID
SHADEMODE = D3DSHADE_GOURAUD
ZWRITEENABLE = TRUE
ALPHATESTENABLE = FALSE
LASTPIXEL = TRUE
SRCBLEND = D3DBLEND_SRC
DESTBLEND = D3DBLEND_INV
CULLMODE = D3DCULL_CCW
ZFUNC = D3DCMP_LESSEQUAL
ALPHAREF = 0
ALPHAFUNC = D3DCMP_ALWAYS
DITHERENABLE = FALSE
ALPHABLENDENABLE = TRUE
FOGENABLE = FALSE

Step Back Step Forward Simple DP Pointer 247 / 247

F5 - Performance F6 - Debug Console F7 - Frame Analysis

NVPerfHUD 3.0 – Freezing the game



- Only possible if game uses time-based animation
- Stop the clock
 - Intercept: `QueryPerformanceCounter(...)`
`timeGetTime(...)`
 - No RDTSC
- $\text{Pos} += V * \text{DeltaTime}$
- When DeltaTime is 0
 - Don't divide by DeltaTime
 - Don't skip Presents()

Coming Soon!



- **NVPerfHUD 4.0**
 - **Pipeline utilization graph**
 - **Automated Bottleneck Identification**

NVPerfHUD Schedule



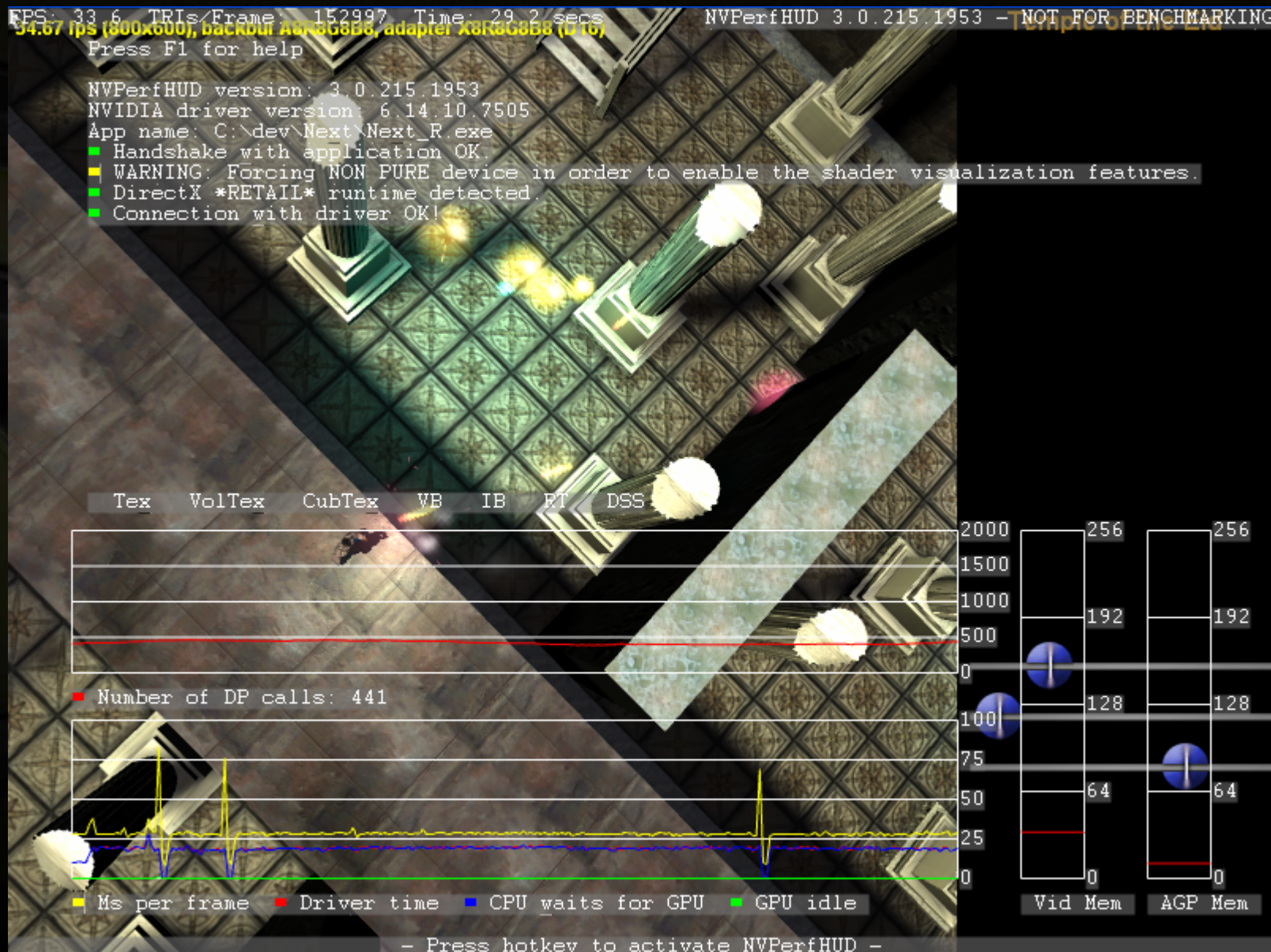
● NVPerfHUD 3.0

- Registered developer Beta 2 : now
- Registered developer Beta 3 : 3/16/2005
- Final Release : 3/23/2005

● NVPerfHUD 4.0

- Registered developer Beta: 5/1/2005
- Final Release : 5/15/2005

NVPerfHUD in real life...



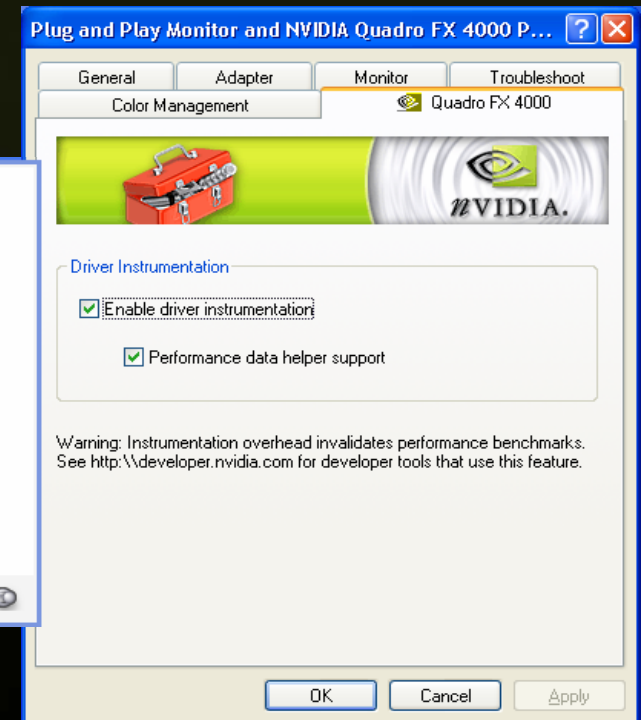
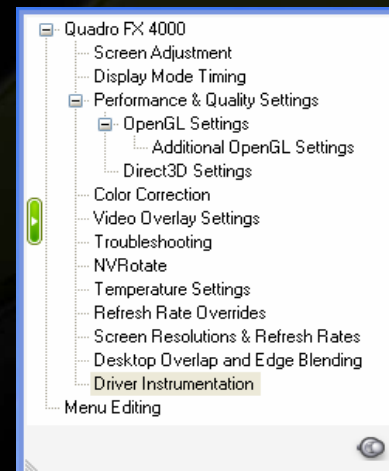


- **Complete Performance Instrumentation Solution**
 - Instrumented Driver
 - NVIDIA Developer Control Panel
 - Plug-in for Microsoft PIX for Windows
 - Support for PDH (Performance Data Helper)
 - Code samples for OpenGL and Direct3D
 - Secure mechanism for authorizing applications to be instrumented

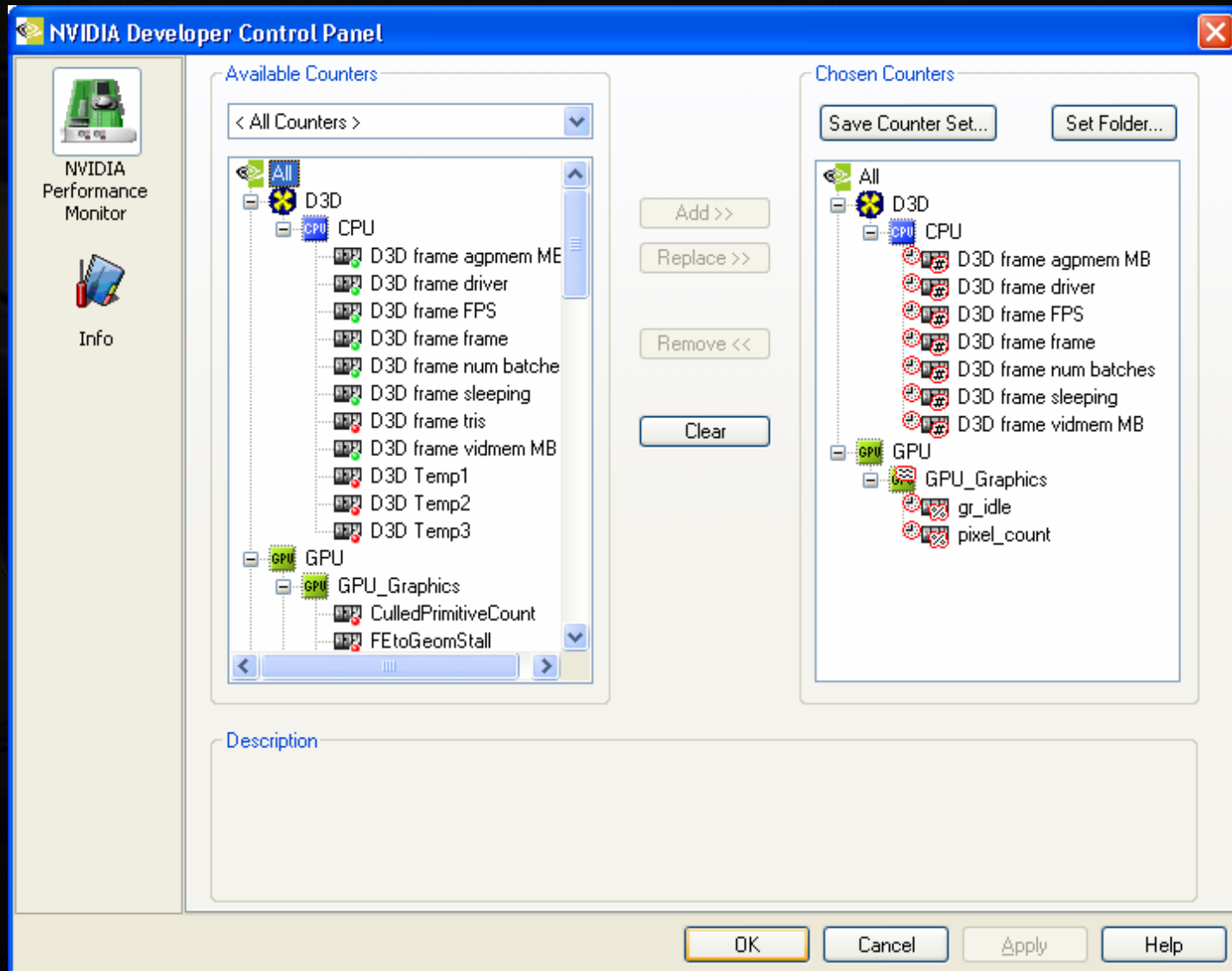


● Instrumented Driver

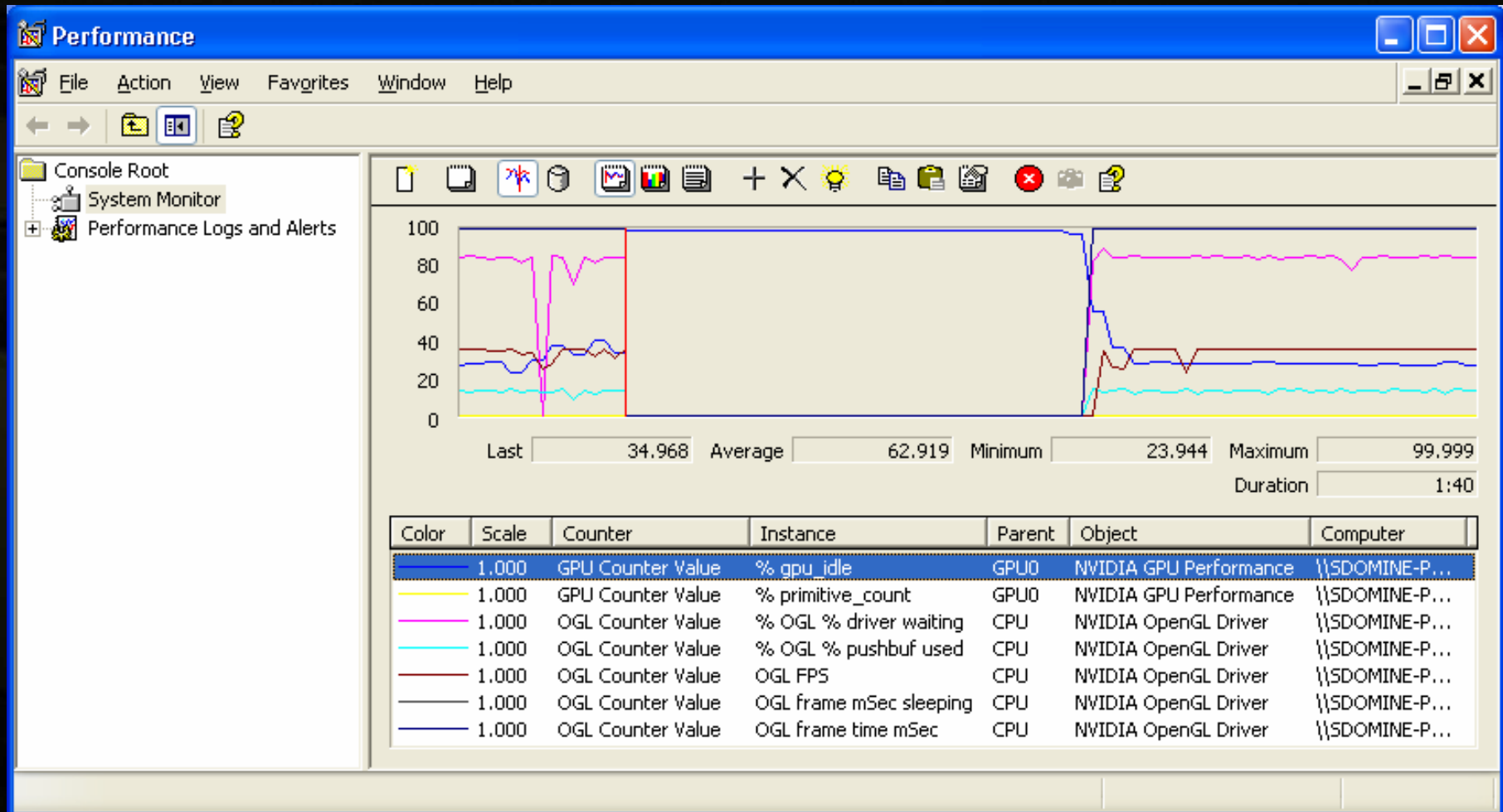
- Exposes GPU and Driver Performance Counters
- Supports OpenGL and Direct3D
- Supports SLI Counters
- Requires GeForce FX or 6 Series



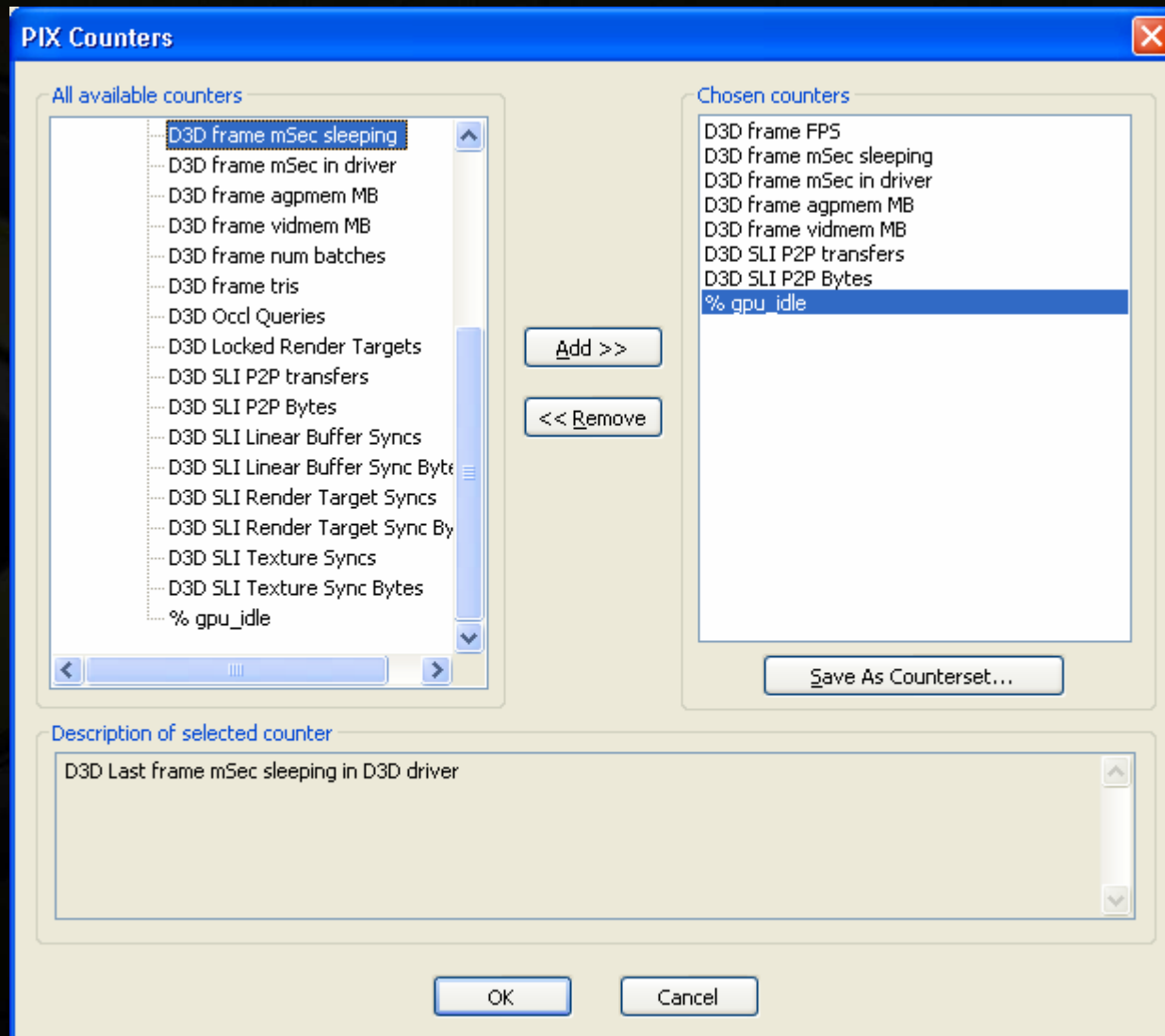
NVPerfKIT NVIDIA Developer Control Panel



NVPerfKIT Performance Data Helper



NVPerfKIT Plugin for Microsoft PIX





● Schedule

- Registered Developer - Early Beta Access : 3/16/2005
- Registered Developer - Beta : 3/23/2005
- Registered Developer - Release : 4/7/2005

```
v2f BumpReflectVS(a2v IN,
```

```
uniform float4x4 WorldViewProj,  
uniform float4x4 World,  
uniform float4x4 ViewIT)
```

NVShaderPerf



```
v2f OUT;  
// Position in object 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 transform 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.TexCoord3.x = dot(World[0].xyz, TangentToObjSpace[0]);  
OUT.TexCoord3.y = dot(World[1].xyz, TangentToObjSpace[0]);  
OUT.TexCoord3.z = dot(World[2].xyz, TangentToObjSpace[0]);
```

```
OUT.TexCoord3.x = dot(World[0].xyz, TangentToObjSpace[1]);  
OUT.TexCoord3.y = dot(World[1].xyz, TangentToObjSpace[1]);  
OUT.TexCoord3.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]);
```

```
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 vector pointing from shaded point to eye in cube space  
float3 eyeVec = mul(worldPos, ViewIT[3]); // view inv. transpose contains eye position in world space
```

```
OUT.TexCoord3.w = eyeVec.x;  
OUT.TexCoord3.y = eyeVec.y;  
OUT.TexCoord3.z = eyeVec.z;
```

```
return OUT;
```

```
////////// Shader Inputs
```

```
float4 BumpReflectVS(a2v IN,
```

```
uniform float4x4 WorldViewProj,
```

```
uniform float4x4 World,
```

```
uniform float4x4 ViewIT)
```

```
uniform float BumpScale;
```

```
uniform samplerCUBE EnvironmentMap;
```

```
uniform float BumpScale; : COLOR
```

```
// fetch the bump normal from the normal map  
float3 normal = tex2D(NormalMap, IN.TexCoord.xy).xyz * 2.0 - 1.0;
```

```
normal = normalize(float3(normal.x * BumpScale, normal.y * BumpScale, normal.z));
```

```
// transform the bump normal into cube space
```

```
// then use the transformed normal and eye vector to compute a reflection vector
```

```
// used to fetch the cube map
```

```
// (we multiply by 2 only to increase
```

```
float3 eyeVec = float3(IN.TexCoord3.x, IN.TexCoord3.y, IN.TexCoord3.z);
```

```
float3 worldNorm;
```

```
worldNorm.x = dot(IN.TexCoord3.x, eyeVec);
```

```
worldNorm.y = dot(IN.TexCoord3.y, eyeVec);
```

```
worldNorm.z = dot(IN.TexCoord3.z, eyeVec);
```

```
float3 lookup = reflect(eyeVec, worldNorm);
```

```
return texCUBE(EnvironmentMap, lookup);
```

```
}
```

GPU Arch:

- GeForce FX
- GeForce 6 Series
- Quadro FX

NVShaderPerf

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.w, 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
```

Outputs:

- Assembly code
- # of cycles
- # of temporary registers
- Pixel throughput
- Forces all fp16 and all fp32

Target: GeForce 6800 Ultra (NV40) :: Unified Compiler: v61.7
Cycles: 14.00 :: R Regs Used: 2 :: R Regs Max Index (0 based)
Pixel throughput (assuming 1 cycle texture lookup) 304.76 M

Shader performance using all FP32
Cycles: 21.00 :: R Regs Used: 3 :: R Regs Max Index (0 based)
Pixel throughput (assuming 1 cycle texture lookup) 457.14 M

C:\Temp\NVShaderPerf_61_77>

NVShaderPerf – Coming up...

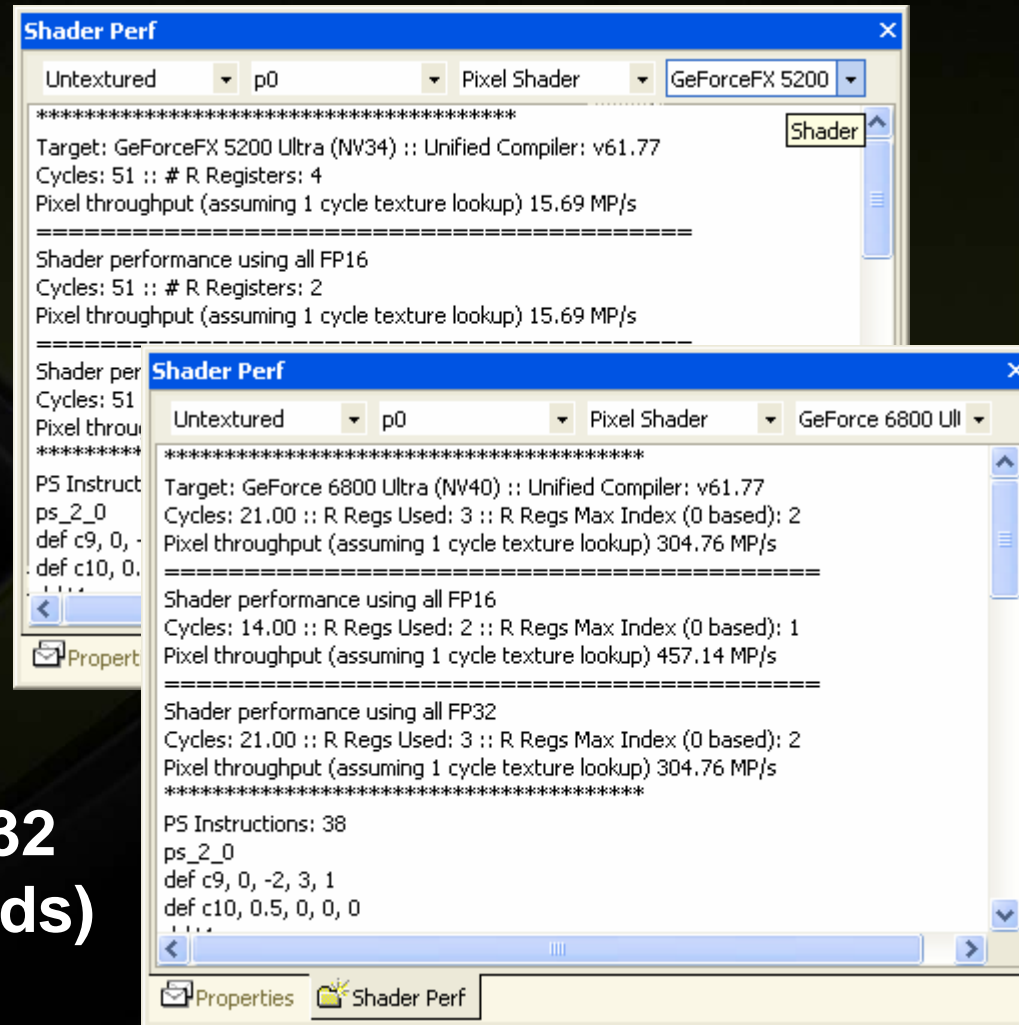


- Vertex throughput
- GLSL vertex program
- Multiple driver versions
- What else do you need?

FX Composer 1.6 – Shader Perf



- Disassembly
- Target GPU
- Driver version match
- Number of Cycles
- Number of Registers
- Pixel Throughput
- Forces all fp16 and all fp32 (gives performance bounds)



Questions?



- NVPerfHUD@nvidia.com
- FXComposer@nvidia.com
- NVshaderPerf@nvidia.com
- NVPerfKIT@nvidia.com

The Source for GPU Programming

developer.nvidia.com

- Latest News
- Developer Events Calendar
- Technical Documentation
- Conference Presentations
- GPU Programming Guide
- Powerful Tools, SDKs and more ...



nVIDIA

Join our FREE registered developer program for early access to NVIDIA drivers, cutting edge tools, online support forums, and more.

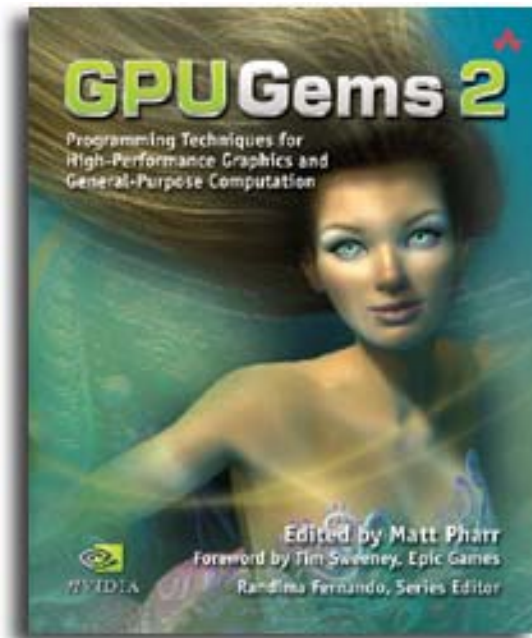
developer.nvidia.com

©2004 NVIDIA Corporation. NVIDIA, and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation. Nalu is ©2004 NVIDIA Corporation. All rights reserved.

GPU Gems 2

Programming Techniques for High-Performance Graphics and General-Purpose Computation

- 880 full-color pages
- 330 figures
- Hard cover
- \$59.99
- Experts from universities and industry



Graphics Programming



- Geometric Complexity
- Shading, Lighting, and Shadows
- High-Quality Rendering

GPGPU Programming



- General Purpose Computation on GPUs: A Primer
- Image-Oriented Computing
- Simulation and Numerical Algorithms