

OpenGL Performance Tools

Sébastien Dominé
NVIDIA Developer Tools



Agenda

- NVShaderPerf 71.84
- NVPerfKIT 1.0
- Conclusion & Q&A



|||||| pixel shader |||||||

```
float4 BumpReflectPS(v2f IN,  

    uniform sampler2D NormalMap : TEX0,  

    uniform samplerCUBE EnvironmentMap : TEX1,  

    uniform float BumpScale) : COLOR  

{  

    // fetch the bump normal from the normal map  

    float3 normal = tex2D(NormalMap, 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  

    // (we multiply by 2.0 to increase brightness)  

    float3 eyevec = float3(IN.TexCoord1.w, IN.TexCoord2.w, IN.TexCoord3.w);  

    worldNorm.x = dot(IN.TexCoord1.xyz, normal);  

    worldNorm.y = dot(IN.TexCoord2.xyz, normal);  

    worldNorm.z = dot(IN.TexCoord3.xyz, normal);  

    float3 reflect = reflect(eyevec, worldNorm);  

    float4 result = texCUBE(EnvironmentMap, reflect);  

    return result;
}
```

- GeForce FX
- GeForce 6 Series
- Quadro FX

NVShaderPerf

NVShaderPerf 71.84

Inputs:

- GLSL (fragments)
- !FP1.0
- !HARBfp1.0
- Cg

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
mul r1.xyz, r0.w, c4
add r0.w, r0.x, c9.w
mul r0.w, r0.x, c10.x
mad r0.x, r0.w, r0, c1
mad r0.xyz, r0.w, r0, c1
mad r2.xyz, r0.w, r2, c3
mul r0.w, c9.w
mov r1.w, c9.w
add r0.w, r1.w, c9.w
mul r0.w, r0.w, r0.w
```

Outputs:

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

// approximately 46 instruction slots used

```
=====  

Target: GeForce 6800 Ultra (NVIDIA) :: Unified Compiler: v61.7  

Cycles: 21.00 :: R Regs Used: 3 :: R Regs Max Index <0 based  

Pixel throughput (assuming 1 cycle texture lookup) 457.14 Mtexels/sec  

=====
```

Shader performance using all FP16

```
Cycles: 14.00 :: R Regs Used: 2 :: R Regs Max Index <0 based  

Pixel throughput (assuming 1 cycle texture lookup) 457.14 Mtexels/sec  

=====
```

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) 304.76 Mtexels/sec  

=====
```

C:\Temp\NVShaderPerf_61_77>



```
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.TexCoord1.x = dot(World[0].xyz, TangentToObjSpace[1]);  

OUT.TexCoord1.y = dot(World[1].xyz, TangentToObjSpace[1]);  

OUT.TexCoord1.z = dot(World[2].xyz, TangentToObjSpace[1]);  

OUT.TexCoord1.x = dot(World[0].xyz, TangentToObjSpace[2]);  

OUT.TexCoord1.y = dot(World[1].xyz, TangentToObjSpace[2]);  

OUT.TexCoord1.z = dot(World[2].xyz, TangentToObjSpace[2]);  

float4 worldPos = mul(IN.Position, World);  

// compute the eye vector (point from shaded point to eye) in cube space  

float4 eyeVector = worldPos - eyeVector; // view inv. transpose contains eye position  

OUT.TexCoord1.w = eyeVector.x;  

OUT.TexCoord1.w = eyeVector.y;  

OUT.TexCoord1.w = eyeVector.z;  

return OUT;
```

}

|||||| pixel shader |||||||

NVShaderPerf - Next...

- Adding more complete support for vertex programs
 - Vertex Throughput
 - GLSL Vertex Scheduling
- Support for multiple driver versions from one release



NVPerfKit 1.0

- Complete Performance Instrumentation Solution
 - Instrumented Driver
 - NVIDIA Developer Control Panel
 - Supports PDH (Performance Data Helper)
 - Code Samples for OpenGL and Direct3D
 - Requires Instrumented Applications to be Authorized



NVPerfKit

- **Instrumented Driver**
 - Exposes GPU and Driver Performance Counters
 - Supports OpenGL
 - Supports SLI Counters
 - Requires GeForce FX and 6 Series
- **NVIDIA Developer Control Panel**
 - Required to enable the counters to be sampled
 - Easy GUI to preset counter bundles



NVPerfKit

- OpenGL Driver Counters:
 - FPS & frame time (1/FPS)
 - AGP texture, VBO, and total memory used
 - Video texture, VBO, and total memory used
 - driver sleep time (driver waits for GPU),



NVPerfKit

- HW/GPU counters:
 - **gpu idle**
 - **pixel shader utilization**
 - **vertex attribute count**
 - **vertex shader utilization**
 - **texture waits for shader**
 - **shader waits for texture**
 - **shader waits for rop**
 - **FastZ utilization (UltraShadow)**
 - **pixel, vertex, triangle, primitive & culled primitive counts**



NVPerfKit

- Microsoft Performance Data Helper(PDH)
 - WMI (Windows Management and Instrumentation)
 - VTune, Perfmon,...
- Access counters in your own application

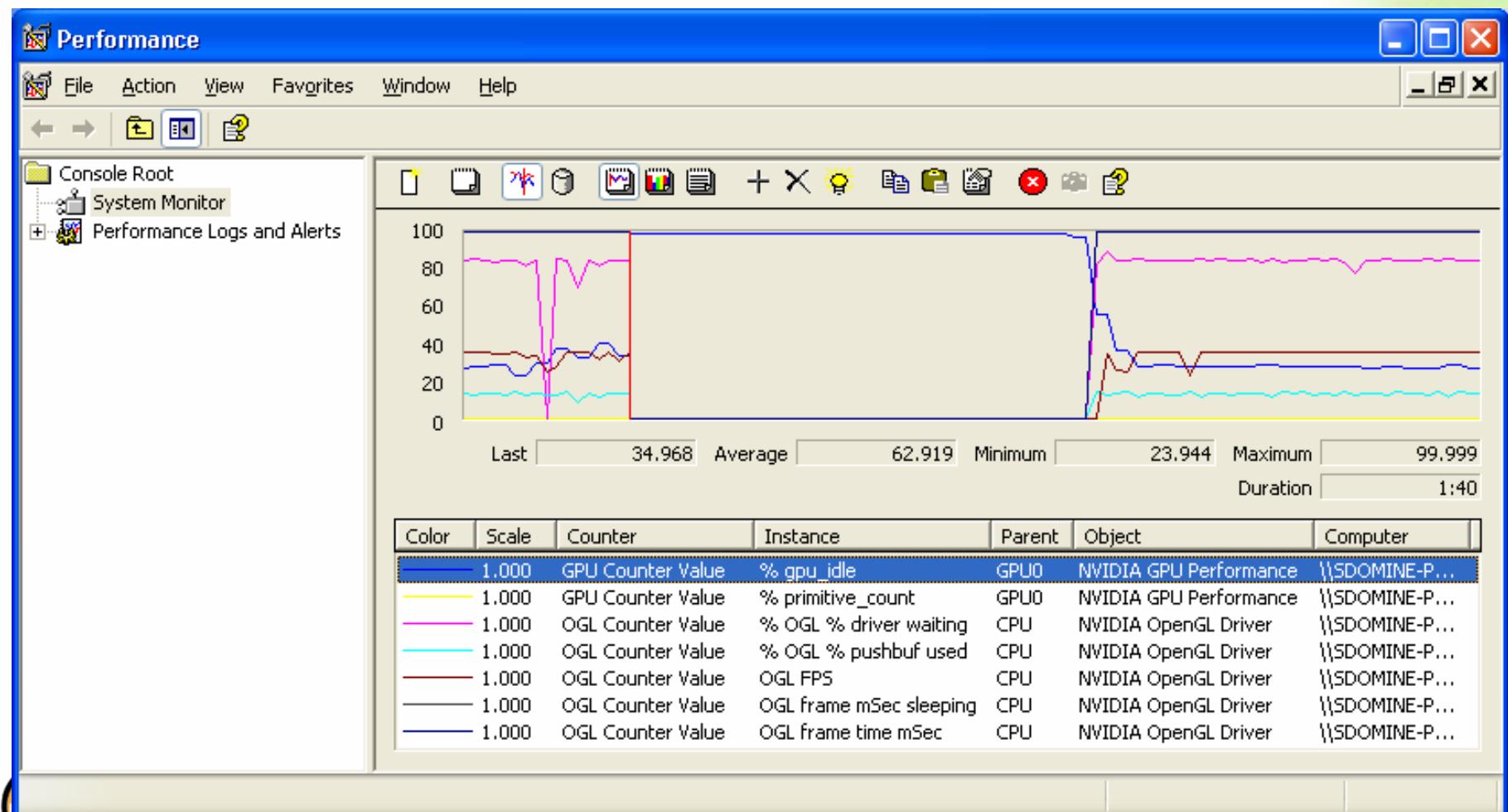
```
// Setup
PDH_HQUERY hQuery;
PDH_COUNTER hCounter;

PDH_STATUS status = PdhOpenQuery(0,0,&hQuery);
PdhAddCounter(hQuery, "\\\\'NVIDIA GPU Performance(GPU0/% gpu_idle)\\GPU Counter
Value",0,&hCounter));

// Periodically...
PDH_STATUS status = PdhCollectQueryData(hQuery);
PDH_FMT_COUNTERVALUE cvValue;
PdhGetFormattedCounterValue(hCounter,PDH_FMT_DOUBLE|PDH_FMT_NOCAP100|PDH_FMT_NOSCALE,0,&cvValue)
;
```



NvPerfKit



NVPerfKit

- Why a Security Mechanism?
 - No Reverse Engineering
 - Have to own the application's code
- How does that work?
 - Application are authenticated
 - Lightweight runtime checks
 - Heartbeat regularly checks for validity



NVPerfKit

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



Conclusion

- Shader Performance Regression Tools
- Complete Instrumentation Solution
 - NvPerfHUD for OGL
 - Just do the HUD rendering in your application
 - use PDH for the rest
- FX Composer 2.0 will support OpenGL
 - GLSL, Cg, CgFX and NVshaderPerf
- ...for everything else, there is Graphics Remedy's gDEBugger



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 ...

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



nVIDIA

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.