GameDevelopers Conference

EVOLVE GDC>04

GPU Performance Tools

Sébastien Dominé

Manager of Developer Technology Tools

NVIDIA Corporation

NVIDIA.

Agenda

- Tools of Today
  - NVPerfHUD 2.0
  - NVShaderPerf
  - FX Composer 1.1
- Tools of Tomorrow
  - Instrumented Driver
  - FX Composer 1.5
- Conclusion and Q&A

The Tools of Today

NVPerfHUD 2.0

- Overlay graph that displays stats from :
  - Direct3D9 API interception layer
  - Direct3D Driver
- Able to bypass and inject some API calls to assist with performance analysis

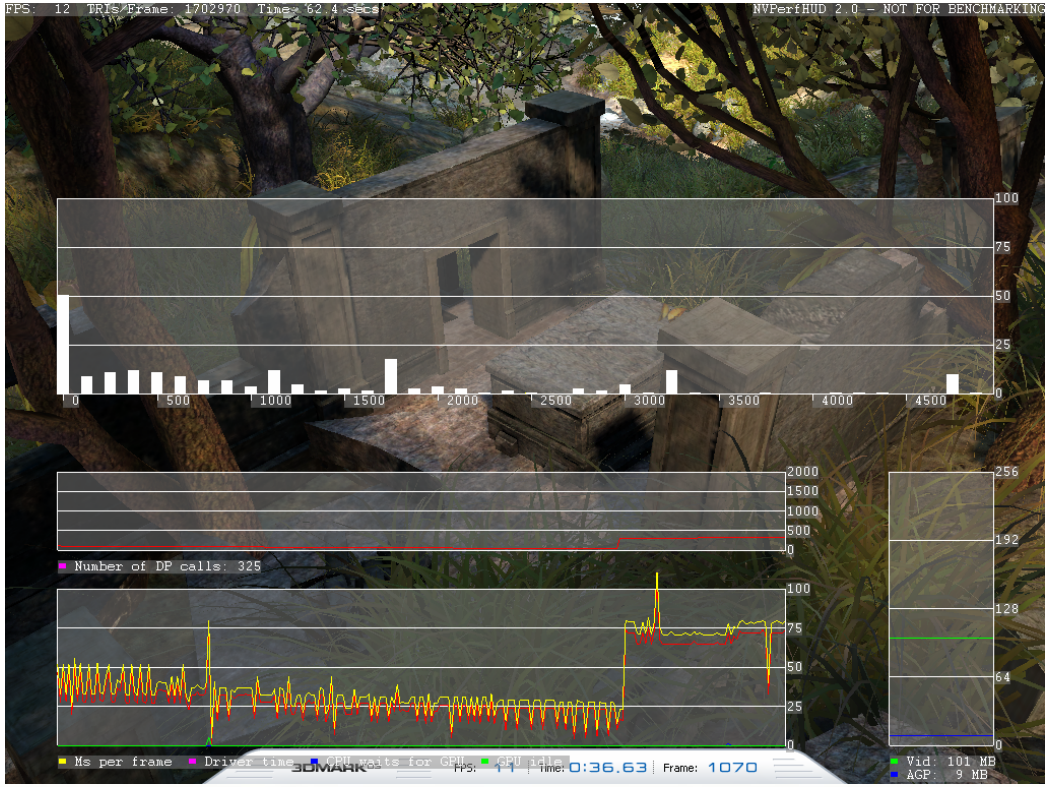


Image courtesy of FutureMark Corp.

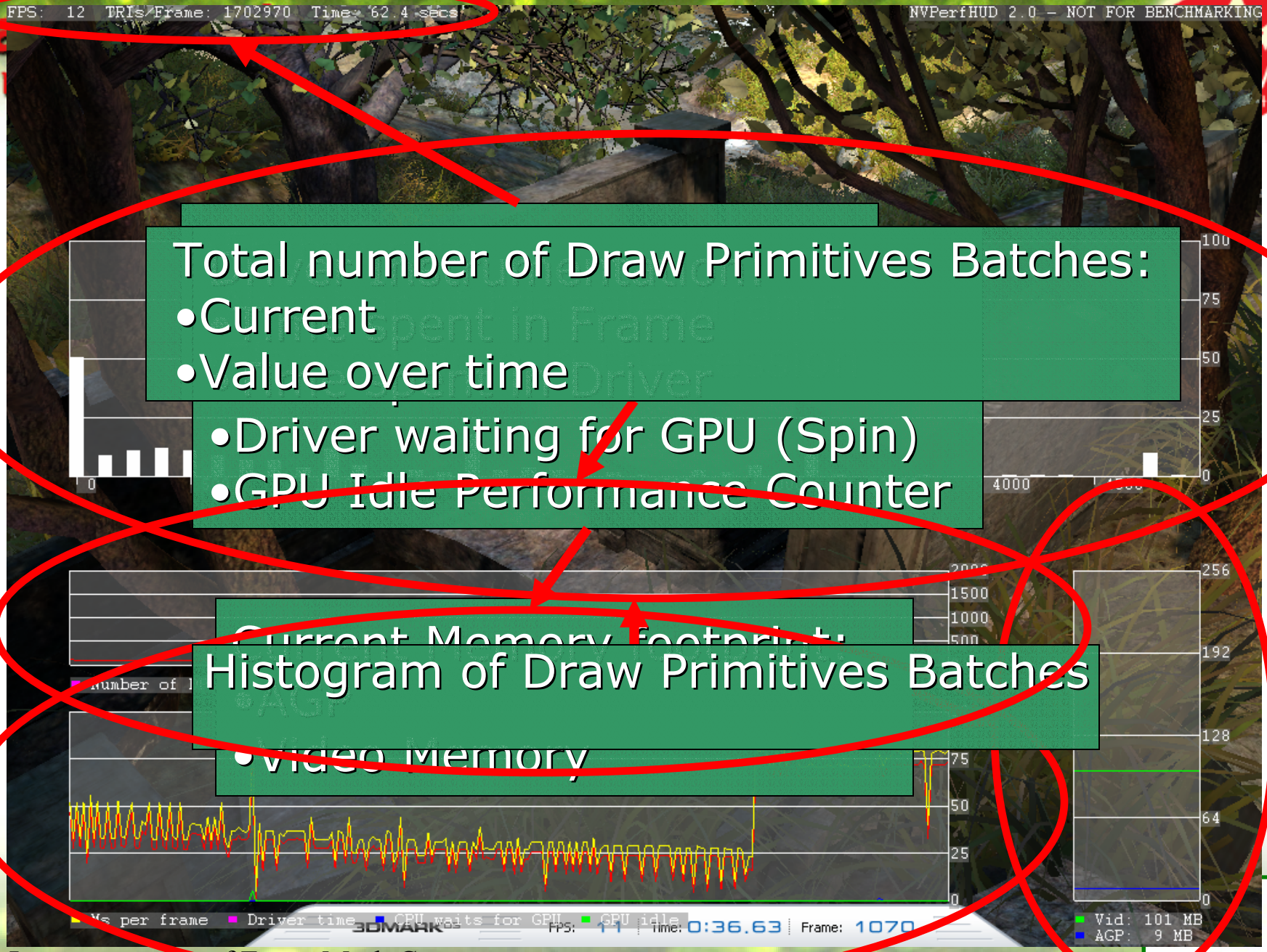
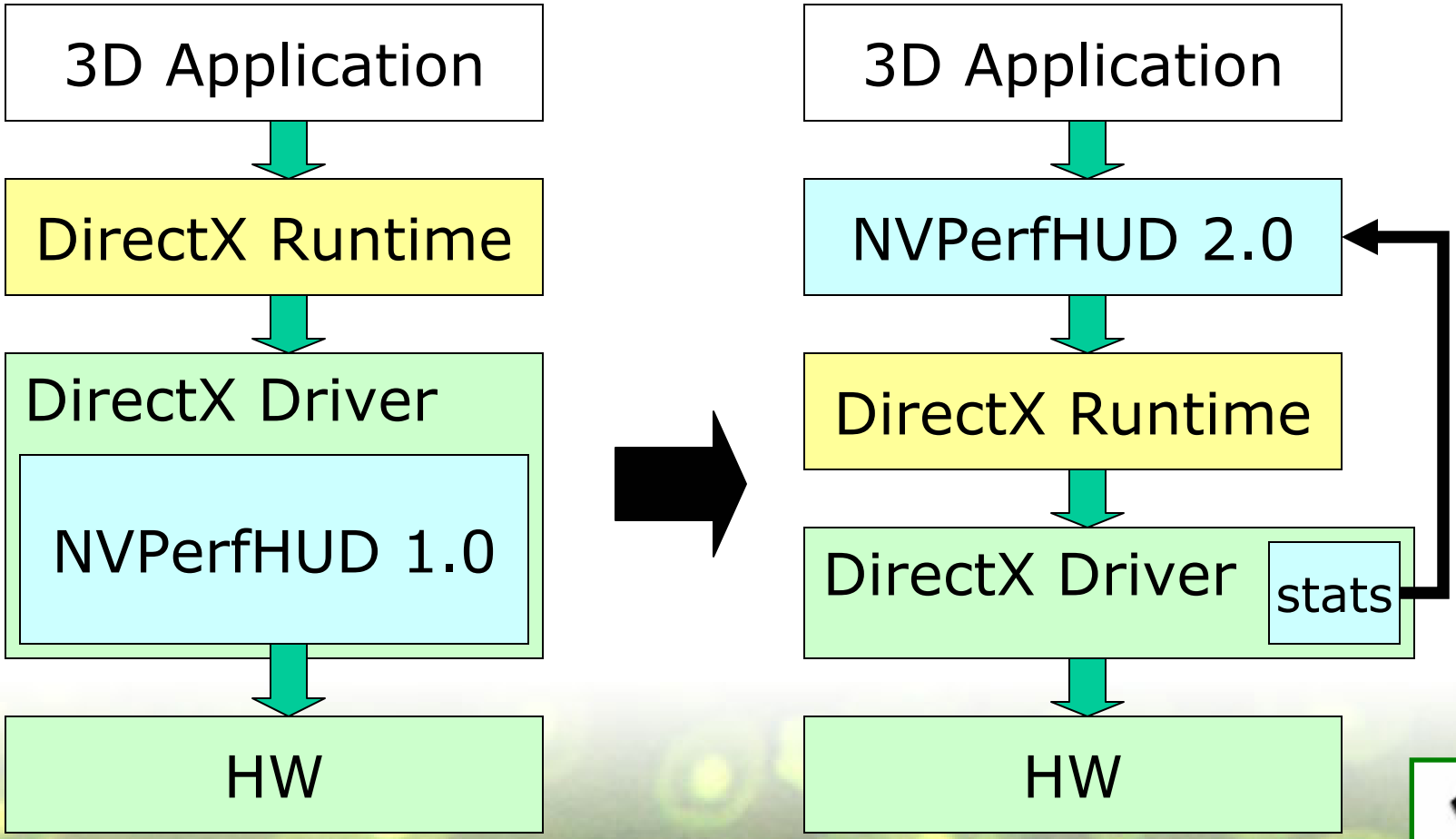
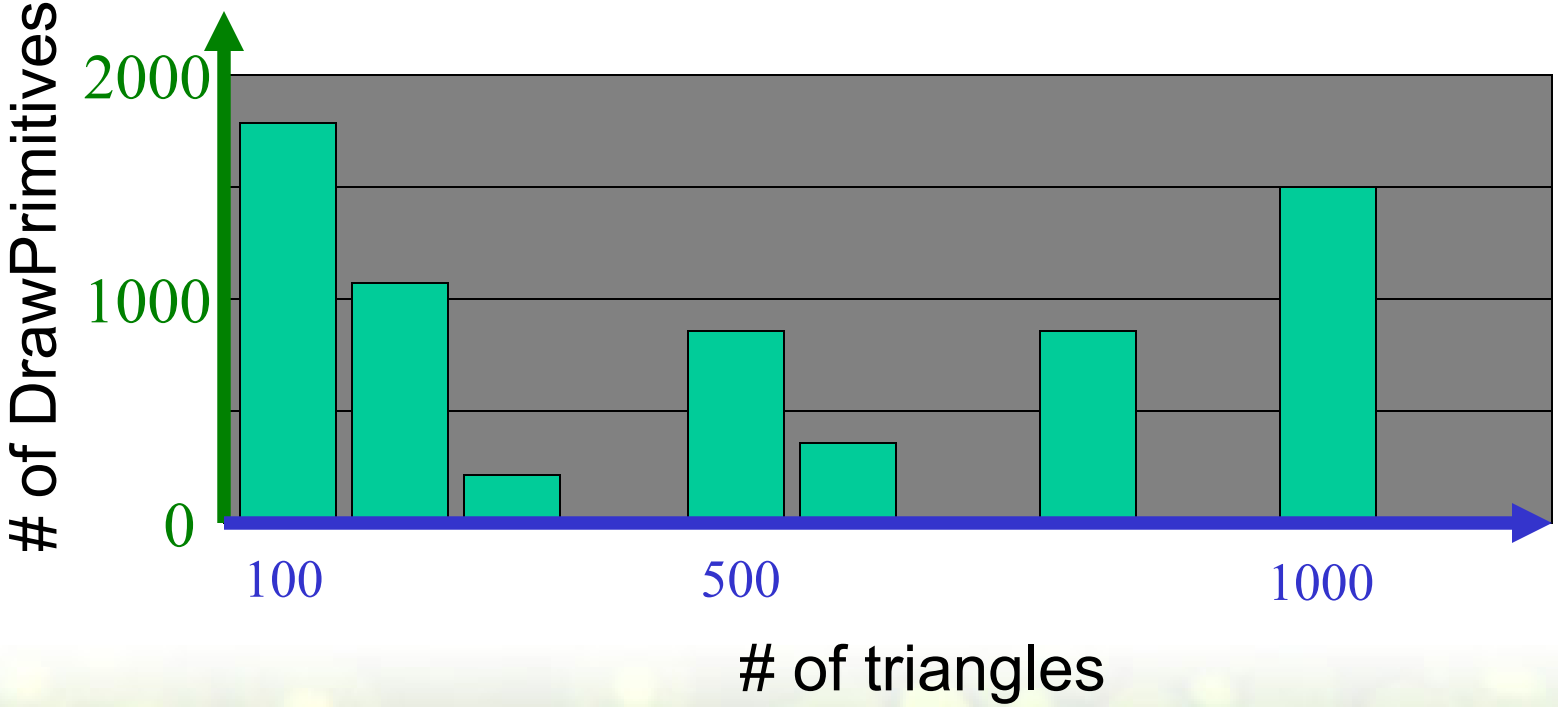


Image courtesy of FutureMark Corp.

What's new in 2.0?



- DrawPrimitives/DrawIndexPrimitives Histogram



Texture Stage States

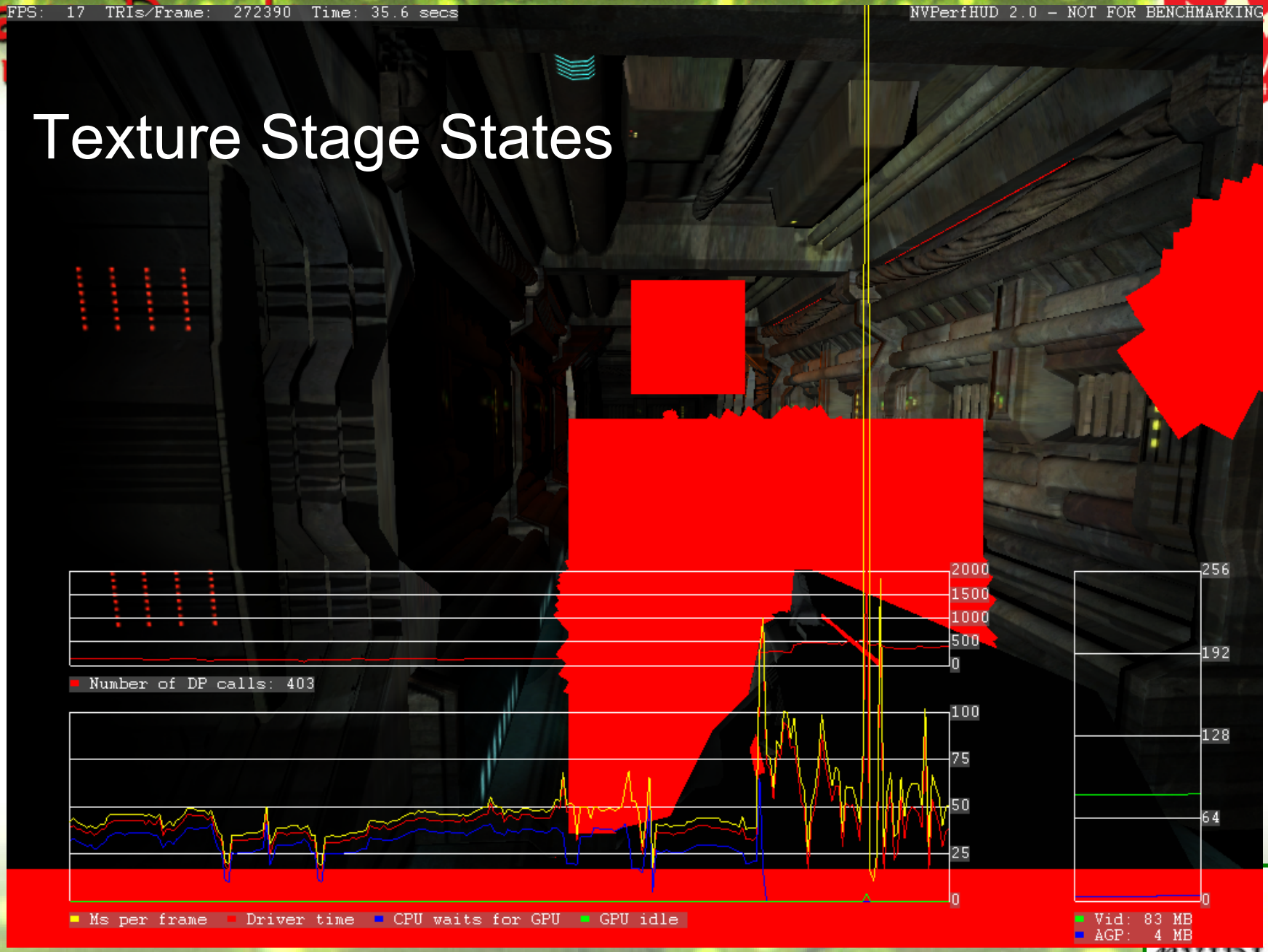


Image courtesy of FutureMark Corp.

Pixel Shaders 1.x

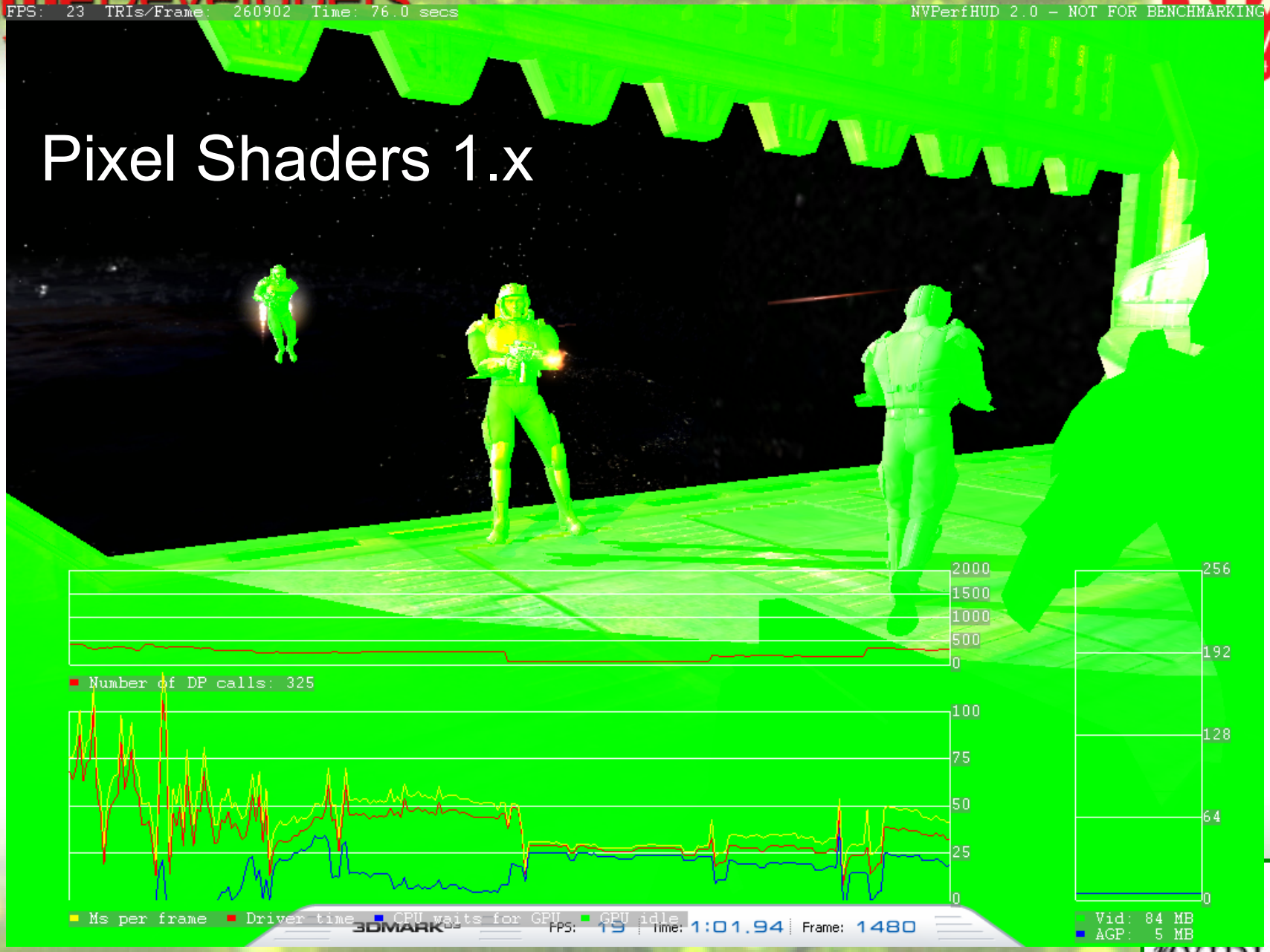


Image courtesy of FutureMark Corp.

Pixel Shaders 2.0

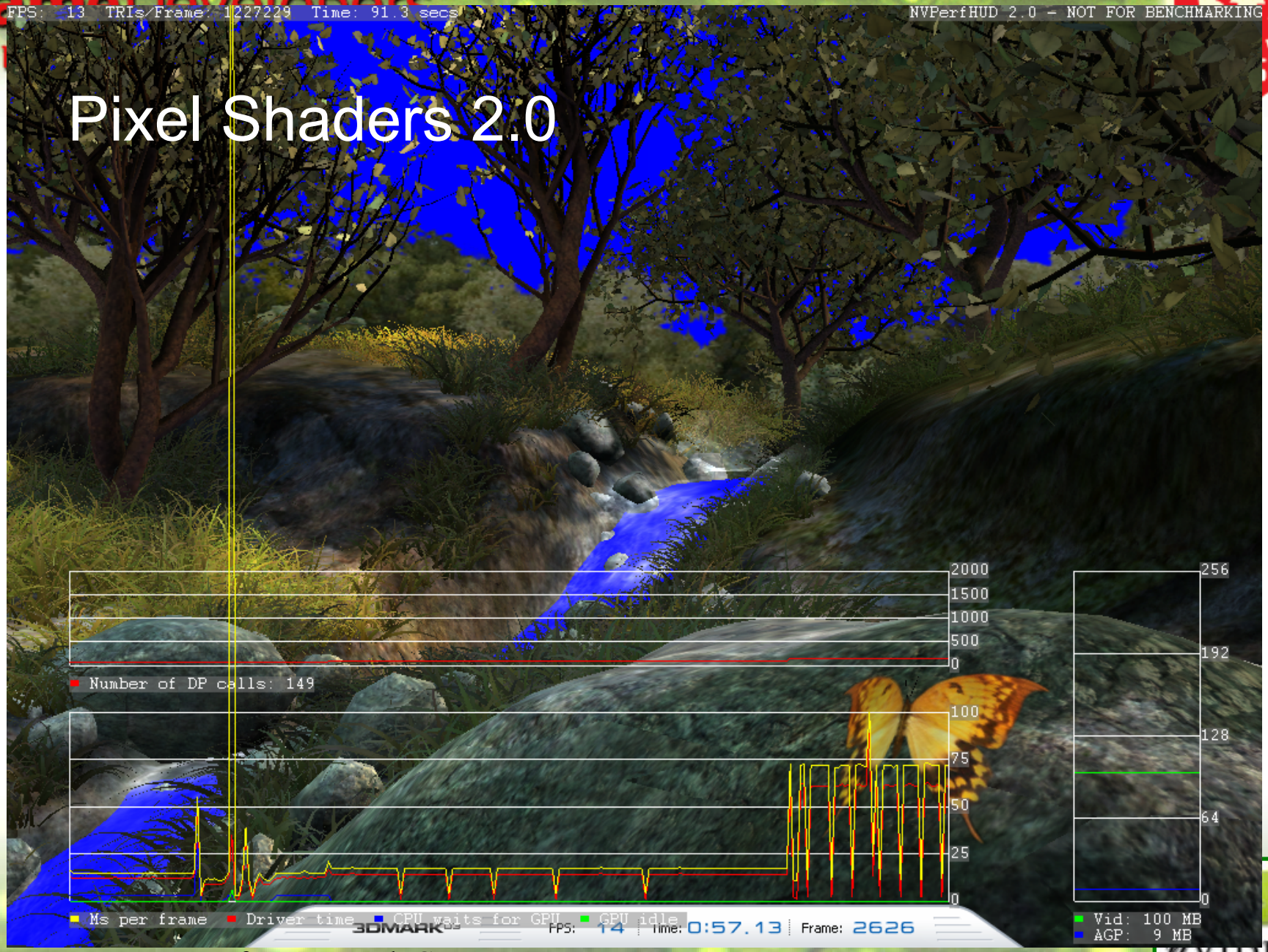


Image courtesy of FutureMark Corp.

2x2 Texture replacement

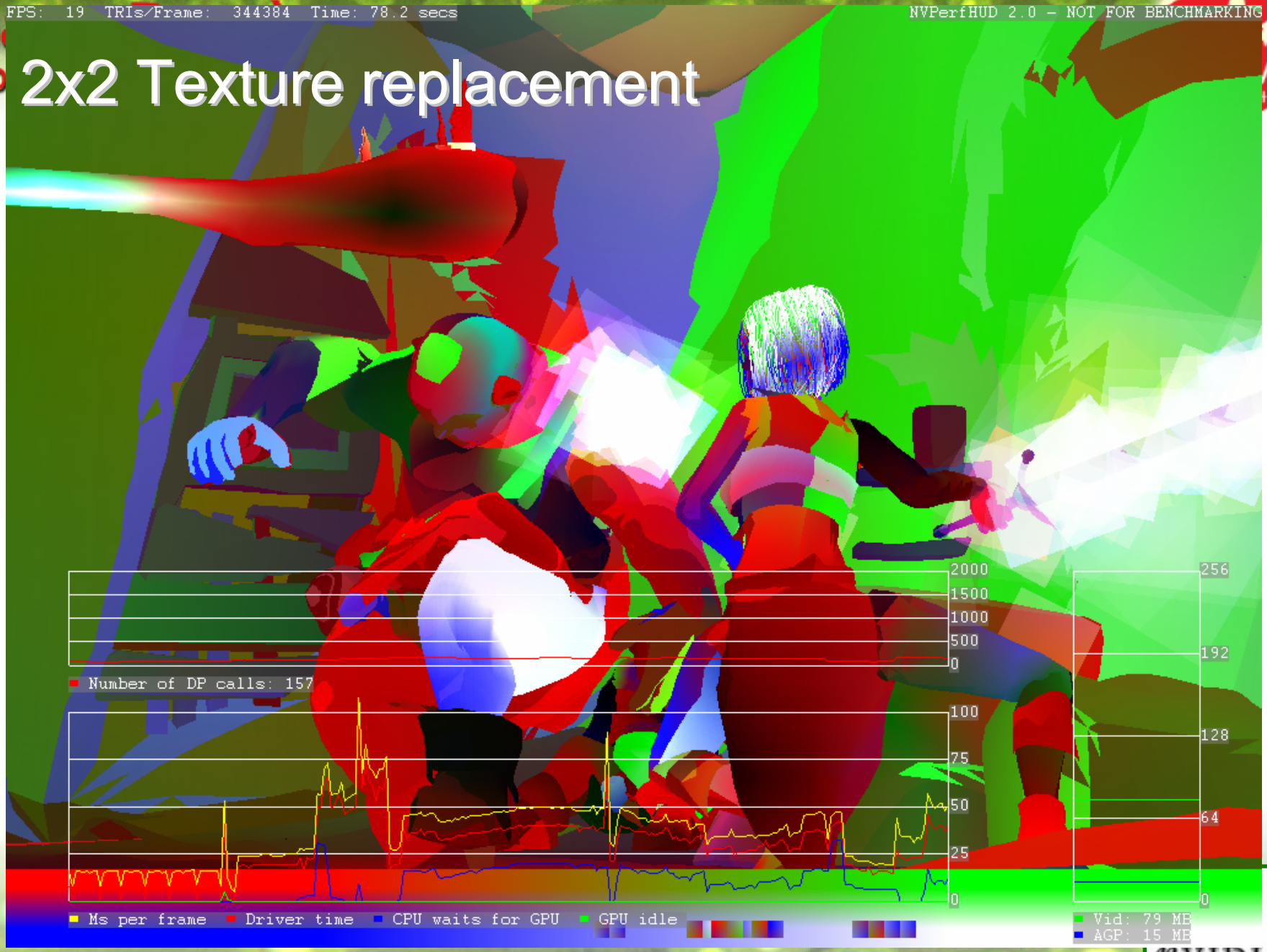


Image courtesy of FutureMark Corp.

Null DrawPrimitive mode

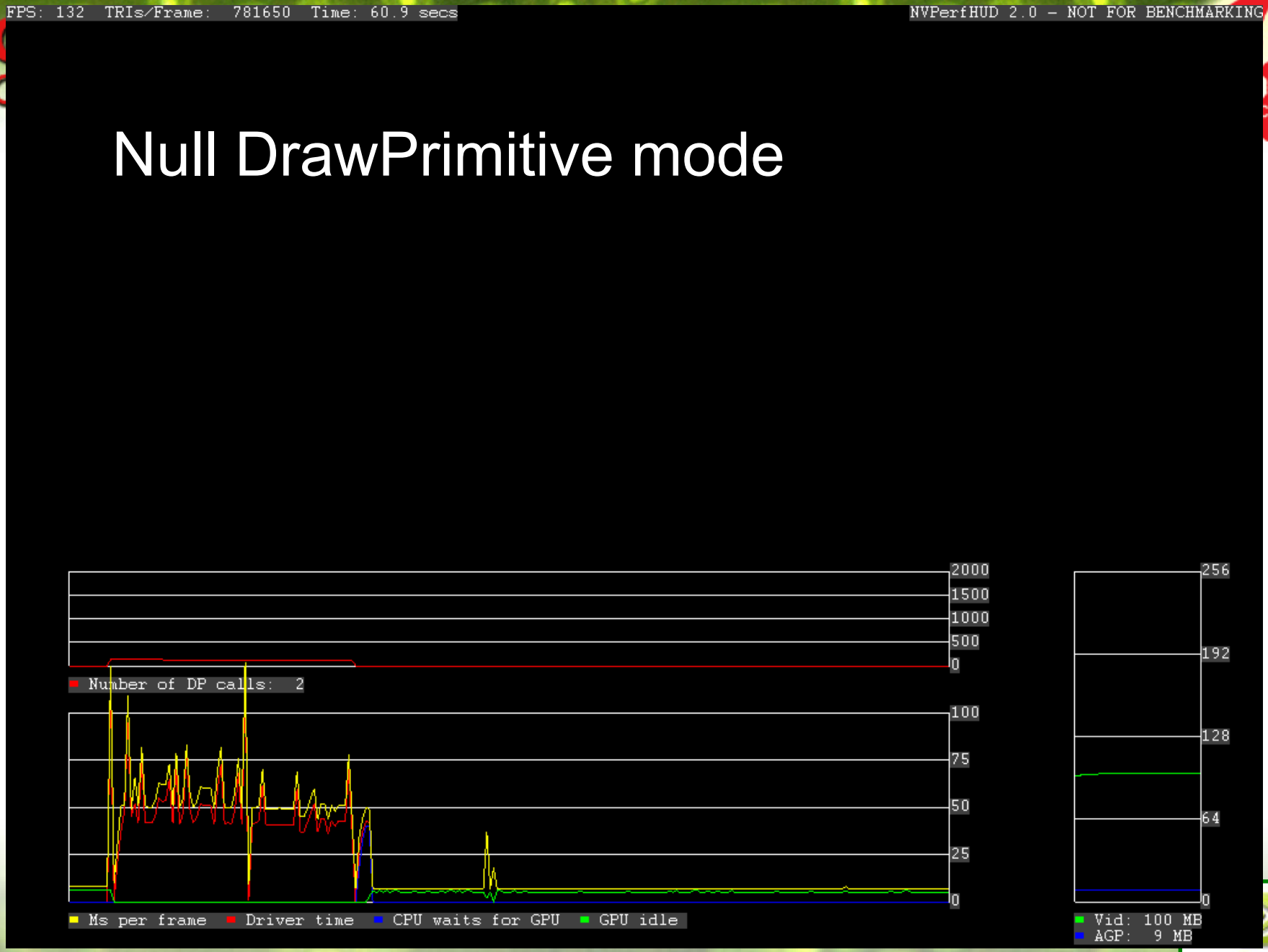


Image courtesy of FutureMark Corp.

NVPerfHUD - Overhead

- NVPerfHUD is fairly lean but...
- Overlay graph and interception only can costs up to 1.3%
- Driver instrumentation can cost up to 6%
- Upper bound for total cost: 7%

NVPerfHUD - Demo

Demo running FutureMark's 3DMark2003

NVShaderPerf

Inputs:

- HLSL
- !!FP1.0
- !!ARBfp1.0
- PS1.x
- PS2.x

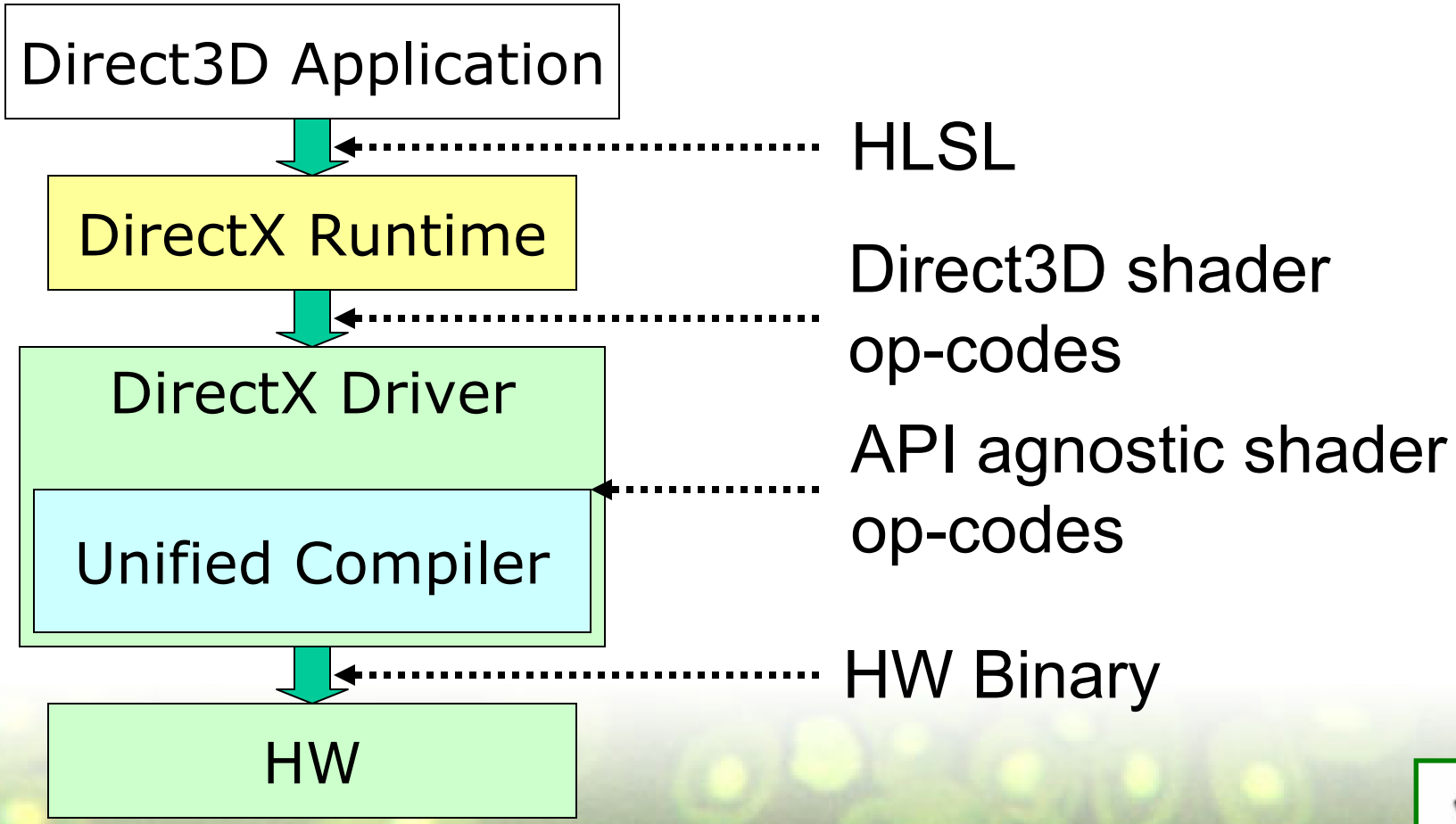
NVShaderPerf

GPU Arch:

- NV3X
- ...and more

```
C:\WINDOWS\System32\cmd.exe
C:\Temp\NVShaderPErf>NVShaderPerf.exe -a nv35 BumpReflectHLSL
------------------------------------------------------------------

Running performance on file BumpReflectHLSL.fx
******************************************************************
Technique <BumpReflect0>, Pass <p0>
-------------------- NV35 --------------------
PS Assembly Code:

 def c1, 2.000000, -1.000000, 0.000000, 0.000000
 dcl t0.rg
 dcl t1
 dcl t2
 dcl t3
 dcl_2d s0
 dcl_cube s1
 texld r0, t0, s0
 mad r0.rgb, c1.r, r0, c1.g
 mul r0.r, r0.r, c0.r
 mul r0.g, r0.g, c0.r
 nrm r1.rgb, r0
 dp3 r0.r, t1, r1
 dp3 r0.g, t2, r1
 dp3 r0.b, t3, r1
 mov r1.r, t1.a
 mov r1.g, t2.a
 mov r1.b, t3.a
 dp3 r2.r, r1, r0
 add r0.a, r2.r, r2.r
 mad r0.rgb, r0, -r0.a, r1
 texld r0, r0, s1
 mov oC0, r0
 end

Target: GeForceFX 5900 (NV35) :: Unified Compiler: v56.63
Cycles: 12 :: # R Registers: 2
GPU Utilization: 40.00%

C:\Temp\NVShaderPErf>
```

NVShaderPerf

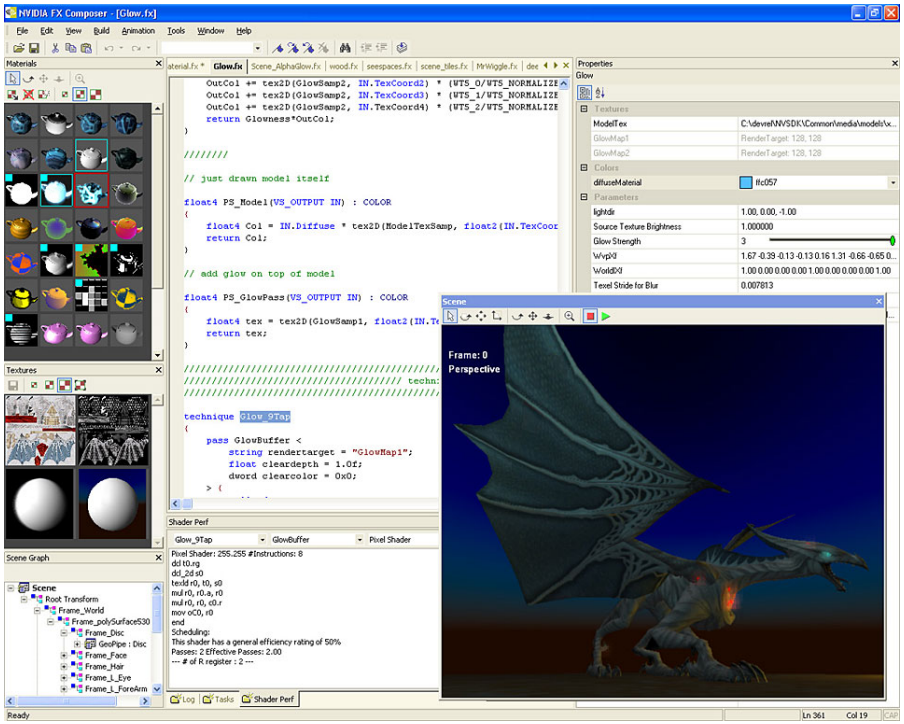


NVShaderPerf - Demo

```
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 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.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 last row.
      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.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 brightness)
      float3 eyevec = float3(IN.TexCoord1.w, IN.TexCoord2.w, IN.TexCoord3.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);
```

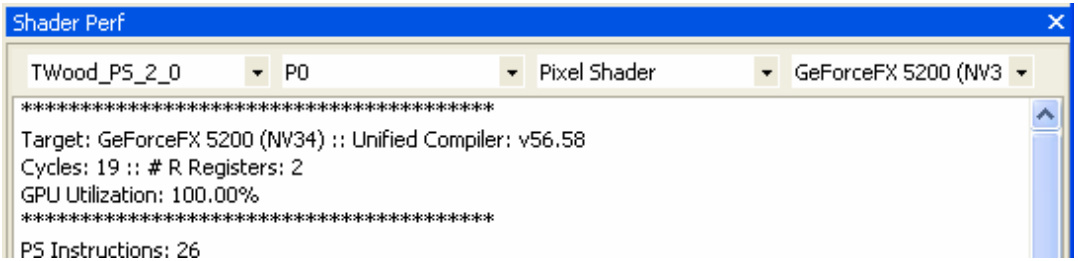
FX Composer 1.1

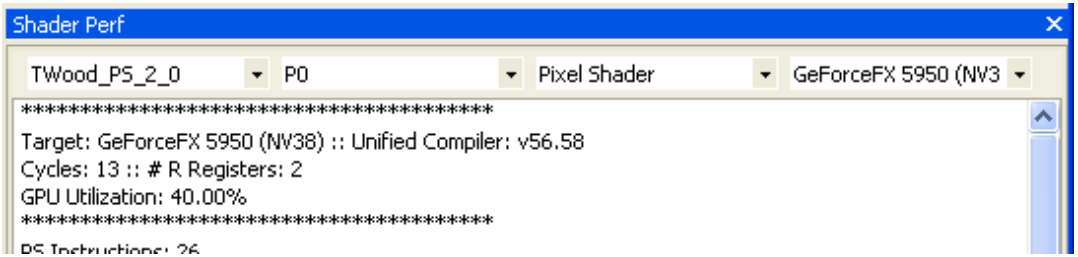
- IDE for HLSL authoring, debugging and optimization
- Pixel Shader scheduling
- Direct3D9 VS/PS op-code disassembly
- Advanced texture generation for baking Look Up Tables



FX Composer – Shader Perf

- Target GPU
- Driver version match
- Number of Cycles
- Number of Registers
- GPU Utilization





FX Composer - Disassembly

- Vertex Shader
- Pixel Shader 1.x, 2.x

Shader Perf

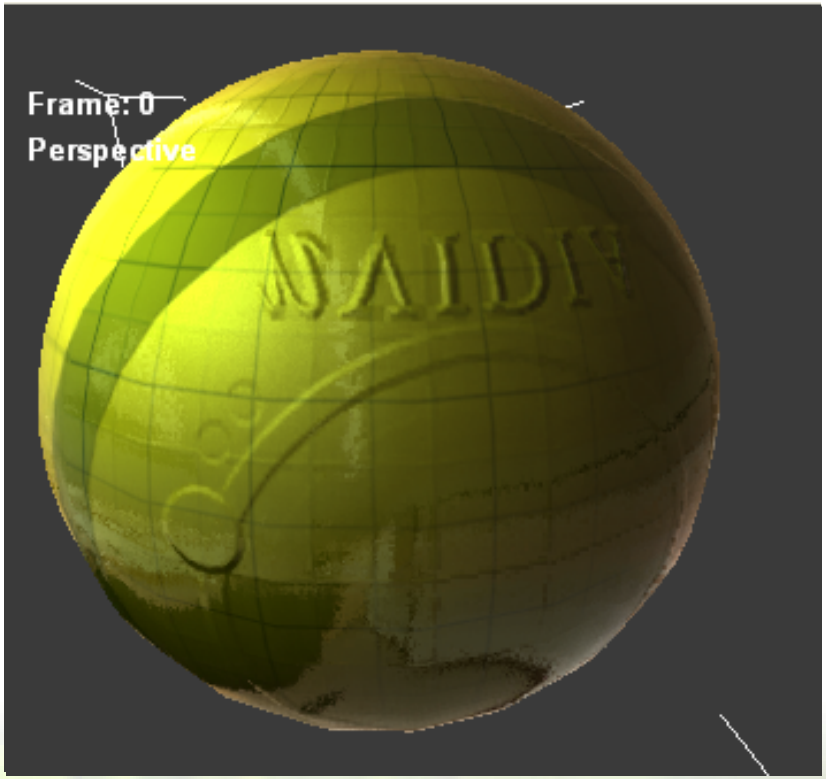
TWood_PS_2_0 | P0 | Pixel Shader | GeForceFX 5800 (NV3

```
PS Instructions: 26
ps_2_0
def c0, -0.500000, 0.200000, 0.000000, 10.000000
def c1, 0.700000, 0.300000, 0.000000, 0.000000
def c2, 0.100000, 0.600000, 0.800000, 1.000000
dcl v0.rgb
dcl v1.rgb
dcl t1.rgb
dcl_volume s0
frc r0.rgb, t1
texld r0, r0, s0
add r0.r, r0.r, c0.r
add r0.g, r0.b, c0.r
mad r0.r, r0.r, c0.g, t1.r
mad r0.g, r0.g, c0.g, t1.b
dp2add r0.a, r0, r0, c0.b
rsq r2.a, r0.a
mul r0.a, r2.a, r0.a
mul r1.rgb, r0.a, c0.a
texld r0, r1, s0
mad r0.a, r0.r, c2.r, r1.b
frc r0.a, r0.a
mad_sat r0.a, r0.a, -c2.g, c2.b
mad r0.a, r0.a, c1.r, c1.g
mov r0.rgb, v0
mad r0.rgb, r0, r0.a, v1
mov r0.a, c2.a
mov oC0, r0
```

Properties | Shader Perf

FX Composer – LUT Optimization

- Bake your own texture for function look up:
  - Normalization cubemaps
  - Lighting computation
  - Expensive math
  - Functions that can be artist controlled...



FX Composer - Demo

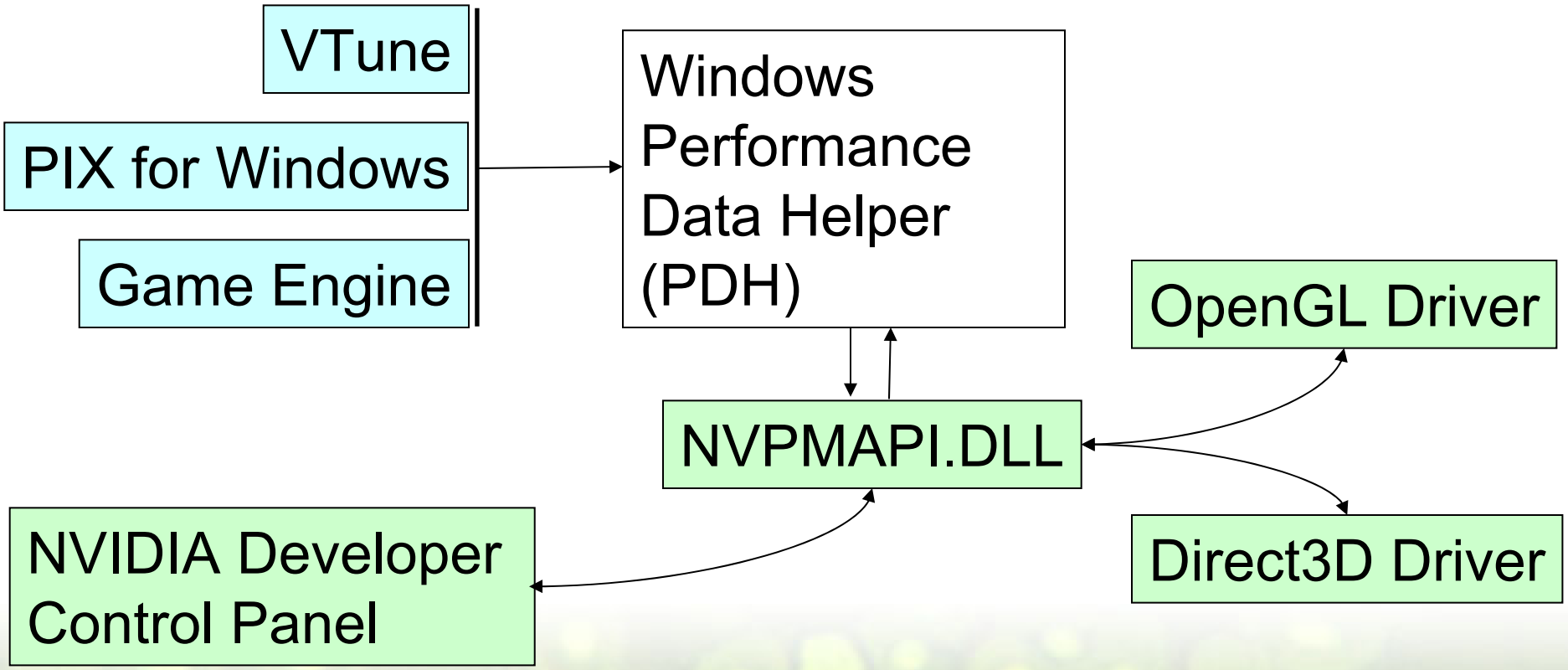
Overview of a performance tutorial

The Tools of Tomorrow

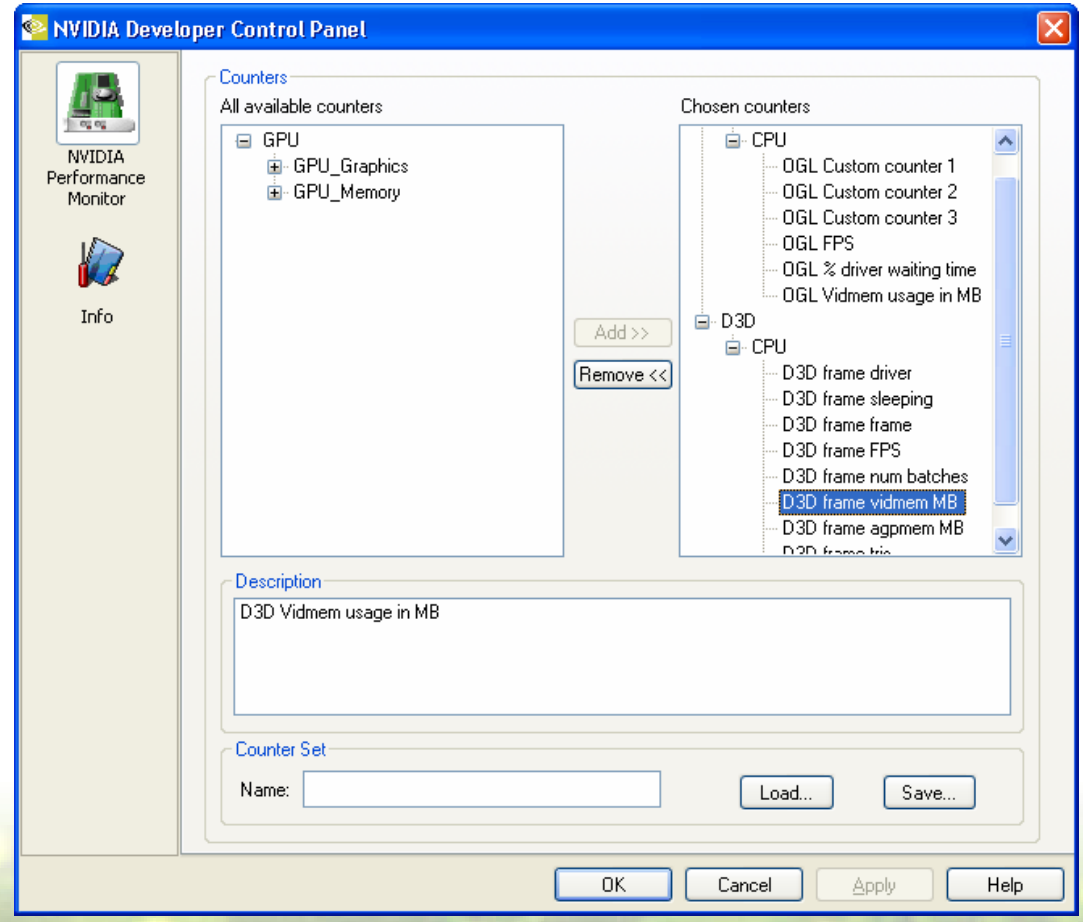
Instrumented Driver

- Perfect companion for Intel VTune, MSFT PIX for Windows, Perfmon, etc...
- Allows 3D applications to monitor:
  - Resource available (AGP, etc...)
  - Driver counters (Spins, etc...)
  - Hardware counters (bottlenecks, etc...)

Driver Instrumentation Architecture



NV Dev Control Panel



Instrumented Driver - Demo

- Direct3D9
- OpenGL
- HW Counters

FX Composer 1.5

- Vertex Shader Scheduling
- Texture Anisotropic Wizardry
- Support for next generation GPU
- …and much more to come!

Conclusion

- API Performance
- Unified Compiler Performance
- Driver Performance
- HW Performance
- What else do you need?

Other talks of interest...

- Wed: Practical Performance Analysis and Tuning [4:00pm – 5:00pm] by Ashu Rege and Clint Brewer

Credits

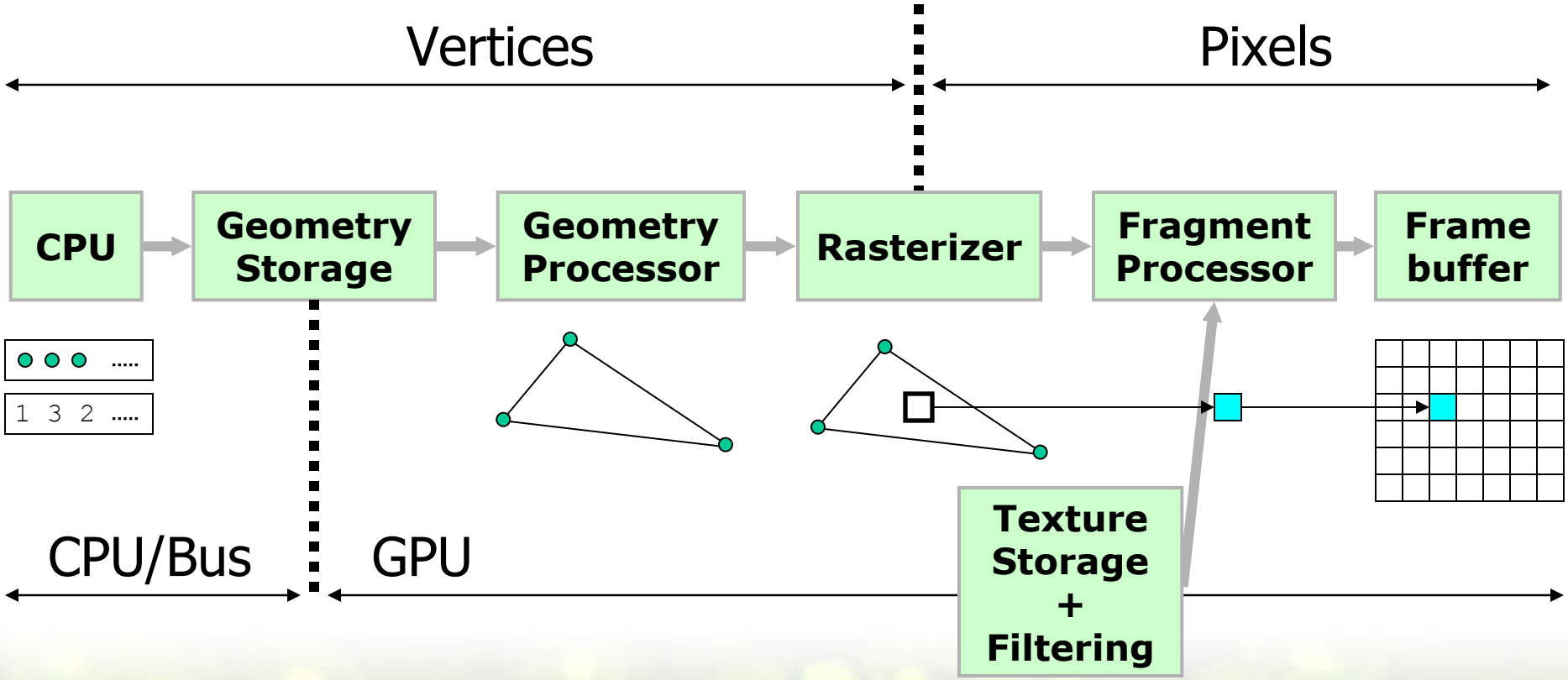
- FutureMark for letting us use 3DMark2003
- Special thanks to Raul Aguaviva, Jeffrey Kiel and Christopher Maughan for making these tools!

Q&A

- Questions?

Sébastien Dominé - sdomine@nvidia.com

Graphics Pipeline



PC Driver Model