STEAM DEV DAYS

GORDON SELLEY
TONY HOSIER
AMD GPU DEVELOPER TOOLS TEAM

OPTIMIZING LINUX GAMES FOR AMD GRAPHICS USING GPU PERFSTUDIO2

Background

- GPU PerfStudio2 is AMD's performance and debugging tool for graphics applications
- Initially developed to support DirectX and OpenGL on Windows® only
- Has recently been ported to Linux®
- Very useful when developing games for Steam Linux®
- Especially useful when optimizing games for AMD GPUs
- We are here to demonstrate GPUPerfStudio2 for Linux®

Presentation Overview

- **Introduction to GPU PerfStudio2**
  - — What it is, what it does, how it works, & who uses it
  - — Usage configurations
- **Using GPU PerfStudio2 for Linux**
  - — How to use it with an OpenGL Linux® app
  - — Demonstration of the main tool features
- **Data-mining your game using GPU PerfStudio2**
  - — Demonstration with a Steam Linux® game
- **What’s new?**
- **Summary**
- **Questions**

INTRODUCTION TO GPU PERFSTUDIO2

What it is, what it does,
how it works, & who uses it

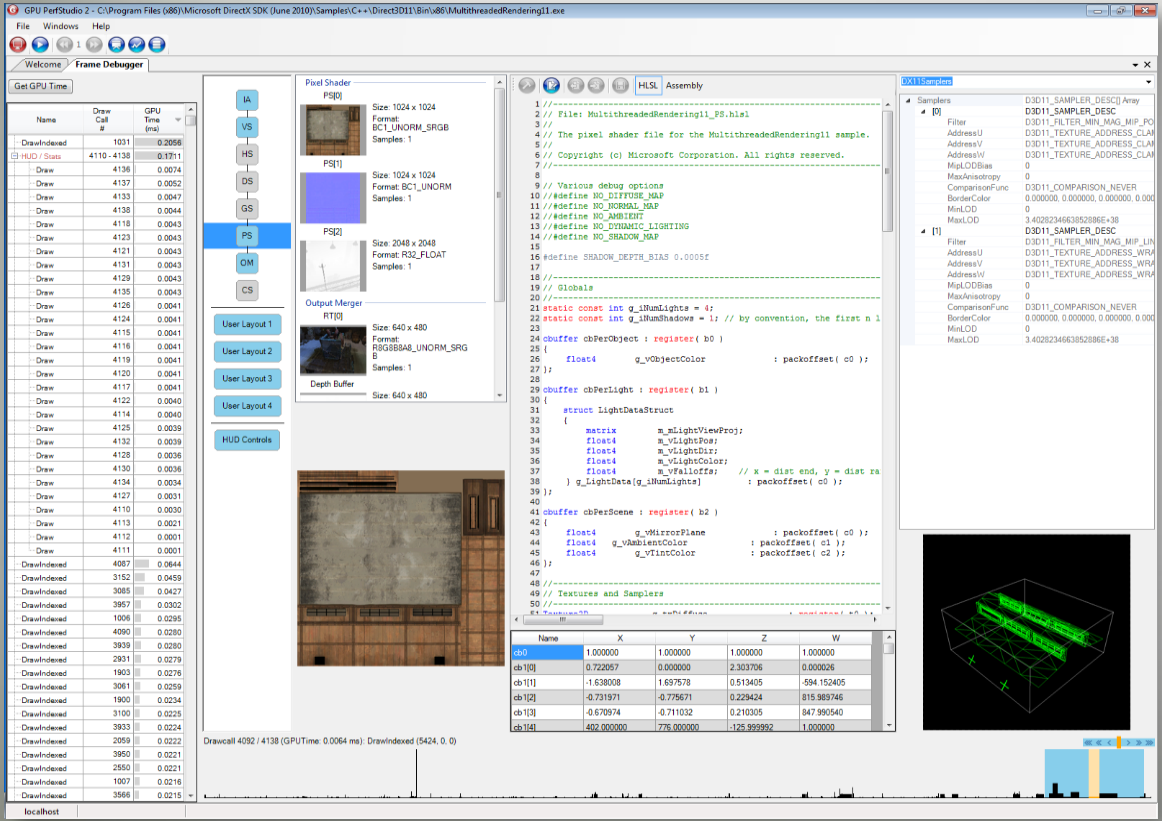
What is GPU PerfStudio2?

- GPU PerfStudio2 is AMD's performance and debugging tool for graphics applications
- A suite of tools that can be used to debug and increase performance on AMD GPUs
- Integrated **Frame Profiler**, **Frame Debugger**, and **API Trace** with CPU timing information
- Supports OpenGL 4.2 applications on Windows**®**
- Supports DirectX® 11, DirectX® 10.1 , DirectX® 10
- Now supports Linux®

What is GPU PerfStudio2?

- A lightweight, no installer, no change to your game, drag and drop, suite of GPU tools
- Run from a USB drive
- No need for Visual Studio integration
- Runs with game executables
- No special driver or compilation required

Frame Debugger



GPU PerfStudio 2 - C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\Samples\C++\Direct3D11\Bin\x86\MultithreadedRendering11.exe

Name	Draw Call #	GPU Time (ms)
DrawIndexed	1031	0.2056
HUD / Stats	4110 - 4138	0.1711
Draw	4136	0.0074
Draw	4137	0.0052
Draw	4133	0.0047
Draw	4138	0.0044
Draw	4118	0.0043
Draw	4123	0.0043
Draw	4121	0.0043
Draw	4131	0.0043
Draw	4129	0.0043
Draw	4135	0.0043
Draw	4126	0.0041
Draw	4124	0.0041
Draw	4115	0.0041
Draw	4116	0.0041
Draw	4119	0.0041
Draw	4120	0.0041
Draw	4117	0.0041
Draw	4122	0.0040
Draw	4114	0.0040
Draw	4125	0.0039
Draw	4132	0.0039
Draw	4128	0.0036
Draw	4130	0.0036
Draw	4134	0.0034
Draw	4127	0.0031
Draw	4110	0.0030
Draw	4113	0.0021
Draw	4112	0.0001
Draw	4111	0.0001
DrawIndexed	4087	0.0644
DrawIndexed	3152	0.0459
DrawIndexed	3085	0.0427
DrawIndexed	3957	0.0302
DrawIndexed	1006	0.0295
DrawIndexed	4090	0.0280
DrawIndexed	3939	0.0280
DrawIndexed	2931	0.0279
DrawIndexed	1903	0.0276
DrawIndexed	3061	0.0259
DrawIndexed	1900	0.0234
DrawIndexed	3100	0.0225
DrawIndexed	3933	0.0224
DrawIndexed	2059	0.0222
DrawIndexed	3950	0.0221
DrawIndexed	2550	0.0221
DrawIndexed	1007	0.0216
DrawIndexed	3566	0.0215

Pixel Shader

PS[0]
Size: 1024 x 1024
Format: BC1_UNORM_SRGB
Samples: 1

PS[1]
Size: 1024 x 1024
Format: BC1_UNORM
Samples: 1

PS[2]
Size: 2048 x 2048
Format: R32_FLOAT
Samples: 1

Output Merger

RT[0]
Size: 640 x 480
Format: R8G8B8A8_UNORM_SRGB
Samples: 1

Depth Buffer
Size: 640 x 480

```
//--------------------------------------------------------------------------------------
// File: MultithreadedRendering11_PS.hlsl
//
// The pixel shader file for the MultithreadedRendering11 sample.
//
// Copyright (c) Microsoft Corporation. All rights reserved.
//--------------------------------------------------------------------------------------

// Various debug options
//#define NO_DIFFUSE_MAP
//#define NO_NORMAL_MAP
//#define NO_AMBIENT
//#define NO_DYNAMIC_LIGHTING
//#define NO_SHADOW_MAP

#define SHADOW_DEPTH_BIAS 0.0005f

//--------------------------------------------------------------------------------------
// Globals
//--------------------------------------------------------------------------------------
static const int g_iNumLights = 4;
static const int g_iNumShadows = 1; // by convention, the first n l

cbuffer cbPerObject : register( b0 )
{
    float4       g_vObjectColor              : packoffset( c0 );
};

cbuffer cbPerLight : register( b1 )
{
    struct LightDataStruct
    {
        matrix          m_mLightViewProj;
        float4          m_vLightPos;
        float4          m_vLightDir;
        float4          m_vLightColor;
        float4          m_vFalloffs;    // x = dist end, y = dist ra
    } g_LightData[g_iNumLights]         : packoffset( c0 );
};

cbuffer cbPerScene : register( b2 )
{
    float4       g_vMirrorPlane              : packoffset( c0 );
    float4   g_vAmbientColor             : packoffset( c1 );
    float4       g_vTintColor            : packoffset( c2 );
};

//--------------------------------------------------------------------------------------
// Textures and Samplers
//--------------------------------------------------------------------------------------
Texture2D              g_txDiffuse                    : register( t0 );
```

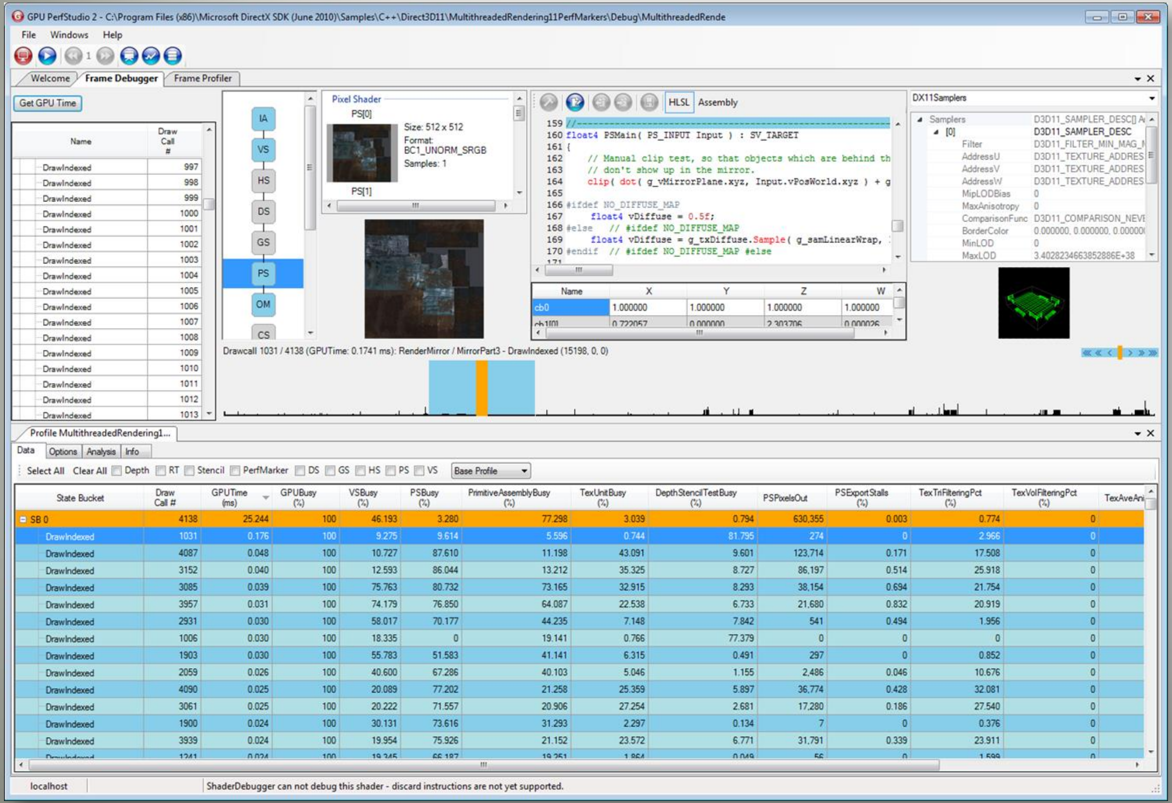
Name	X	Y	Z	W
cb0	1.000000	1.000000	1.000000	1.000000
cb1[0]	0.722057	0.000000	2.303706	0.000026
cb1[1]	-1.638008	1.697578	0.513405	-594.152405
cb1[2]	-0.731971	-0.775671	0.229424	815.989746
cb1[3]	-0.670974	-0.711032	0.210305	847.990540
cb1[4]	402.000000	776.000000	-125.999992	1.000000

Samplers	D3D11_SAMPLER_DESC[] Array
[0]	D3D11_SAMPLER_DESC
Filter	D3D11_FILTER_MIN_MAG_MIP_PO
AddressU	D3D11_TEXTURE_ADDRESS_CLAI
AddressV	D3D11_TEXTURE_ADDRESS_CLAI
AddressW	D3D11_TEXTURE_ADDRESS_CLAI
MipLODBias	0
MaxAnisotropy	0
ComparisonFunc	D3D11_COMPARISON_NEVER
BorderColor	0.000000, 0.000000, 0.000000, 0.000
MinLOD	0
MaxLOD	3.402823466385288 6E+38
[1]	D3D11_SAMPLER_DESC
Filter	D3D11_FILTER_MIN_MAG_MIP_LIN
AddressU	D3D11_TEXTURE_ADDRESS_WRA
AddressV	D3D11_TEXTURE_ADDRESS_WRA
AddressW	D3D11_TEXTURE_ADDRESS_WRA
MipLODBias	0
MaxAnisotropy	0
ComparisonFunc	D3D11_COMPARISON_NEVER
BorderColor	0.000000, 0.000000, 0.000000, 0.000
MinLOD	0
MaxLOD	3.4028234663852886E+38

Drawcall 4092 / 4138 (GPUTime: 0.0064 ms): DrawIndexed (5424, 0, 0)

- Capture, play back and view the contents of a frame
- Scrub through draw calls
- Visualize the GPU time for each draw call
- View all game resources and state bound at each draw call
- Inspect the resources at each stage of the pipeline
- View, edit and debug shader code

Frame Profiler


GPU PerfStudio 2 - C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\Samples\C++\Direct3D11\MultithreadedRendering11PerfMarkers\Debug\MultithreadedRende
File
Windows
Help
Welcome
Frame Debugger
Frame Profiler
Get GPU Time
IA
VS
HS
DS
GS
PS
OM
CS
Pixel Shader
PS[0]
PS[1]
HLSL
Assembly
DX11Samplers

Size: 512 x 512
Format: BC1_UNORM_SRGB
Samples: 1

Name	Draw Call #
DrawIndexed	997
DrawIndexed	998
DrawIndexed	999
DrawIndexed	1000
DrawIndexed	1001
DrawIndexed	1002
DrawIndexed	1003
DrawIndexed	1004
DrawIndexed	1005
DrawIndexed	1006
DrawIndexed	1007
DrawIndexed	1008
DrawIndexed	1009
DrawIndexed	1010
DrawIndexed	1011
DrawIndexed	1012
DrawIndexed	1013

```
//-----------------------------------------------------------------
float4 PSMain( PS_INPUT Input ) : SV_TARGET
{
    // Manual clip test, so that objects which are behind th
    // don't show up in the mirror.
    clip( dot( g_vMirrorPlane.xyz, Input.vPosWorld.xyz ) + g

#ifdef NO_DIFFUSE_MAP
     float4 vDiffuse = 0.5f;
#else   // #ifdef NO_DIFFUSE_MAP
     float4 vDiffuse = g_txDiffuse.Sample( g_samLinearWrap, 
#endif  // #ifdef NO_DIFFUSE_MAP #else
```

Samplers	D3D11_SAMPLER_DESC[] A
[0]	D3D11_SAMPLER_DESC
Filter	D3D11_FILTER_MIN_MAG_I
AddressU	D3D11_TEXTURE_ADDRES
AddressV	D3D11_TEXTURE_ADDRES
AddressW	D3D11_TEXTURE_ADDRES
MipLODBias	0
MaxAnisotropy	0
ComparisonFunc	D3D11_COMPARISON_NEVE
BorderColor	0.000000, 0.000000, 0.00000
MinLOD	0
MaxLOD	3.402823466385286E+38

Name	X	Y	Z	W
cb0	1.000000	1.000000	1.000000	1.000000
cb1[0]	0.722057	0.000000	2.303706	0.000026

Drawcall 1031 / 4138 (GPUTime: 0.1741 ms): RenderMirror / MirrorPart3 - DrawIndexed (15198, 0, 0)

Profile MultithreadedRendering1...

Data | Options | Analysis | Info

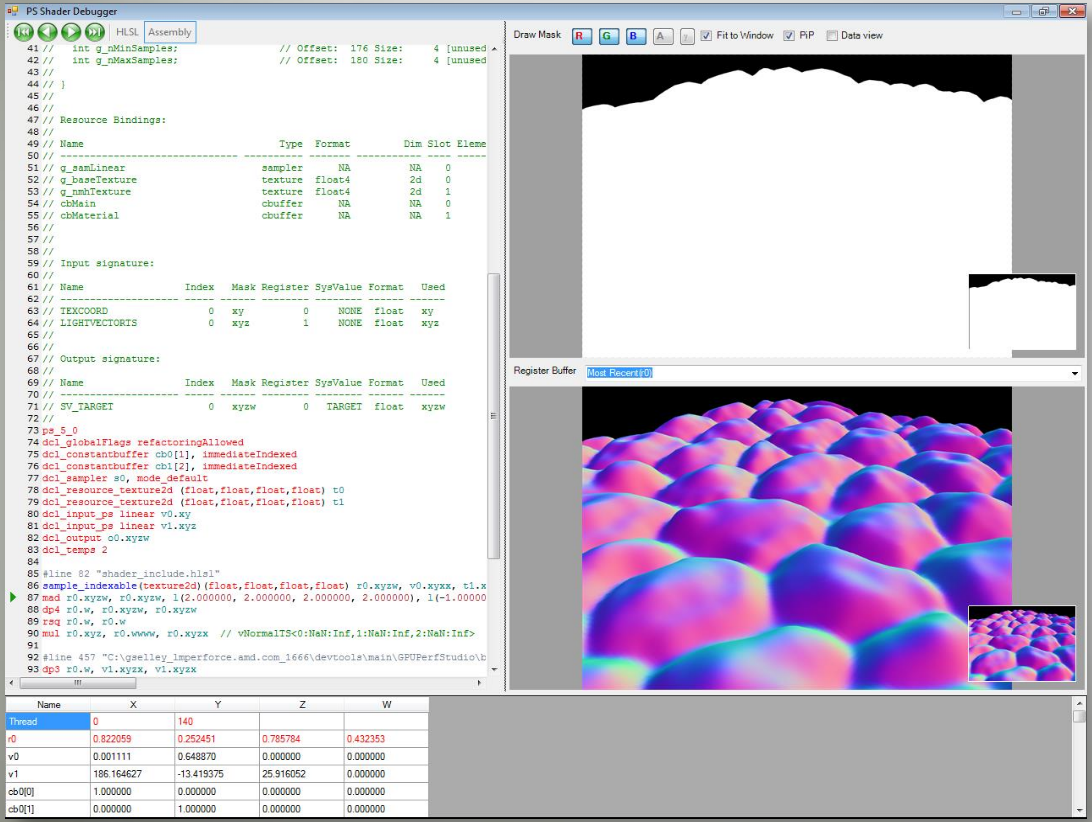
Select All Clear All □ Depth □ RT □ Stencil □ PerfMarker □ DS □ GS □ HS □ PS □ VS Base Profile

State Bucket	Draw Call #	GPUTime (ms)	GPUBusy (%)	VSBusy (%)	PSBusy (%)	PrimitiveAssemblyBusy (%)	TexUnitBusy (%)	DepthStencilTestBusy (%)	PSPixelsOut	PSExportStalls (%)	TexTriFilteringPct (%)	TexVolFilteringPct (%)	TexAveAni
SB 0	4138	25.244	100	46.193	3.280	77.298	3.039	0.794	630,355	0.003	0.774	0	
DrawIndexed	1031	0.176	100	9.275	9.614	5.596	0.744	81.795	274	0	2.966	0	
DrawIndexed	4087	0.048	100	10.727	87.610	11.198	43.091	9.601	123,714	0.171	17.508	0	
DrawIndexed	3152	0.040	100	12.593	86.044	13.212	35.325	8.727	86,197	0.514	25.918	0	
DrawIndexed	3085	0.039	100	75.763	80.732	73.165	32.915	8.293	38,154	0.694	21.754	0	
DrawIndexed	3957	0.031	100	74.179	76.850	64.087	22.538	6.733	21,680	0.832	20.919	0	
DrawIndexed	2931	0.030	100	58.017	70.177	44.235	7.148	7.842	541	0.494	1.956	0	
DrawIndexed	1006	0.030	100	18.335	0	19.141	0.766	77.379	0	0	0	0	
DrawIndexed	1903	0.030	100	55.783	51.583	41.141	6.315	0.491	297	0	0.852	0	
DrawIndexed	2059	0.026	100	40.600	67.286	40.103	5.046	1.155	2,486	0.046	10.676	0	
DrawIndexed	4090	0.025	100	20.089	77.202	21.258	25.359	5.897	36,774	0.428	32.081	0	
DrawIndexed	3061	0.025	100	20.222	71.557	20.906	27.254	2.681	17,280	0.186	27.540	0	
DrawIndexed	1900	0.024	100	30.131	73.616	31.293	2.297	0.134	7	0	0.376	0	
DrawIndexed	3939	0.024	100	19.954	75.926	21.152	23.572	6.771	31,791	0.339	23.911	0	
DrawIndexed	1241	0.024	100	19.345	66.187	19.251	1.864	0.049	56	0	1.599	0	

localhost | ShaderDebugger can not debug this shader - discard instructions are not yet supported.

- Identify costly draw calls
- Detect GPU pipeline stage bottlenecks
- Investigate performance issues at the counter level

Shader Debugger



PS Shader Debugger

☑ Fit to Window ☑ PiP □ Data view

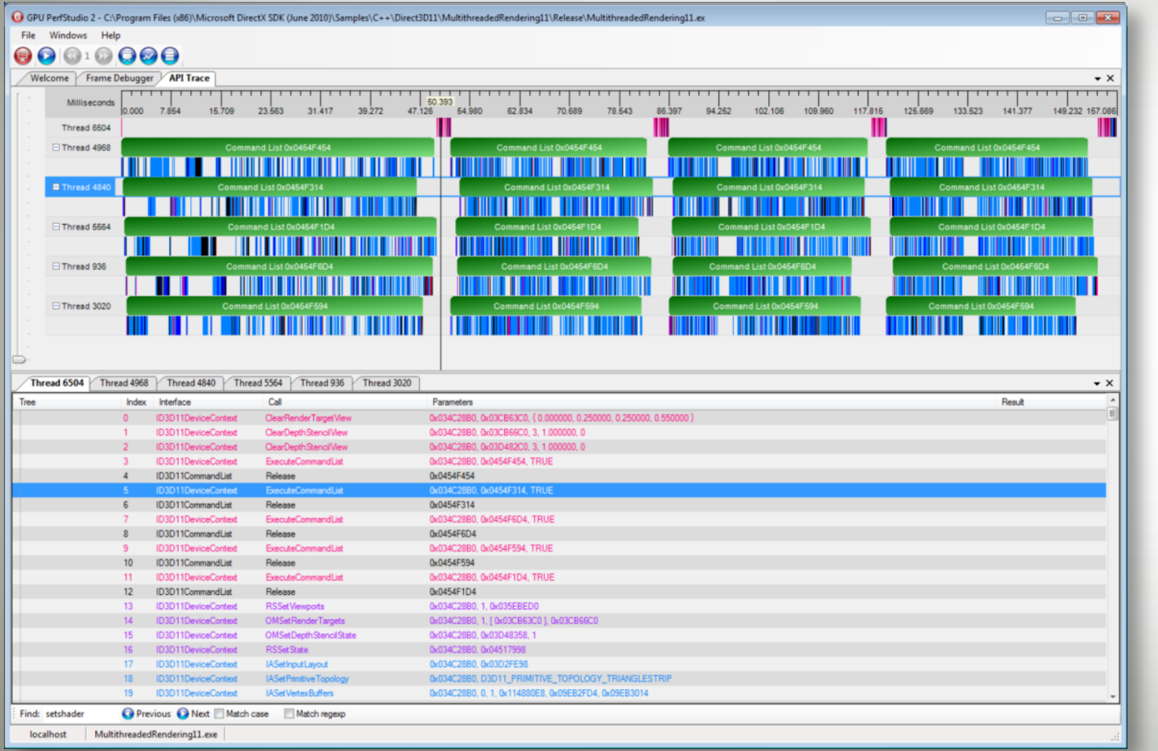
Register Buffer: Most Recent(r0)

```
//   int g_nMinSamples;                 // Offset:  176 Size:     4 [unused
//   int g_nMaxSamples;                 // Offset:  180 Size:     4 [unused
//
// }
//
//
// Resource Bindings:
//
// Name                                 Type  Format         Dim Slot Eleme
// ------------------------------ ---------- ------- ----------- ---- -----
// g_samLinear                       sampler      NA          NA    0
// g_baseTexture                     texture  float4          2d    0
// g_nmhTexture                      texture  float4          2d    1
// cbMain                            cbuffer      NA          NA    0
// cbMaterial                        cbuffer      NA          NA    1
//
//
//
// Input signature:
//
// Name                 Index   Mask Register SysValue Format   Used
// -------------------- ----- ------ -------- -------- ------ ------
// TEXCOORD                 0   xy          0     NONE  float   xy
// LIGHTVECTORTS            0   xyz         1     NONE  float   xyz
//
//
// Output signature:
//
// Name                 Index   Mask Register SysValue Format   Used
// -------------------- ----- ------ -------- -------- ------ ------
// SV_TARGET                0   xyzw        0   TARGET  float   xyzw
//
ps_5_0
dcl_globalFlags refactoringAllowed
dcl_constantbuffer cb0[1], immediateIndexed
dcl_constantbuffer cb1[2], immediateIndexed
dcl_sampler s0, mode_default
dcl_resource_texture2d (float,float,float,float) t0
dcl_resource_texture2d (float,float,float,float) t1
dcl_input_ps linear v0.xy
dcl_input_ps linear v1.xyz
dcl_output o0.xyzw
dcl_temps 2

#line 82 "shader_include.hlsl"
sample_indexable(texture2d)(float,float,float,float) r0.xyzw, v0.xyxx, t1.x
mad r0.xyzw, r0.xyzw, l(2.000000, 2.000000, 2.000000, 2.000000), l(-1.00000
dp4 r0.w, r0.xyzw, r0.xyzw
rsq r0.w, r0.w
mul r0.xyz, r0.wwww, r0.xyzx  // vNormalTS<0:NaN:Inf,1:NaN:Inf,2:NaN:Inf>

#line 457 "C:\gselley_lmperforce.amd.com_1666\devtools\main\GPUPerfStudio\b
dp3 r0.w, v1.xyzx, v1.xyzx
```

Name	X	Y	Z	W
Thread	0	140		
r0	0.822059	0.252451	0.785784	0.432353
v0	0.001111	0.648870	0.000000	0.000000
v1	186.164627	-13.419375	25.916052	0.000000
cb0[0]	1.000000	0.000000	0.000000	0.000000
cb0[1]	0.000000	1.000000	0.000000	0.000000

- Edit the live HLSL or GLSL code inside your app while running in the tool
- Debug the live HLSL or Assembly code inside your app while running in the tool
  - Step through shader code
  - Inspect all register values
  - Insert and run-to break points
- Compare before and after edit performance using the Profiler

API Trace



Tree	Index	Interface	Call	Parameters	Result
	0	ID3D11DeviceContext	ClearRenderTargetView	0x034C28B0, 0x03CB63C0, { 0.000000, 0.250000, 0.250000, 0.550000 }	
	1	ID3D11DeviceContext	ClearDepthStencilView	0x034C28B0, 0x03CB66C0, 3, 1.000000, 0	
	2	ID3D11DeviceContext	ClearDepthStencilView	0x034C28B0, 0x03D482C0, 3, 1.000000, 0	
	3	ID3D11DeviceContext	ExecuteCommandList	0x034C28B0, 0x0454F454, TRUE	
	4	ID3D11CommandList	Release	0x0454F454	
	5	ID3D11DeviceContext	ExecuteCommandList	0x034C28B0, 0x0454F314, TRUE	
	6	ID3D11CommandList	Release	0x0454F314	
	7	ID3D11DeviceContext	ExecuteCommandList	0x034C28B0, 0x0454F6D4, TRUE	
	8	ID3D11CommandList	Release	0x0454F6D4	
	9	ID3D11DeviceContext	ExecuteCommandList	0x034C28B0, 0x0454F594, TRUE	
	10	ID3D11CommandList	Release	0x0454F594	
	11	ID3D11DeviceContext	ExecuteCommandList	0x034C28B0, 0x0454F1D4, TRUE	
	12	ID3D11CommandList	Release	0x0454F1D4	
	13	ID3D11DeviceContext	RSSetViewports	0x034C28B0, 1, 0x035EBED0	
	14	ID3D11DeviceContext	OMSetRenderTargets	0x034C28B0, 1, [0x03CB63C0], 0x03CB66C0	
	15	ID3D11DeviceContext	OMSetDepthStencilState	0x034C28B0, 0x03D48358, 1	
	16	ID3D11DeviceContext	RSSetState	0x034C28B0, 0x04517998	
	17	ID3D11DeviceContext	IASetInputLayout	0x034C28B0, 0x03D2FE98	
	18	ID3D11DeviceContext	IASetPrimitiveTopology	0x034C28B0, D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	
	19	ID3D11DeviceContext	IASetVertexBuffers	0x034C28B0, 0, 1, 0x114880E8, 0x09EB2FD4, 0x09EB3014	

- Inspect all API calls (with arguments)
- CPU timeline information for each API call
- Visualize multi-threaded API usage
- Supports DirectX®11 Command Lists and deferred contexts

Who uses GPU PerfStudio2?

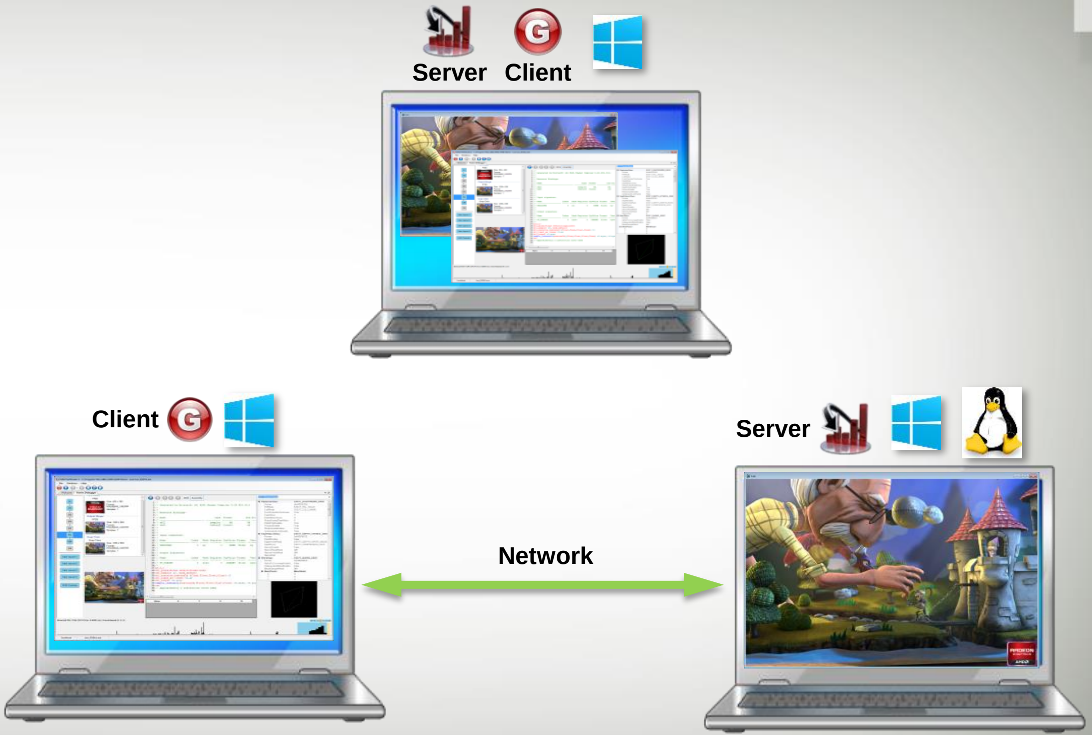
- **Widely used by internal groups in AMD**
  - **AMD Developer Technology Engineers**: Optimize & debug game titles in conjunction with developers
  - **AMD Driver Performance Team**: Improve GPU benchmarks and titles at the driver level
  - **AMD Driver Team**: Inspect apps that cause driver problems
  - **AMD Game Compute Team**: Debug and optimize game technologies for new GPU hardware - AMD Mecha Demo, Ladybug, Leo demo

- **External users**
  - **Graphics developers**: Used in the development of DirectX®11 and OpenGL graphics applications

Remote and local debug sessions

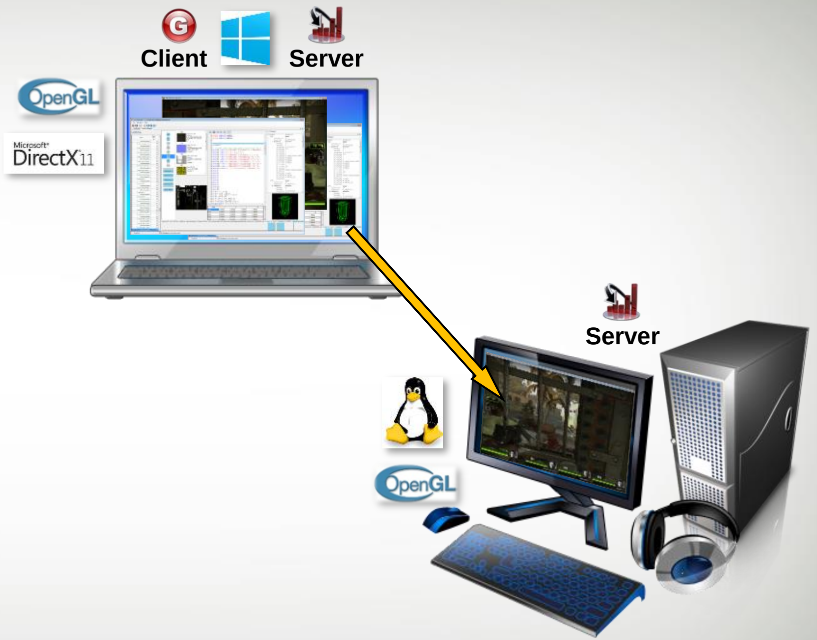
- **Local usage** client and server run on a single machine (Windows® only – DirectX® or OpenGL)

- **Remote usage** client and server run on separate computers. Allows the game to be run full screen. Higher profiling accuracy, useful during final optimization (Server - Windows® or Linux®)



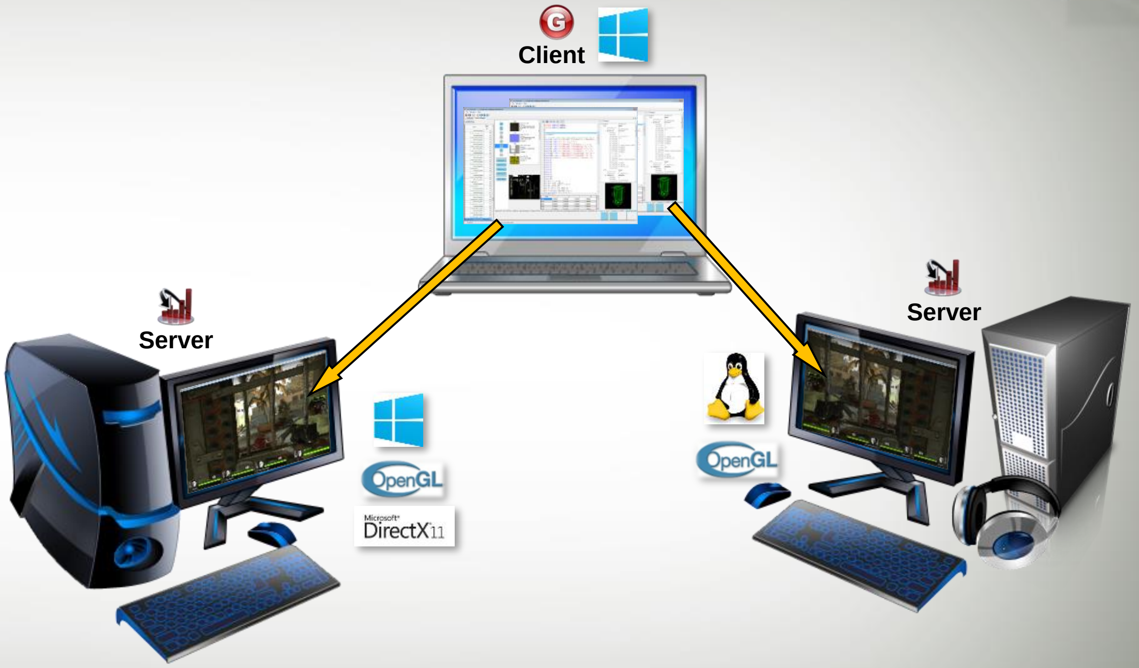
Local and remote debug sessions

- Two clients can connect to a local and remote server simultaneously
- First client – connect to remote game running on Linux®
- Second client – connect to local game running on Windows®
- Compare DirectX® 11 on Windows® to OpenGL Linux®
- Compare OpenGL on Windows® to OpenGL on Linux®
- This is the scenario we will be demonstrating today



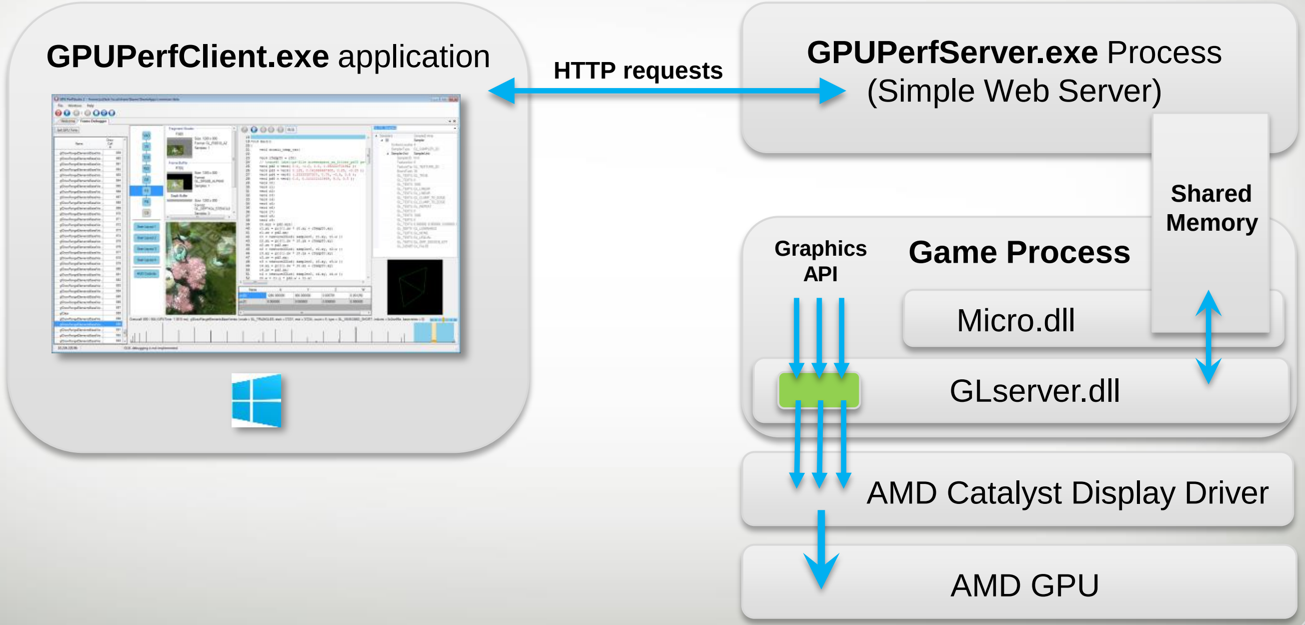
Dual remote debug sessions

- Two clients can connect to remote servers simultaneously
- First client connects to remote game running on Linux®
- Second client connects to remote game running on Windows®
- Compare DirectX®11 on Windows® to OpenGL on Linux®
- Compare OpenGL on Windows® to OpenGL on Linux®



How GPS2 works

Game Host Computer



GPUPerfClient.exe application

HTTP requests

GPUPerfServer.exe Process
(Simple Web Server)

Shared Memory

Graphics API

Game Process

Micro.dll

GLserver.dll

AMD Catalyst Display Driver

AMD GPU

USING GPU PERFSTUDIO2 FOR LINUX

GPU PerfStudio2’s OpenGL Background

- GPU PerfStudio2 (GPS2) supported OpenGL early in its development
- OpenGL support grew during the development of Brink and Rage
  - Used in house at AMD to debug driver issues and for GPU profiling
- GPS2 was used by Valve in the porting of Source Engine to OpenGL
  - First tool that would work with a “large” OpenGL application
- AMD’s gDEBugger was also used by Valve
  - Helps in debugging context creation code by checking for common OpenGL context creation errors
  - gDEBugger features now supported by AMD’s CodeXL

GPU PerfStudio2's OpenGL Background

- GPS2 was used by Valve in the porting of Source Engine to Linux®
- How? GPS2 only ran on Windows® at the time?
  - — Valve found that most AMD driver issues on Linux® also existed in the Windows® driver so could be debugged/reported on Windows®
  - — The tools ecosystem on Windows® was already well developed so most of the work could be done on Windows®
  - — No real need to move existing tools to Linux® (at that time)

GPU PerfStudio2’s Linux Background

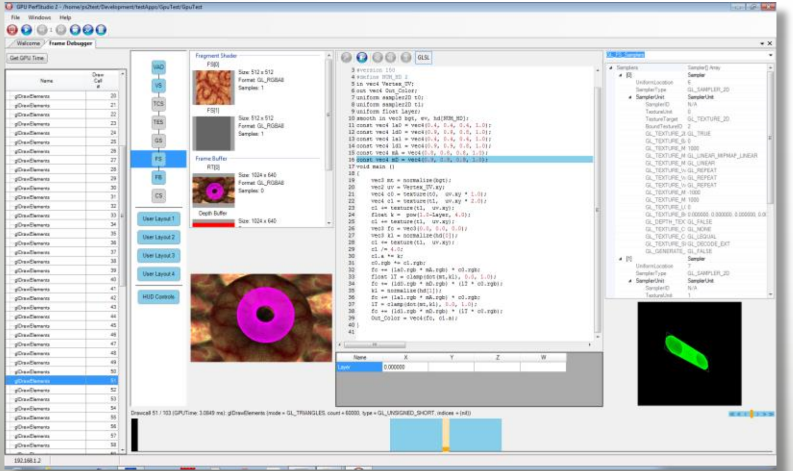
- Drawback of GPS2 only running on Windows®
  - — Not possible to profile the GPU directly on Linux® using GPS2
- What about GPUPerfAPI?
  - — AMD’s library for accessing GPU performance counters on AMD GPUs
  - — Available for Linux® and Windows®
  - — Developers can integrate GPU profiling into their own tools using GPUPerfAPI
  - — AMD’s GPS2 and CodeXL use GPUPerfAPI under the hood
- With the release of Steam for Linux® GPU tools are even more important to the game developer community at large
- AMD started porting GPU PerfStudio2 to Linux® in mid-2013
  - — Targeted Steam for Linux® games
  - — Standalone OpenGL applications

GPU PerfStudio2 for Linux

- Works with most current Steam for Linux games
  - Left4Dead2
  - Portal
  - DOTA 2
  - Half-Life^2
  - Counter Strike
  - TeamFortress2
- Targeting Ubuntu12.04
- Currently in beta testing
- Availability end of Q1 2014

Stop talking! Show me

- GPU PerfStudio2 running with the GpuTest Furmark OpenGL benchmarking application
- Download: http://www.geeks3d.com/gputest/



Name	Draw Call #
glDrawElements	20
glDrawElements	21
glDrawElements	22
glDrawElements	23
glDrawElements	24
glDrawElements	25
glDrawElements	26
glDrawElements	27
glDrawElements	28
glDrawElements	29
glDrawElements	30
glDrawElements	31
glDrawElements	32
glDrawElements	33
glDrawElements	34
glDrawElements	35
glDrawElements	36
glDrawElements	37
glDrawElements	38
glDrawElements	39
glDrawElements	40
glDrawElements	41
glDrawElements	42
glDrawElements	43
glDrawElements	44
glDrawElements	45
glDrawElements	46
glDrawElements	47
glDrawElements	48
glDrawElements	49
glDrawElements	50
glDrawElements	51
glDrawElements	52
glDrawElements	53
glDrawElements	54
glDrawElements	55
glDrawElements	56
glDrawElements	57
glDrawElements	58

Size: 512 x 512
Format: GL_RGBA8
Samples: 1

Size: 512 x 512
Format: GL_RGBA8
Samples: 1

Size: 1024 x 640
Format: GL_RGBA8
Samples: 0

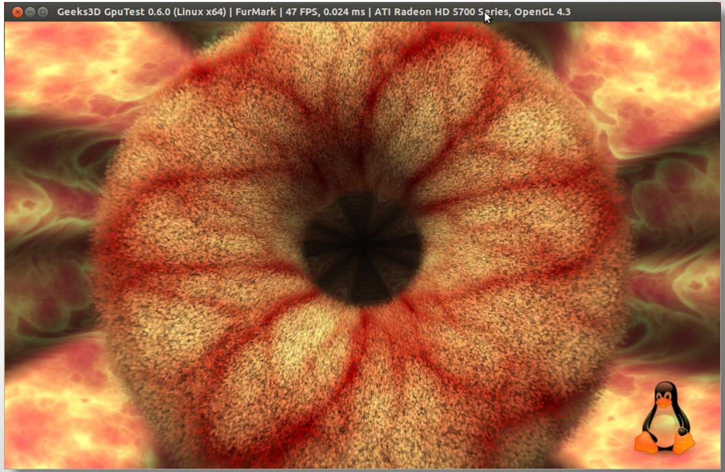
Size: 1024 x 640

```
#version 150
#define NUM_HD 2
in vec4 Vertex_UV;
out vec4 Out_Color;
uniform sampler2D t0;
uniform sampler2D t1;
uniform float Layer;
smooth in vec3 bgt, ev, hd[NUM_HD];
const vec4 la0 = vec4(0.4, 0.4, 0.4, 1.0);
const vec4 ld0 = vec4(0.9, 0.9, 0.8, 1.0);
const vec4 la1 = vec4(0.4, 0.4, 0.4, 1.0);
const vec4 ld1 = vec4(0.9, 0.9, 0.8, 1.0);
const vec4 mA = vec4(0.8, 0.8, 0.8, 1.0);
const vec4 mD = vec4(0.9, 0.9, 0.8, 1.0);
void main ()
{
    vec3 mt = normalize(bgt);
    vec2 uv = Vertex_UV.xy;
    vec4 c0 = texture(t0,  uv.xy * 1.0);
    vec4 c1 = texture(t1,  uv.xy * 2.0);
    c1 += texture(t1,  uv.xy);
    float k =  pow(1.0-Layer, 4.0);
    c1 += texture(t1,  uv.xy);
    vec3 fc = vec3(0.0, 0.0, 0.0);
    vec3 kl = normalize(hd[0]);
    c1 += texture(t1,  uv.xy);
    c1 /= 4.0;
    c1.a *= k;
    c0.rgb *= c1.rgb;
    fc += (la0.rgb * mA.rgb) * c0.rgb;
    float lT = clamp(dot(mt,kl), 0.0, 1.0);
    fc += (ld0.rgb * mD.rgb) * (lT * c0.rgb);
    kl = normalize(hd[1]);
    fc += (la1.rgb * mA.rgb) * c0.rgb;
    lT = clamp(dot(mt,kl), 0.0, 1.0);
    fc += (ld1.rgb * mD.rgb) * (lT * c0.rgb);
    Out_Color = vec4(fc, c1.a);
}
```

Name	X	Y	Z	W
Layer	0.000000			

Samplers	Sampler[] Array
[0]	Sampler
UniformLocation	6
SamplerType	GL_SAMPLER_2D
SamplerUnit	SamplerUnit
SamplerID	N/A
TextureUnit	0
TextureTarget	GL_TEXTURE_2D
BoundTextureID	2
GL_TEXTURE_2I	GL_TRUE
GL_TEXTURE_B	0
GL_TEXTURE_M	1000
GL_TEXTURE_M	GL_LINEAR_MIPMAP_LINEAR
GL_TEXTURE_M	GL_LINEAR
GL_TEXTURE_W	GL_REPEAT
GL_TEXTURE_W	GL_REPEAT
GL_TEXTURE_W	GL_REPEAT
GL_TEXTURE_M	-1000
GL_TEXTURE_M	1000
GL_TEXTURE_L	0
GL_TEXTURE_B	0.000000, 0.000000, 0.000000, 0.0
GL_DEPTH_TEX	GL_FALSE
GL_TEXTURE_C	GL_NONE
GL_TEXTURE_C	GL_LEQUAL
GL_TEXTURE_SI	GL_DECODE_EXT
GL_GENERATE_	GL_FALSE
[1]	Sampler
UniformLocation	7
SamplerType	GL_SAMPLER_2D
SamplerUnit	SamplerUnit
SamplerID	N/A
TextureUnit	1

Drawcall 51 / 103 (GPUTime: 3.0849 ms): glDrawElements (mode = GL_TRIANGLES, count = 60000, type = GL_UNSIGNED_SHORT, indices = (nil))


Geeks3D GpuTest 0.6.0 (Linux x64) | FurMark | 47 FPS, 0.024 ms | ATI Radeon HD 5700 Series, OpenGL 4.3

GPU PerfStudio2 Linux setup

- Extract the GPUPerfStudio2 tarball in:

```
~/Development/GPUPerfStudio
```

- Install GpuTest Furmark in:

```
~/Development/testApps/GpuTest
```

- Create a shell script in the following dir:

```
~Development/scripts/furmark.s
```

- Contents of the above furmark.sh scri

```
cd ~/Development/testApps/GpuTest
~/Development/GPUPerfstudio/x64/GPUPerfServer -S start_furmark_windowed_1024x640.sh
```

Full p

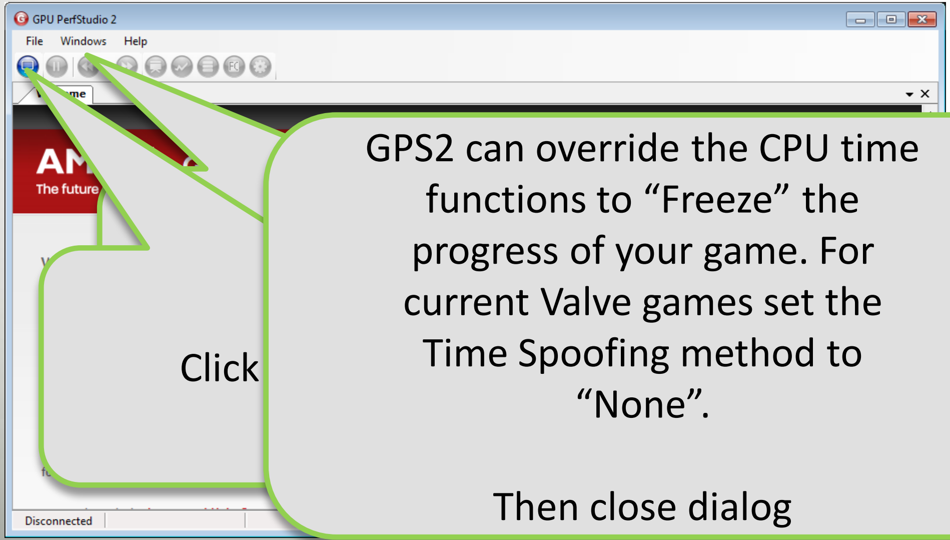
The Furmark startup shell script
(provided by Furmark)

GPU PerfStudio2 Linux startup

- To run Furmark with GPU PerfStudio

```
cd ~/Development/scripts
./furmark.sh
```

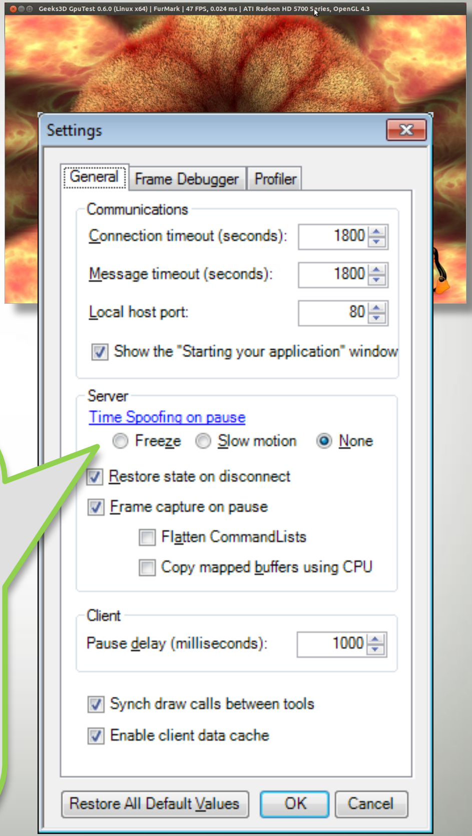
- Start GPUPerfClient on Windows®



Click

GPS2 can override the CPU time functions to “Freeze” the progress of your game. For current Valve games set the Time Spoofing method to “None”.

Then close dialog


Geeks3D GpuTest 0.6.0 (Linux x64) | FurMark | 47 FPS, 0.024 ms | ATI Radeon HD 5700 Series, OpenGL 4.3
Settings
General
Frame Debugger
Profiler
Restore All Default Values
OK
Cancel

Communications

Connection timeout (seconds): 1800

Message timeout (seconds): 1800

Local host port: 80

☑ Show the "Starting your application" window

Server

Time Spoofing on pause

○ Freeze ○ Slow motion ● None

☑ Restore state on disconnect

☑ Frame capture on pause

□ Flatten CommandLists

□ Copy mapped buffers using CPU

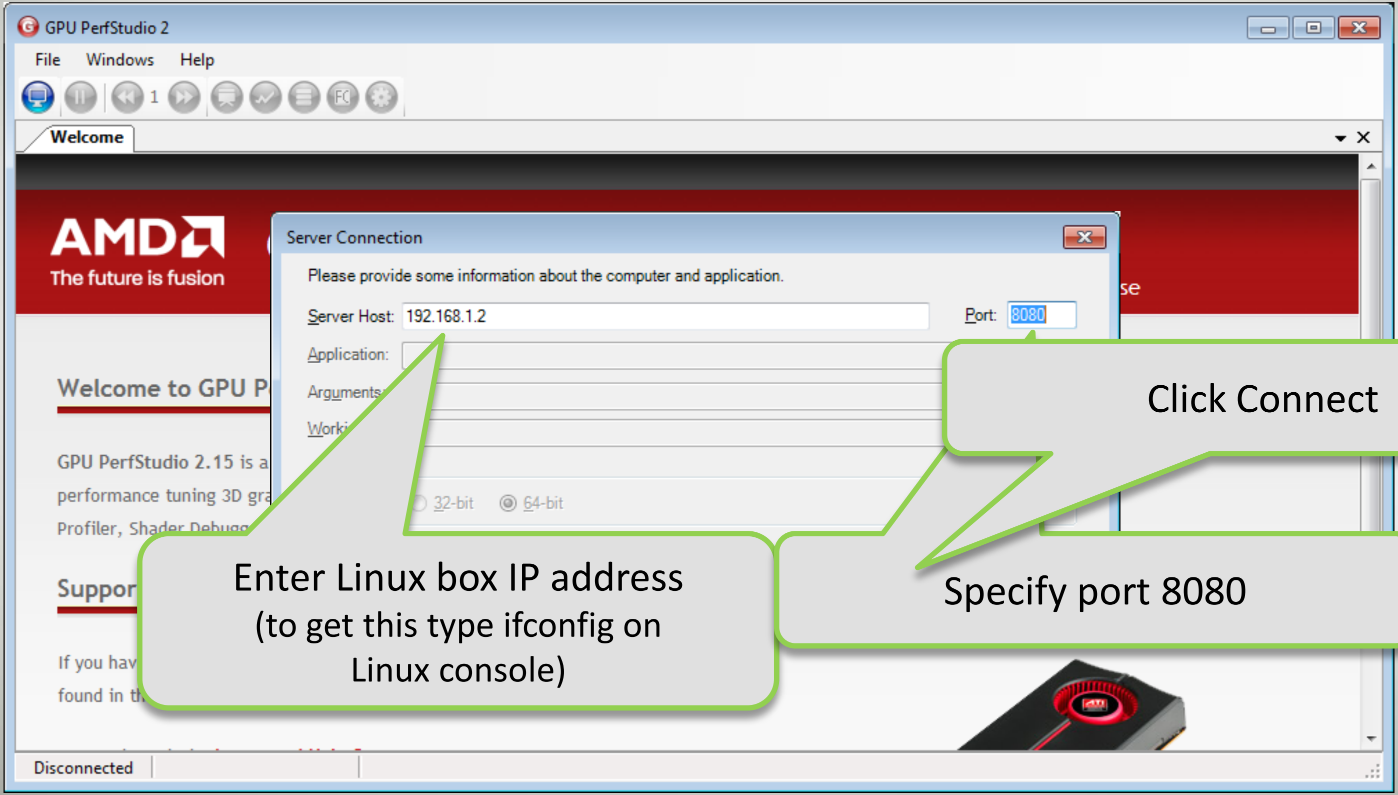
Client

Pause delay (milliseconds): 1000

☑ Synch draw calls between tools

☑ Enable client data cache

GPU PerfStudio2 Client Connection



Please provide some information about the computer and application.

Server Host: 192.168.1.2

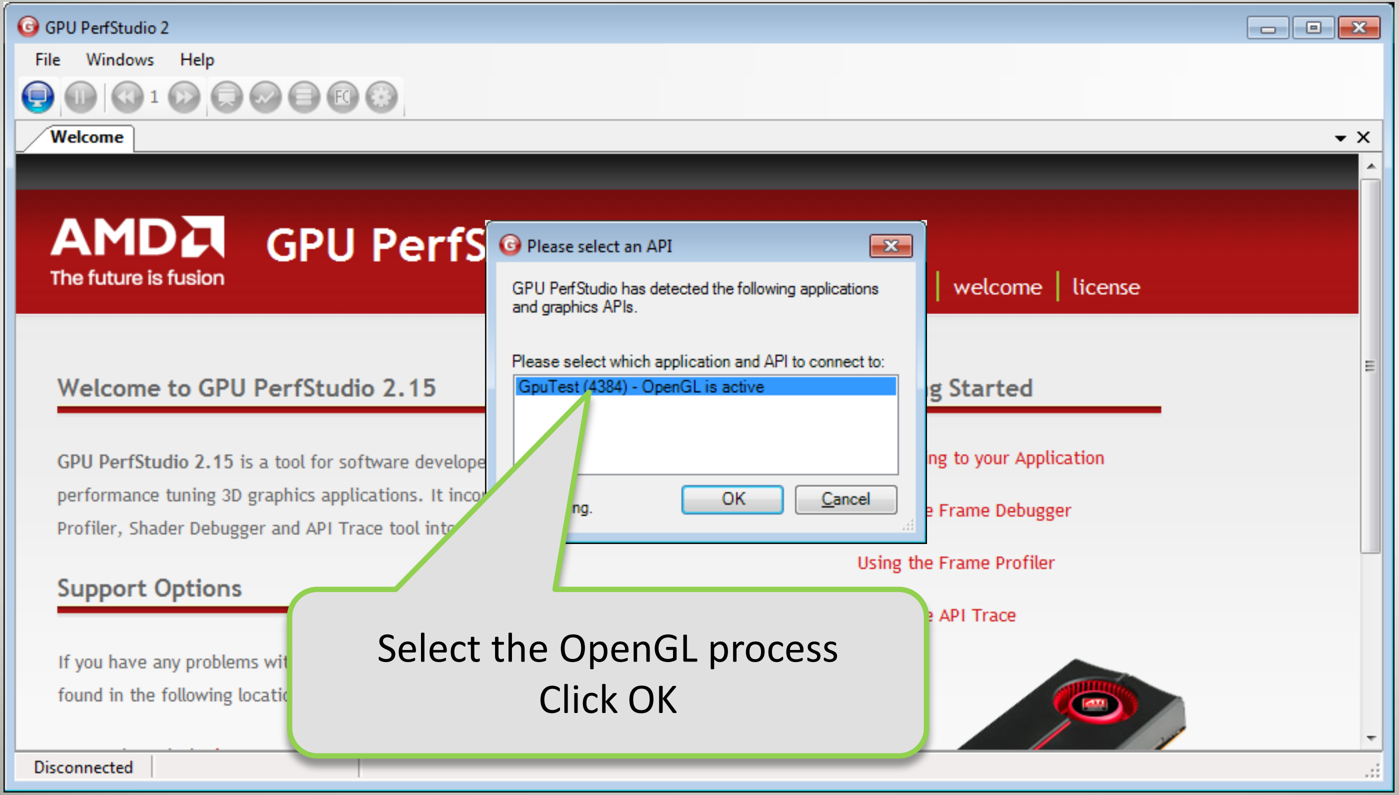
Port: 8080

Click Connect

Enter Linux box IP address (to get this type ifconfig on Linux console)

Specify port 8080

GPU PerfStudio2 Client Connection



Please select an API

GPU PerfStudio has detected the following applications and graphics APIs.

Please select which application and API to connect to:

GpuTest (4384) - OpenGL is active

ng.

OK

Cancel

Welcome to GPU PerfStudio 2.15

GPU PerfStudio 2.15 is a tool for software develope performance tuning 3D graphics applications. It inco Profiler, Shader Debugger and API Trace tool int

Support Options

If you have any problems wit found in the following locati

g Started

ng to your Application

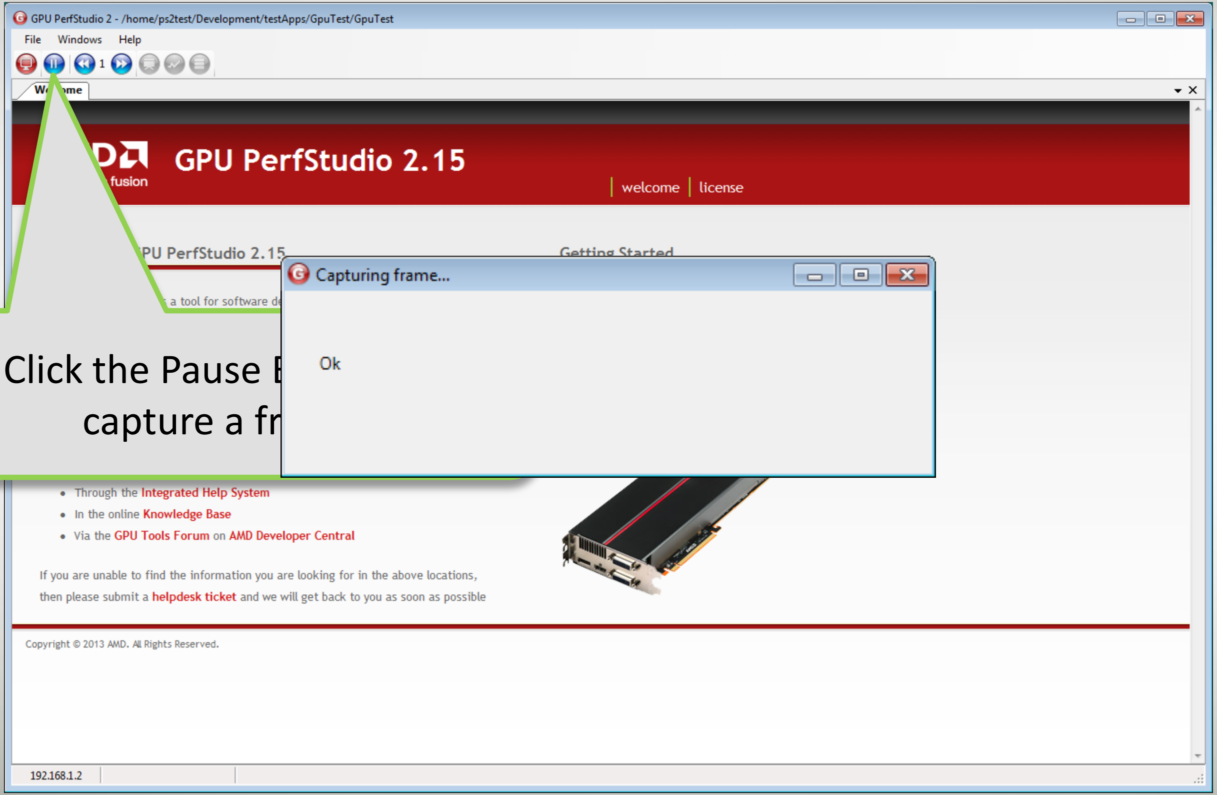
e Frame Debugger

Using the Frame Profiler

e API Trace

Select the OpenGL process
Click OK

GPU PerfStudio2 Client Connection



GPU PerfStudio 2 - /home/ps2test/Development/testApps/GpuTest/GpuTest

PU PerfStudio 2.15

a tool for software de

- Through the Integrated Help System
- In the online Knowledge Base
- Via the GPU Tools Forum on AMD Developer Central

If you are unable to find the information you are looking for in the above locations, then please submit a helpdesk ticket and we will get back to you as soon as possible

Copyright © 2013 AMD. All Rights Reserved.

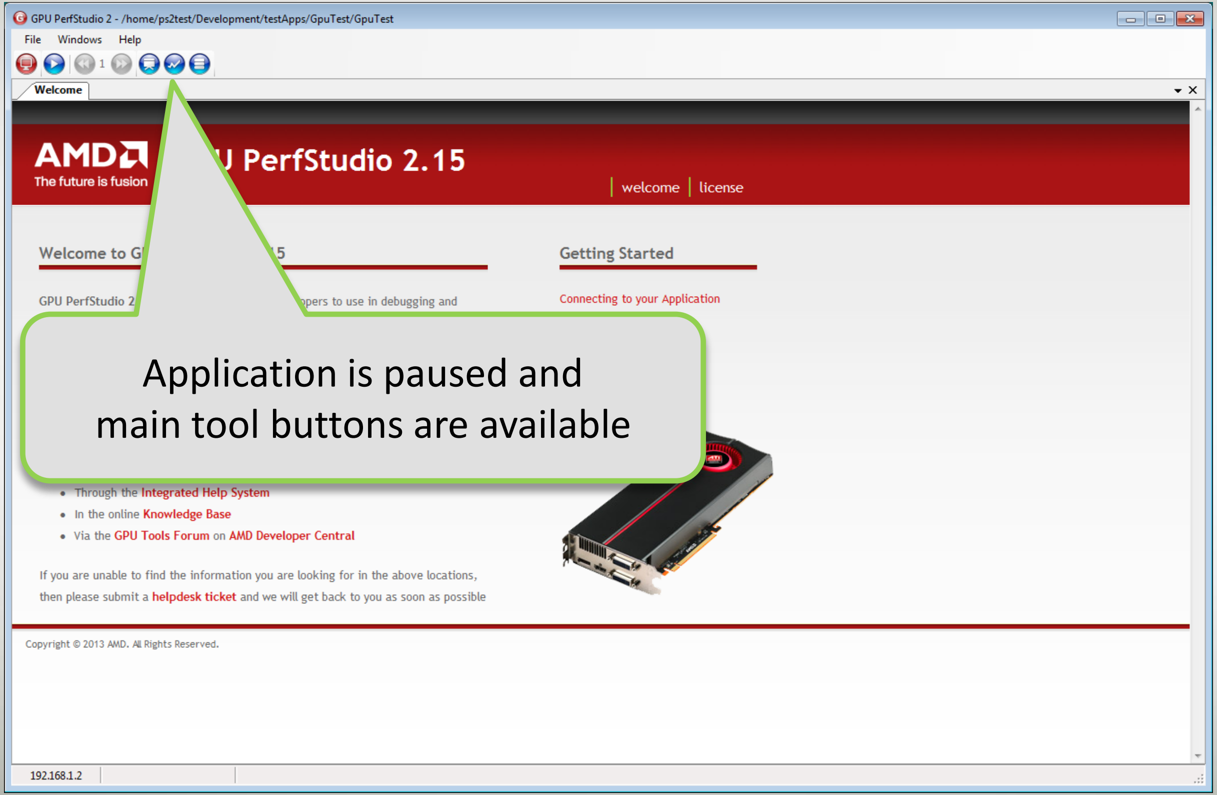
192.168.1.2

Capturing frame...

Ok

Click the Pause B
capture a fr

GPU PerfStudio2 Client Connection



Welcome to G 5

GPU PerfStudio 2 pers to use in debugging and

Getting Started

Connecting to your Application

- Through the Integrated Help System
- In the online Knowledge Base
- Via the GPU Tools Forum on AMD Developer Central

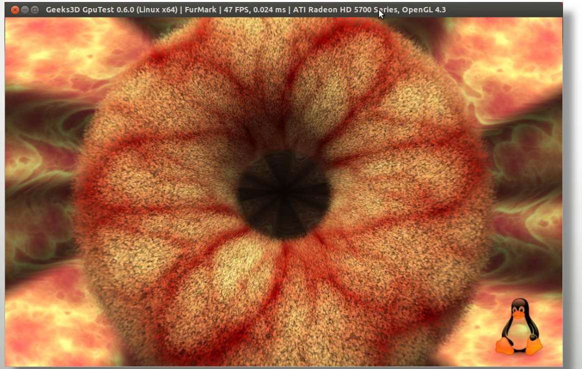
If you are unable to find the information you are looking for in the above locations, then please submit a helpdesk ticket and we will get back to you as soon as possible

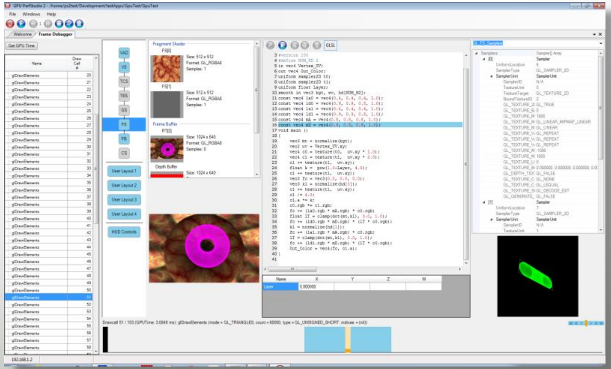
Copyright © 2013 AMD. All Rights Reserved.

Application is paused and main tool buttons are available

Seriously, Stop talking! Show me

- GPU PerfStudio2 running with GpuTest Furmark on Linux® and Windows®
- Overview of the Frame Debugger, Profiler and API Trace
- Using the Profiler and shader editor to optimize your shaders
- Running Windows® and Linux® sessions simultaneously (compare OpenGL on Windows® to OpenGL on Linux®)


Geeks3D GpuTest 0.6.0 (Linux x64) | FurMark | 47 FPS, 0.024 ms | ATI Radeon HD 5700 Series, OpenGL 4.3



```
#version 150
#define NUM_HD 2
in vec4 Vertex_UV;
out vec4 Out_Color;
uniform sampler2D t0;
uniform sampler2D t1;
uniform float layer;
smooth in vec3 bgt, ev, hd[NUM_HD];
const vec4 la0 = vec4(0.4, 0.4, 0.4, 1.0);
const vec4 ld0 = vec4(0.9, 0.9, 0.8, 1.0);
const vec4 la1 = vec4(0.4, 0.4, 0.4, 1.0);
const vec4 ld1 = vec4(0.9, 0.9, 0.8, 1.0);
const vec4 mA = vec4(0.8, 0.8, 0.8, 1.0);
const vec4 mD = vec4(0.9, 0.9, 0.9, 1.0);
void main ()
{
    vec3 mt = normalize(bgt);
    vec2 uv = Vertex_UV.xy;
    vec4 c0 = texture(t0,  uv.xy * 1.0);
    vec4 c1 = texture(t1,  uv.xy * 2.0);
    c1 += texture(t1,  uv.xy);
    float k =  pow(1.0-Layer, 4.0);
    c1 += texture(t1,  uv.xy);
    vec3 fc = vec3(0.0, 0.0, 0.0);
    vec3 k1 = normalize(hd[0]);
    c1 += texture(t1,  uv.xy);
    c1 /= 4.0;
    c1.a *= k;
    c0.rgb *= c1.rgb;
    fc += (la0.rgb * mA.rgb) * c0.rgb;
    float lT = clamp(dot(mt,k1), 0.0, 1.0);
    fc += (ld0.rgb * mD.rgb) * (lT * c0.rgb);
    k1 = normalize(hd[1]);
    fc += (la1.rgb * mA.rgb) * c0.rgb;
    lT = clamp(dot(mt,k1), 0.0, 1.0);
    fc += (ld1.rgb * mD.rgb) * (lT * c0.rgb);
    Out_Color = vec4(fc, c1.a);
}
```

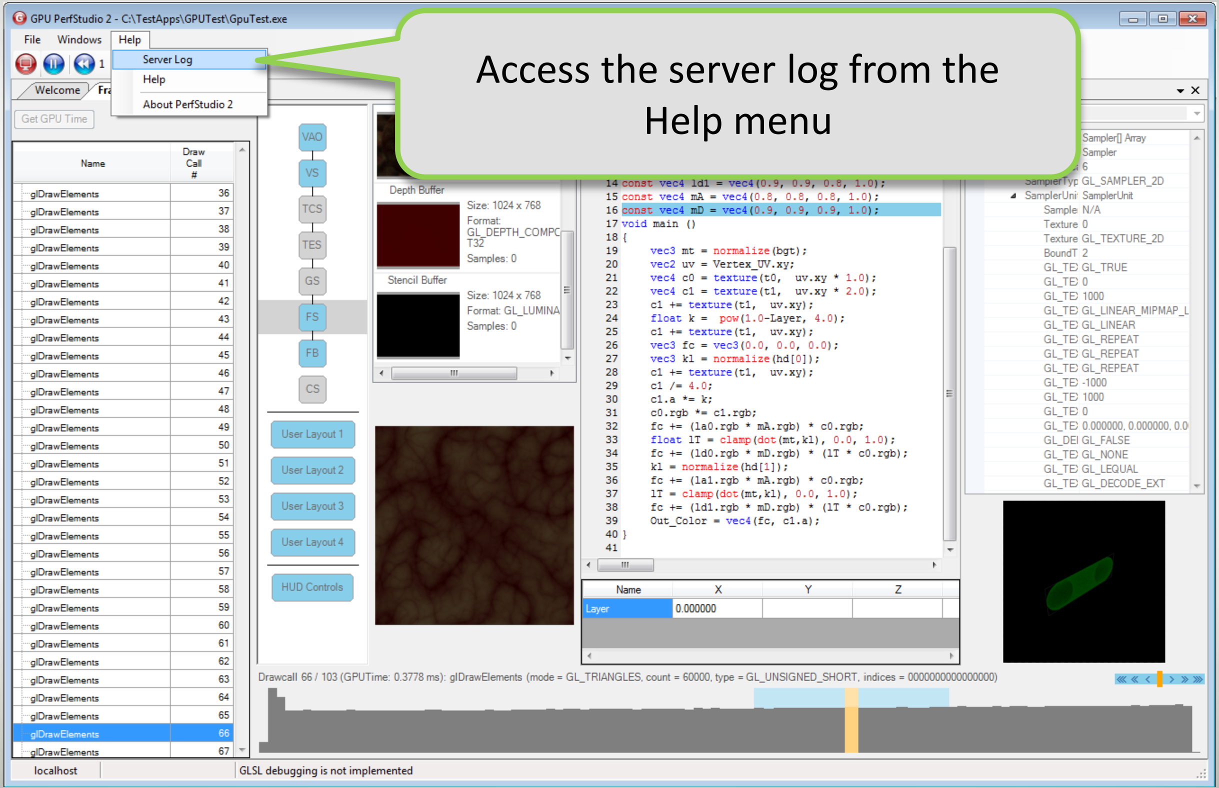
Name	X	Y	Z	W
Layer	0.000000			

DATA-MINING YOUR GAME USING GPU PERFSTUDIO2

Data-mining your game using GPU PerfStudio2

- As we mentioned earlier GPU PerfStudio2 has web-like behavior
- GPU PerfStudio2 modifies your game into a server that responds to specific commands for graphics API data
- The GPUPerfClient (a .NET app) makes requests to port 80 on Windows®, and port 8080 for Linux® servers
- You can see the requests for data in the console output of the server
- A history of the requests can be accessed in the GPUPerfClient

Data-mining your game using GPU PerfStudio2



Access the server log from the Help menu

Name	Draw Call #
glDrawElements	36
glDrawElements	37
glDrawElements	38
glDrawElements	39
glDrawElements	40
glDrawElements	41
glDrawElements	42
glDrawElements	43
glDrawElements	44
glDrawElements	45
glDrawElements	46
glDrawElements	47
glDrawElements	48
glDrawElements	49
glDrawElements	50
glDrawElements	51
glDrawElements	52
glDrawElements	53
glDrawElements	54
glDrawElements	55
glDrawElements	56
glDrawElements	57
glDrawElements	58
glDrawElements	59
glDrawElements	60
glDrawElements	61
glDrawElements	62
glDrawElements	63
glDrawElements	64
glDrawElements	65
glDrawElements	66
glDrawElements	67

Size: 1024 x 768
Format: GL_DEPTH_COMPC T32
Samples: 0

Size: 1024 x 768
Format: GL_LUMINA
Samples: 0

```
const vec4 ld1 = vec4(0.9, 0.9, 0.8, 1.0);
const vec4 mA = vec4(0.8, 0.8, 0.8, 1.0);
const vec4 mD = vec4(0.9, 0.9, 0.9, 1.0);
void main ()
{
    vec3 mt = normalize(bgt);
    vec2 uv = Vertex_UV.xy;
    vec4 c0 = texture(t0,  uv.xy * 1.0);
    vec4 c1 = texture(t1,  uv.xy * 2.0);
    c1 += texture(t1,  uv.xy);
    float k =  pow(1.0-Layer, 4.0);
    c1 += texture(t1,  uv.xy);
    vec3 fc = vec3(0.0, 0.0, 0.0);
    vec3 kl = normalize(hd[0]);
    c1 += texture(t1,  uv.xy);
    c1 /= 4.0;
    c1.a *= k;
    c0.rgb *= c1.rgb;
    fc += (la0.rgb * mA.rgb) * c0.rgb;
    float lT = clamp(dot(mt,kl), 0.0, 1.0);
    fc += (ld0.rgb * mD.rgb) * (lT * c0.rgb);
    kl = normalize(hd[1]);
    fc += (la1.rgb * mA.rgb) * c0.rgb;
    lT = clamp(dot(mt,kl), 0.0, 1.0);
    fc += (ld1.rgb * mD.rgb) * (lT * c0.rgb);
    Out_Color = vec4(fc, c1.a);
}
```

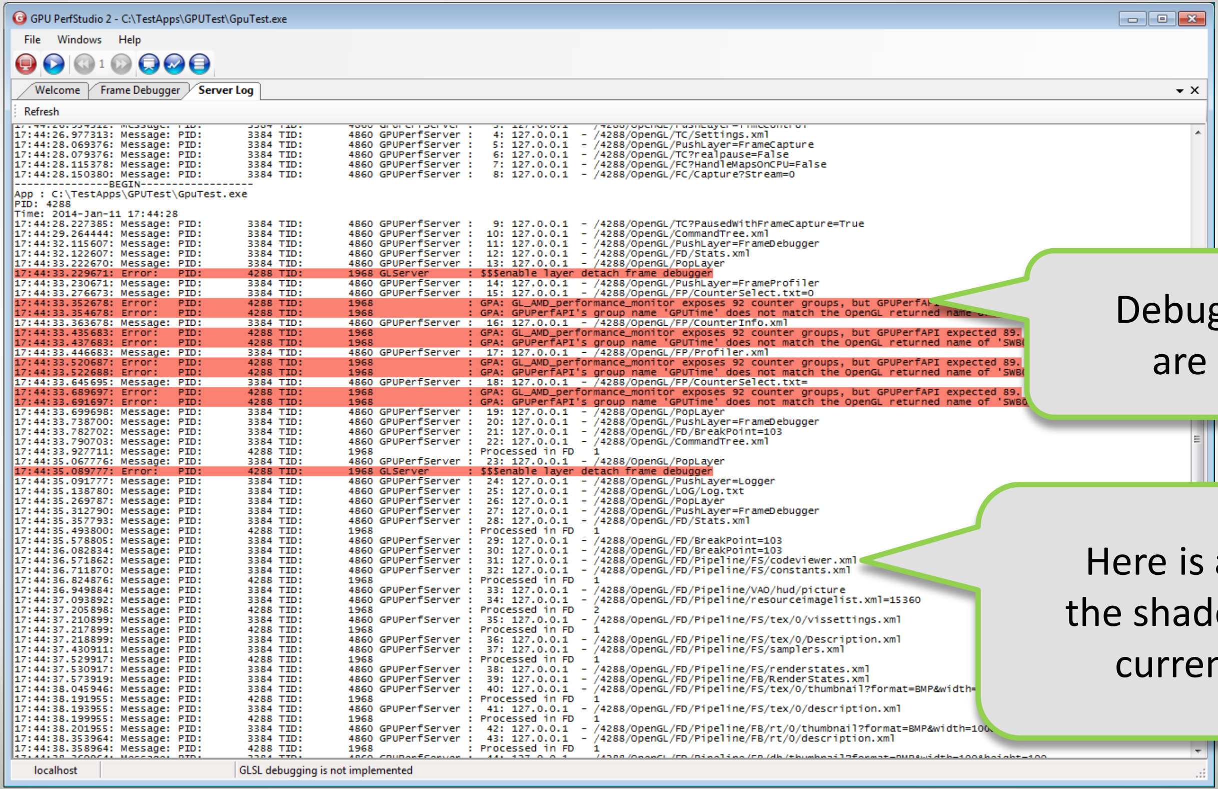
Name	X	Y	Z
Layer	0.000000		

	Sampler[] Array
	Sampler
	6
SamplerTyp	GL_SAMPLER_2D
SamplerUni	SamplerUnit
Sample	N/A
Texture	0
Texture	GL_TEXTURE_2D
BoundT	2
GL_TE	GL_TRUE
GL_TE	0
GL_TE	1000
GL_TE	GL_LINEAR_MIPMAP_L
GL_TE	GL_LINEAR
GL_TE	GL_REPEAT
GL_TE	GL_REPEAT
GL_TE	GL_REPEAT
GL_TE	-1000
GL_TE	1000
GL_TE	0
GL_TE	0.000000, 0.000000, 0.0
GL_DEI	GL_FALSE
GL_TE	GL_NONE
GL_TE	GL_LEQUAL
GL_TE	GL_DECODE_EXT

Drawcall 66 / 103 (GPUTime: 0.3778 ms): glDrawElements (mode = GL_TRIANGLES, count = 60000, type = GL_UNSIGNED_SHORT, indices = 0000000000000000)

localhost | GLSL debugging is not implemented

Data-mining your game using GPU PerfStudio2



```
17:44:26.977313: Message: PID:       3384 TID:       4860 GPUPerfServer :   4: 127.0.0.1  - /4288/OpenGL/TC/Settings.xml
17:44:28.069376: Message: PID:       3384 TID:       4860 GPUPerfServer :   5: 127.0.0.1  - /4288/OpenGL/PushLayer=FrameCapture
17:44:28.079376: Message: PID:       3384 TID:       4860 GPUPerfServer :   6: 127.0.0.1  - /4288/OpenGL/TC?realpause=False
17:44:28.115378: Message: PID:       3384 TID:       4860 GPUPerfServer :   7: 127.0.0.1  - /4288/OpenGL/FC?HandleMapsOnCPU=False
17:44:28.150380: Message: PID:       3384 TID:       4860 GPUPerfServer :   8: 127.0.0.1  - /4288/OpenGL/FC/Capture?Stream=0
---------------BEGIN------------------
App : C:\TestApps\GPUTest\GpuTest.exe
PID: 4288
Time: 2014-Jan-11 17:44:28
17:44:28.227385: Message: PID:       3384 TID:       4860 GPUPerfServer :   9: 127.0.0.1  - /4288/OpenGL/TC?PausedWithFrameCapture=True
17:44:29.264444: Message: PID:       3384 TID:       4860 GPUPerfServer :  10: 127.0.0.1  - /4288/OpenGL/CommandTree.xml
17:44:32.115607: Message: PID:       3384 TID:       4860 GPUPerfServer :  11: 127.0.0.1  - /4288/OpenGL/PushLayer=FrameDebugger
17:44:32.122607: Message: PID:       3384 TID:       4860 GPUPerfServer :  12: 127.0.0.1  - /4288/OpenGL/FD/Stats.xml
17:44:33.222670: Message: PID:       3384 TID:       4860 GPUPerfServer :  13: 127.0.0.1  - /4288/OpenGL/PopLayer
17:44:33.229671: Error:   PID:       4288 TID:       1968 GLServer      : $$$enable layer detach frame debugger
17:44:33.230671: Message: PID:       3384 TID:       4860 GPUPerfServer :  14: 127.0.0.1  - /4288/OpenGL/PushLayer=FrameProfiler
17:44:33.276673: Message: PID:       3384 TID:       4860 GPUPerfServer :  15: 127.0.0.1  - /4288/OpenGL/FP/CounterSelect.txt=0
17:44:33.352678: Error:   PID:       4288 TID:       1968               : GPA: GL_AMD_performance_monitor exposes 92 counter groups, but GPUPerfA
17:44:33.354678: Error:   PID:       4288 TID:       1968               : GPA: GPUPerfAPI's group name 'GPUTime' does not match the OpenGL returned name
17:44:33.363678: Message: PID:       3384 TID:       4860 GPUPerfServer :  16: 127.0.0.1  - /4288/OpenGL/FP/CounterInfo.xml
17:44:33.435683: Error:   PID:       4288 TID:       1968               : GPA: GL_AMD_performance_monitor exposes 92 counter groups, but GPUPerfAPI expected 89.
17:44:33.437683: Error:   PID:       4288 TID:       1968               : GPA: GPUPerfAPI's group name 'GPUTime' does not match the OpenGL returned name of 'SWB
17:44:33.446683: Message: PID:       3384 TID:       4860 GPUPerfServer :  17: 127.0.0.1  - /4288/OpenGL/FP/Profiler.xml
17:44:33.520687: Error:   PID:       4288 TID:       1968               : GPA: GL_AMD_performance_monitor exposes 92 counter groups, but GPUPerfAPI expected 89.
17:44:33.522688: Error:   PID:       4288 TID:       1968               : GPA: GPUPerfAPI's group name 'GPUTime' does not match the OpenGL returned name of 'SWB
17:44:33.645695: Message: PID:       3384 TID:       4860 GPUPerfServer :  18: 127.0.0.1  - /4288/OpenGL/FP/CounterSelect.txt=
17:44:33.689697: Error:   PID:       4288 TID:       1968               : GPA: GL_AMD_performance_monitor exposes 92 counter groups, but GPUPerfAPI expected 89.
17:44:33.691697: Error:   PID:       4288 TID:       1968               : GPA: GPUPerfAPI's group name 'GPUTime' does not match the OpenGL returned name of 'SWB
17:44:33.699698: Message: PID:       3384 TID:       4860 GPUPerfServer :  19: 127.0.0.1  - /4288/OpenGL/PopLayer
17:44:33.738700: Message: PID:       3384 TID:       4860 GPUPerfServer :  20: 127.0.0.1  - /4288/OpenGL/PushLayer=FrameDebugger
17:44:33.782702: Message: PID:       3384 TID:       4860 GPUPerfServer :  21: 127.0.0.1  - /4288/OpenGL/FD/BreakPoint=103
17:44:33.790703: Message: PID:       3384 TID:       4860 GPUPerfServer :  22: 127.0.0.1  - /4288/OpenGL/CommandTree.xml
17:44:33.927711: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:35.067776: Message: PID:       3384 TID:       4860 GPUPerfServer :  23: 127.0.0.1  - /4288/OpenGL/PopLayer
17:44:35.089777: Error:   PID:       4288 TID:       1968 GLServer      : $$$enable layer detach frame debugger
17:44:35.091777: Message: PID:       3384 TID:       4860 GPUPerfServer :  24: 127.0.0.1  - /4288/OpenGL/PushLayer=Logger
17:44:35.138780: Message: PID:       3384 TID:       4860 GPUPerfServer :  25: 127.0.0.1  - /4288/OpenGL/LOG/Log.txt
17:44:35.269787: Message: PID:       3384 TID:       4860 GPUPerfServer :  26: 127.0.0.1  - /4288/OpenGL/PopLayer
17:44:35.312790: Message: PID:       3384 TID:       4860 GPUPerfServer :  27: 127.0.0.1  - /4288/OpenGL/PushLayer=FrameDebugger
17:44:35.357793: Message: PID:       3384 TID:       4860 GPUPerfServer :  28: 127.0.0.1  - /4288/OpenGL/FD/Stats.xml
17:44:35.493800: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:35.578805: Message: PID:       3384 TID:       4860 GPUPerfServer :  29: 127.0.0.1  - /4288/OpenGL/FD/BreakPoint=103
17:44:36.082834: Message: PID:       3384 TID:       4860 GPUPerfServer :  30: 127.0.0.1  - /4288/OpenGL/FD/BreakPoint=103
17:44:36.571862: Message: PID:       3384 TID:       4860 GPUPerfServer :  31: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/codeviewer.xml
17:44:36.711870: Message: PID:       3384 TID:       4860 GPUPerfServer :  32: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/constants.xml
17:44:36.824876: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:36.949884: Message: PID:       3384 TID:       4860 GPUPerfServer :  33: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/VAO/hud/picture
17:44:37.093892: Message: PID:       3384 TID:       4860 GPUPerfServer :  34: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/resourceimagelist.xml=15360
17:44:37.205898: Message: PID:       4288 TID:       1968               : Processed in FD    2
17:44:37.210899: Message: PID:       3384 TID:       4860 GPUPerfServer :  35: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/tex/0/vissettings.xml
17:44:37.217899: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:37.218899: Message: PID:       3384 TID:       4860 GPUPerfServer :  36: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/tex/0/Description.xml
17:44:37.430911: Message: PID:       3384 TID:       4860 GPUPerfServer :  37: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/samplers.xml
17:44:37.529917: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:37.530917: Message: PID:       3384 TID:       4860 GPUPerfServer :  38: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/renderstates.xml
17:44:37.573919: Message: PID:       3384 TID:       4860 GPUPerfServer :  39: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FB/RenderStates.xml
17:44:38.045946: Message: PID:       3384 TID:       4860 GPUPerfServer :  40: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/tex/0/thumbnail?format=BMP&width=
17:44:38.191955: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:38.193955: Message: PID:       3384 TID:       4860 GPUPerfServer :  41: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FS/tex/0/description.xml
17:44:38.199955: Message: PID:       4288 TID:       1968               : Processed in FD    1
17:44:38.201955: Message: PID:       3384 TID:       4860 GPUPerfServer :  42: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FB/rt/0/thumbnail?format=BMP&width=100
17:44:38.353964: Message: PID:       3384 TID:       4860 GPUPerfServer :  43: 127.0.0.1  - /4288/OpenGL/FD/Pipeline/FB/rt/0/description.xml
17:44:38.358964: Message: PID:       4288 TID:       1968               : Processed in FD    1
```

localhost

GLSL debugging is not implemented

Debug messages are included

Here is a request for the shader code at the current draw call

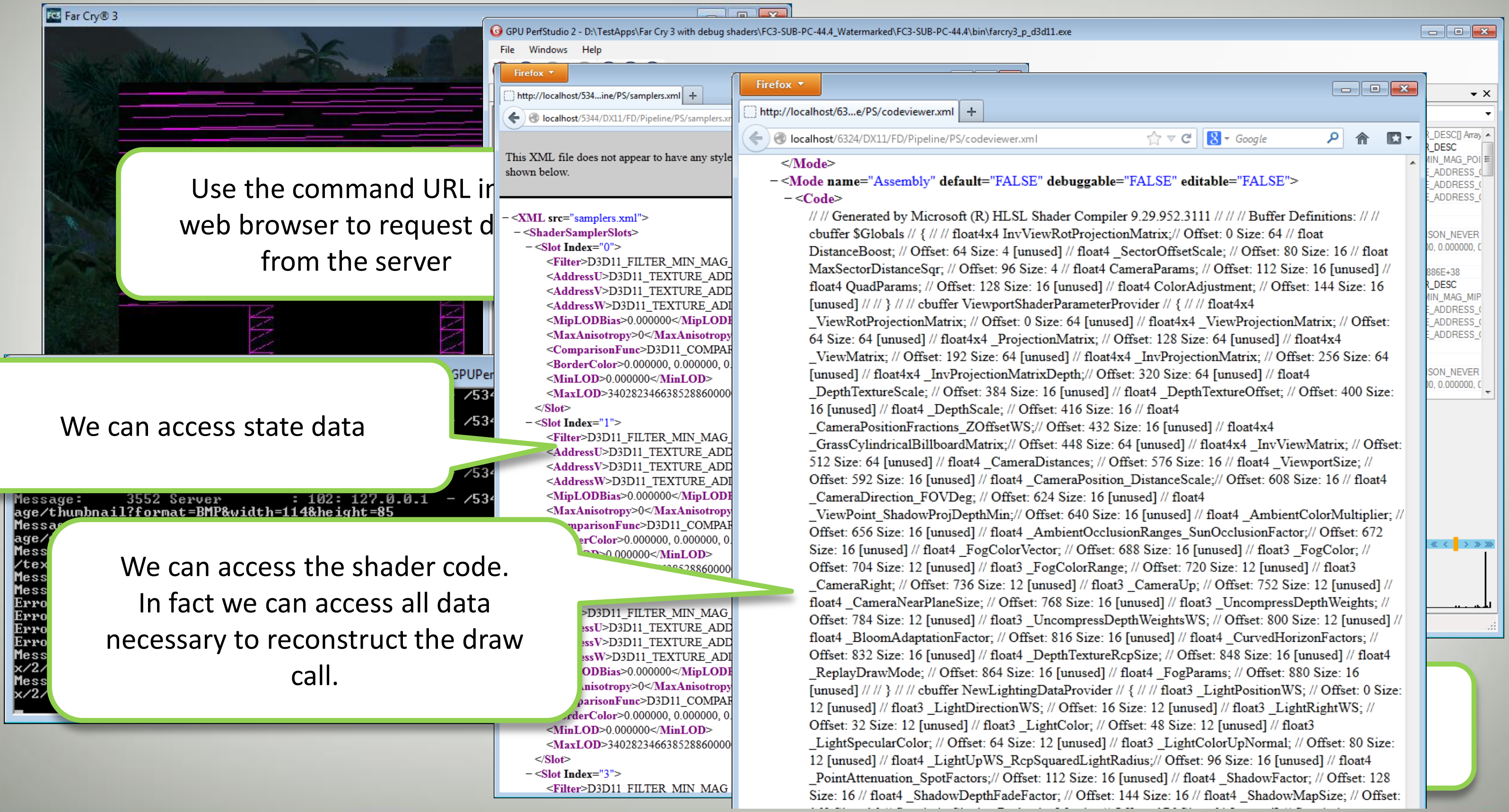
Data-mining your game using GPU PerfStudio2

- Requests to the server are in the form:

```
http://192.168.1.2/2876/OpenGL/FD/Pipeline/FS/codeviewer.xml
```

- It is possible to use PerfStudio2 web requests in scripts to automate and customize access to your app data
- As part of the work carried out on Far Cry3 we needed to know where specific sections of HLSL code were being used in a frame
- We were able to use a script to retrieve the HLSL code from each draw call in a frame an search the code for keywords that would identify the code.

Data-mining your game using GPU PerfStudio2


Far Cry® 3
GPU PerfStudio 2 - D:\TestApps\Far Cry 3 with debug shaders\FC3-SUB-PC-44.4_Watermarked\FC3-SUB-PC-44.4\bin\farcry3_p_d3d11.exe
File
Windows
Help
Firefox
http://localhost/534...ine/PS/samplers.xml
localhost/5344/DX11/FD/Pipeline/PS/samplers.xml
Firefox
http://localhost/63...e/PS/codeviewer.xml
localhost/6324/DX11/FD/Pipeline/PS/codeviewer.xml
Google
GPUPer

Use the command URL in web browser to request d from the server

We can access state data

We can access the shader code. In fact we can access all data necessary to reconstruct the draw call.

```
Message:    3552 Server       : 102: 127.0.0.1  - /534
age/thumbnail?format=BMP&width=114&height=85
```

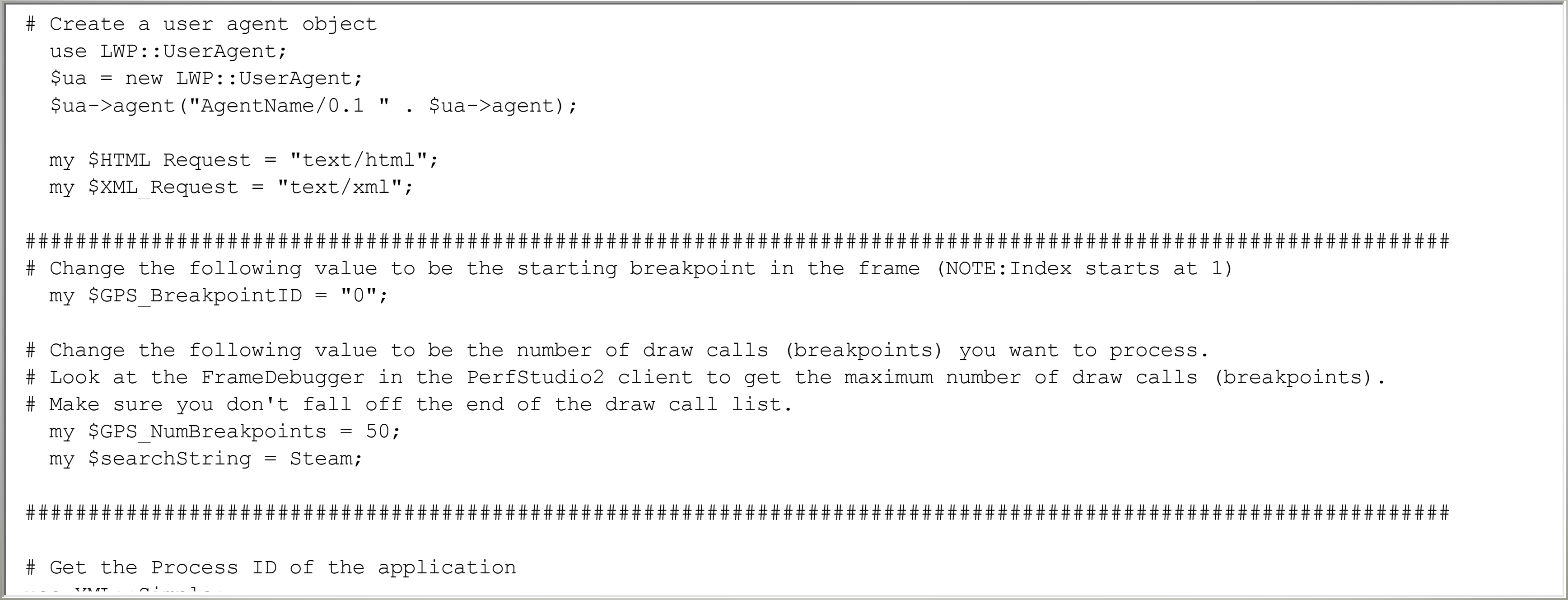
This XML file does not appear to have any style shown below.

```
- <XML src="samplers.xml">
  - <ShaderSamplerSlots>
    - <Slot Index="0">
        <Filter>D3D11_FILTER_MIN_MAG_
        <AddressU>D3D11_TEXTURE_ADD
        <AddressV>D3D11_TEXTURE_ADD
        <AddressW>D3D11_TEXTURE_ADI
        <MipLODBias>0.000000</MipLODE
        <MaxAnisotropy>0</MaxAnisotropy
        <ComparisonFunc>D3D11_COMPAR
        <BorderColor>0.000000, 0.000000, 0.
        <MinLOD>0.000000</MinLOD>
        <MaxLOD>340282346638528860000
      </Slot>
    - <Slot Index="1">
        <Filter>D3D11_FILTER_MIN_MAG_
        <AddressU>D3D11_TEXTURE_ADD
        <AddressV>D3D11_TEXTURE_ADD
        <AddressW>D3D11_TEXTURE_ADI
        <MipLODBias>0.000000</MipLODE
        <MaxAnisotropy>0</MaxAnisotropy
        mparisonFunc>D3D11_COMPAR
        rColor>0.000000, 0.000000, 0.
        D>0.000000</MinLOD>
        >D3D11_FILTER_MIN_MAG_
        ssU>D3D11_TEXTURE_ADD
        ssV>D3D11_TEXTURE_ADD
        ssW>D3D11_TEXTURE_ADI
        ODBias>0.000000</MipLODE
        nisotropy>0</MaxAnisotropy
        parisonFunc>D3D11_COMPAR
        rderColor>0.000000, 0.000000, 0.
        <MinLOD>0.000000</MinLOD>
        <MaxLOD>340282346638528860000
      </Slot>
    - <Slot Index="3">
        <Filter>D3D11_FILTER_MIN_MAG_
```

```
  </Mode>
- <Mode name="Assembly" default="FALSE" debuggable="FALSE" editable="FALSE">
  - <Code>
      // // Generated by Microsoft (R) HLSL Shader Compiler 9.29.952.3111 // // // Buffer Definitions: // //
      cbuffer $Globals // { // // float4x4 InvViewRotProjectionMatrix;// Offset: 0 Size: 64 // float
      DistanceBoost; // Offset: 64 Size: 4 [unused] // float4 _SectorOffsetScale; // Offset: 80 Size: 16 // float
      MaxSectorDistanceSqr; // Offset: 96 Size: 4 // float4 CameraParams; // Offset: 112 Size: 16 [unused] //
      float4 QuadParams; // Offset: 128 Size: 16 [unused] // float4 ColorAdjustment; // Offset: 144 Size: 16
      [unused] // // } // // cbuffer ViewportShaderParameterProvider // { // // float4x4
      _ViewRotProjectionMatrix; // Offset: 0 Size: 64 [unused] // float4x4 _ViewProjectionMatrix; // Offset:
      64 Size: 64 [unused] // float4x4 _ProjectionMatrix; // Offset: 128 Size: 64 [unused] // float4x4
      _ViewMatrix; // Offset: 192 Size: 64 [unused] // float4x4 _InvProjectionMatrix; // Offset: 256 Size: 64
      [unused] // float4x4 _InvProjectionMatrixDepth;// Offset: 320 Size: 64 [unused] // float4
      _DepthTextureScale; // Offset: 384 Size: 16 [unused] // float4 _DepthTextureOffset; // Offset: 400 Size:
      16 [unused] // float4 _DepthScale; // Offset: 416 Size: 16 // float4
      _CameraPositionFractions_ZOffsetWS;// Offset: 432 Size: 16 [unused] // float4x4
      _GrassCylindricalBillboardMatrix;// Offset: 448 Size: 64 [unused] // float4x4 _InvViewMatrix; // Offset:
      512 Size: 64 [unused] // float4 _CameraDistances; // Offset: 576 Size: 16 // float4 _ViewportSize; //
      Offset: 592 Size: 16 [unused] // float4 _CameraPosition_DistanceScale;// Offset: 608 Size: 16 // float4
      _CameraDirection_FOVDeg; // Offset: 624 Size: 16 [unused] // float4
      _ViewPoint_ShadowProjDepthMin;// Offset: 640 Size: 16 [unused] // float4 _AmbientColorMultiplier; //
      Offset: 656 Size: 16 [unused] // float4 _AmbientOcclusionRanges_SunOcclusionFactor;// Offset: 672
      Size: 16 [unused] // float4 _FogColorVector; // Offset: 688 Size: 16 [unused] // float3 _FogColor; //
      Offset: 704 Size: 12 [unused] // float3 _FogColorRange; // Offset: 720 Size: 12 [unused] // float3
      _CameraRight; // Offset: 736 Size: 12 [unused] // float3 _CameraUp; // Offset: 752 Size: 12 [unused] //
      float4 _CameraNearPlaneSize; // Offset: 768 Size: 16 [unused] // float3 _UncompressDepthWeights; //
      Offset: 784 Size: 12 [unused] // float3 _UncompressDepthWeightsWS; // Offset: 800 Size: 12 [unused] //
      float4 _BloomAdaptationFactor; // Offset: 816 Size: 16 [unused] // float4 _CurvedHorizonFactors; //
      Offset: 832 Size: 16 [unused] // float4 _DepthTextureRcpSize; // Offset: 848 Size: 16 [unused] // float4
      _ReplayDrawMode; // Offset: 864 Size: 16 [unused] // float4 _FogParams; // Offset: 880 Size: 16
      [unused] // // } // // cbuffer NewLightingDataProvider // { // // float3 _LightPositionWS; // Offset: 0 Size:
      12 [unused] // float3 _LightDirectionWS; // Offset: 16 Size: 12 [unused] // float3 _LightRightWS; //
      Offset: 32 Size: 12 [unused] // float3 _LightColor; // Offset: 48 Size: 12 [unused] // float3
      _LightSpecularColor; // Offset: 64 Size: 12 [unused] // float3 _LightColorUpNormal; // Offset: 80 Size:
      12 [unused] // float4 _LightUpWS_RcpSquaredLightRadius;// Offset: 96 Size: 16 [unused] // float4
      _PointAttenuation_SpotFactors;// Offset: 112 Size: 16 [unused] // float4 _ShadowFactor; // Offset: 128
      Size: 16 // float4 _ShadowDepthFadeFactor; // Offset: 144 Size: 16 // float4 _ShadowMapSize; // Offset:
```

Data-mining your game using GPU PerfStudio2

Script that searches the first 50 draw calls for fragment shaders that contain the string “Steam”.



```
# Create a user agent object
  use LWP::UserAgent;
  $ua = new LWP::UserAgent;
  $ua->agent("AgentName/0.1 " . $ua->agent);

  my $HTML_Request = "text/html";
  my $XML_Request = "text/xml";

#################################################################################################################
# Change the following value to be the starting breakpoint in the frame (NOTE:Index starts at 1)
  my $GPS_BreakpointID = "0";

# Change the following value to be the number of draw calls (breakpoints) you want to process.
# Look at the FrameDebugger in the PerfStudio2 client to get the maximum number of draw calls (breakpoints).
# Make sure you don't fall off the end of the draw call list.
  my $GPS_NumBreakpoints = 50;
  my $searchString = Steam;

#################################################################################################################

# Get the Process ID of the application
use XML::Simple;
```

Data-mining your game using GPU PerfStudio2

- NOTE for Linux® users
- Port 80 is not available in user mode for web access
- GPU PerfStudio2 for Linux has a script to redirect web access to port 8080
- You can find the script in the GPUPerfStudio directory

```
redirport80.sh
```

Starting Steam for Linux games with GPS2

- Steam games for Linux are downloaded to:

```
~/.steam/steam/SteamApps/common/
```

- DOTA2 is downloaded to:

```
~/.steam/steam/SteamApps/common/Dota 2 beta
```

- In this directory is a shell script named “**dota.sh**”, edit it as follows:

1. Change the export LD_LIBRARY_PATH to point to the GPS2 server folder:

```
Export LD_LIBRARY_PATH="${GAMEROOT}"/bin:~/Development/GPUPerfStudio/x86:$LD_LIBRARY_PATH
```

2. Set the GAME_DEBUGGER option as follows:

```
GAME_DEBUGGER=~/Development/GPUPerfStudio/x86/GPUPerfServer
```

Starting Steam for Linux games with GPS2

- To run the game
  - Make sure the steam executable isn't running. If it is, it will show up in the app bar on the left of the screen. This will ensure that GPU PerfStudio2 will use the console window for output
  - Each Steam game has its own ID - DOTA2 is 570
  - go to root steam directory:

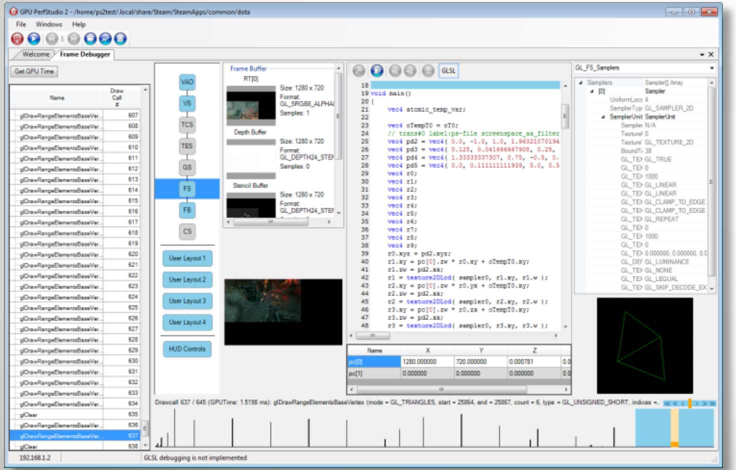
```
$ cd ~/.steam/steam
```

  - From there, type:

```
$ steam steam://rungameid/570
```

Data-mining your game using GPU PerfStudio2

- Demonstration of more profiler features
- Demonstration of Scripting DOTA2 (Linux)



Name	Draw Call #
glDrawRangeElementsBaseVer...	607
glDrawRangeElementsBaseVer...	608
glDrawRangeElementsBaseVer...	609
glDrawRangeElementsBaseVer...	610
glDrawRangeElementsBaseVer...	611
glDrawRangeElementsBaseVer...	612
glDrawRangeElementsBaseVer...	613
glDrawRangeElementsBaseVer...	614
glDrawRangeElementsBaseVer...	615
glDrawRangeElementsBaseVer...	616
glDrawRangeElementsBaseVer...	617
glDrawRangeElementsBaseVer...	618
glDrawRangeElementsBaseVer...	619
glDrawRangeElementsBaseVer...	620
glDrawRangeElementsBaseVer...	621
glDrawRangeElementsBaseVer...	622
glDrawRangeElementsBaseVer...	623
glDrawRangeElementsBaseVer...	624
glDrawRangeElementsBaseVer...	625
glDrawRangeElementsBaseVer...	626
glDrawRangeElementsBaseVer...	627
glDrawRangeElementsBaseVer...	628
glDrawRangeElementsBaseVer...	629
glDrawRangeElementsBaseVer...	630
glDrawRangeElementsBaseVer...	631
glDrawRangeElementsBaseVer...	632
glDrawRangeElementsBaseVer...	633
glDrawRangeElementsBaseVer...	634
glClear	635
glDrawRangeElementsBaseVer...	636
glDrawRangeElementsBaseVer...	637
glClear	638

Frame Buffer
RT[0]
Size: 1280 x 720
Format: GL_SRGB8_ALPHA
Samples: 1

Depth Buffer
Size: 1280 x 720
Format: GL_DEPTH24_STE
Samples: 0

Stencil Buffer
Size: 1280 x 720
Format: GL_DEPTH24_STE

```

void main()
{
    vec4 atomic_temp_var;

    vec4 oTempT0 = oT0;
    // trans#0 label:ps-file screenspace_aa_filter
    vec4 pd2 = vec4( 0.0, -1.0, 1.0, 1.96321070194
    vec4 pd3 = vec4( 0.125, 0.04166666790S, 0.25,
    vec4 pd4 = vec4( 1.33333337307, 0.75, -0.5, 0.
    vec4 pd5 = vec4( 0.0, 0.111111111939, 5.0, 0.5
    vec4 r0;
    vec4 r1;
    vec4 r2;
    vec4 r3;
    vec4 r4;
    vec4 r5;
    vec4 r6;
    vec4 r7;
    vec4 r8;
    vec4 r9;
    r0.xyz = pd2.xyz;
    r1.xy = pc[0].zw * r0.xy + oTempT0.xy;
    r1.zw = pd2.xx;
    r1 = texture2DLod( sampler0, r1.xy, r1.w );
    r2.xy = pc[0].zw * r0.yx + oTempT0.xy;
    r2.zw = pd2.xx;
    r2 = texture2DLod( sampler0, r2.xy, r2.w );
    r3.xy = pc[0].zw * r0.zx + oTempT0.xy;
    r3.zw = pd2.xx;
    r3 = texture2DLod( sampler0, r3.xy, r3.w );
```

Name	X	Y	Z	
pc[0]	1280.000000	720.000000	0.000781	0.0
pc[1]	0.000000	0.000000	0.000000	0.0

Samplers	Sampler[] Array
[0]	Sampler
UniformLoca	4
SamplerTyp	GL_SAMPLER_2D
SamplerUnit	SamplerUnit
Sampler	N/A
Texture	0
Texture	GL_TEXTURE_2D
BoundT	38
GL_TE	GL_TRUE
GL_TE	0
GL_TE	1000
GL_TE	GL_LINEAR
GL_TE	GL_LINEAR
GL_TE	GL_CLAMP_TO_EDGE
GL_TE	GL_CLAMP_TO_EDGE
GL_TE	GL_REPEAT
GL_TE	0
GL_TE	1000
GL_TE	0
GL_TE	0.000000, 0.000000, 0.0
GL_DEF	GL_LUMINANCE
GL_TE	GL_NONE
GL_TE	GL_LEQUAL
GL_TE	GL_SKIP_DECODE_EX

Drawcall 637 / 645 (GPUTime: 1.5198 ms): glDrawRangeElementsBaseVertex (mode = GL_TRIANGLES, start = 25864, end = 25867, count = 6, type = GL_UNSIGNED_SHORT, indices =..

192.168.1.2

GLSL debugging is not implemented

GPU PerfStudio2 and APITrace

- APITrace - https://github.com/apitrace/apitrace
  - Trace OpenGL, OpenGL ES, Direct3D, and DirectDraw APIs calls to a file
  - Replay OpenGL and OpenGL ES calls from a file
  - Inspect OpenGL state at any call while retracing
  - Visualize and edit trace files
- Use APITrace to capture OpenGL traces on Linux® or Windows® and playback on either
- GPU PerfStudio2 supports the playback of traces allowing you to debug and optimize using a small subset of game frames
- Ideal for capturing rendering issues and sharing them between developers for solutions

GPU PerfStudio2 Latest Version

What's new in GPS2.14?

- Hardware counter support for AMD "Hawaii" (R9 290 series) GPU's
- Improved support for multithreaded applications
- Pipeline specific counters for OpenGL
- Support for OpenGL Compute

Currently in development

- Support for Linux®/OpenGL applications
- Support for Mantle on Windows7®

Summary

- GPU PerfStudio2 is AMD's performance and debugging tool for graphics applications
- A suite of tools that can be used to debug and increase performance on AMD GPUs
- Works on Windows® and Linux®
- Ideal for debugging and optimizing OpenGL games on Windows® and Linux
- Supports Steam for Linux games
- Available end of Q1 2014

Thank You

- Rich Geldreich, Jason Mitchell and all at Valve who have used and supported GPUPerfStudio2
- Dan Ginsburg, Peter Lohrmann, and Graham Sellers for OpenGL support
- Valve for inviting us to attend and present at Steam Dev Days 2014
- All who attended this presentation

AMD Graphics Tools Download Information

- All AMD Graphics Tools
  - http://developer.amd.com/tools-and-sdks/graphics-development/
- GPU PerfStudio2
  - http://developer.amd.com/tools-and-sdks/graphics-development/gpu-perfstudio-2/
- GPUPerfAPI – Performance Counter Library
  - http://developer.amd.com/tools-and-sdks/graphics-development/gpuperfapi/
- CodeXL – GPU debugging for OpenCL™ & OpenGL API calls and OpenCL™ kernel
  - http://developer.amd.com/tools-and-sdks/heterogeneous-computing/codexl/
- gDEBugger – OpenCL/OpenGL debugger (end-of-line)
  - http://developer.amd.com/tools-and-sdks/heterogeneous-computing/archived-tools/amd-gdebugger/

Questions?

Gordon Selley

gordon.selley@amd.com

Tony Hosier

tony.hosier@amd.com

Download AMD Graphics Tools

http://developer.amd.com/tools-and-sdks/graphics-development/

STEAM DEV DAYS

GORDON SELLEY
TONY HOSIER
AMD GPU DEVELOPER TOOLS TEAM

OPTIMIZING LINUX GAMES FOR AMD GRAPHICS USING GPU PERFSTUDIO2