

Optimizing and Debugging Graphics Applications with AMD's GPU PerfStudio 2.2

GPG Developer Tools

Raul Aguaviva

Gordon Selley

Seth Sowerby



Introducing GPU PerfStudio 2.2

- A GPU Performance Analysis tool from AMD
 - Help you create more stunning games
 - Designed for Game Developers
 - Currently being used in AAA Game development
 - Used in AMD by driver and performance teams
- Debug and Profile DirectX®11, DirectX®10, and OpenGL now on ATI Radeon HD 5xxx Series GPU's now!
- Debug and Profile DirectX®10, and OpenGL, on 5xxx, 4xxx, 3xxx and 2xxx series GPU's.



Introducing GPU PerfStudio 2.2

- Frame Debugger
 - Select a draw call
 - View ALL game resources bound to the pipeline
 - Frame Capture
- Shader Debugger
 - Debug HLSL and Assembly code from inside your app
 - Step, breakpoints, inspect register values.
- Frame Profiler
 - Identify costly draw calls and
 - Identify GPU bottlenecks
- API Trace viewer
 - Inspect all API calls
 - Jump to draw call in Frame Profiler and Frame Debugger



GPU PerfStudio 2.2 - Requirements

- Open Development Tool
 - Works on all vendor hardware
 - Server - Win7, Vista (32 and 64bit versions)
 - Client – Win7, Vista, XP.
 - Supports OpenGL 3.0, DirectX®10 & 10.1, DirectX®11.
 - No custom driver requirements
 - Small footprint – no installation
 - No modifications to target application necessary
 - Simple drag and drop to start debugging!
 - Its Free!
 - \$0.00
 - £0.00
 - €0.00

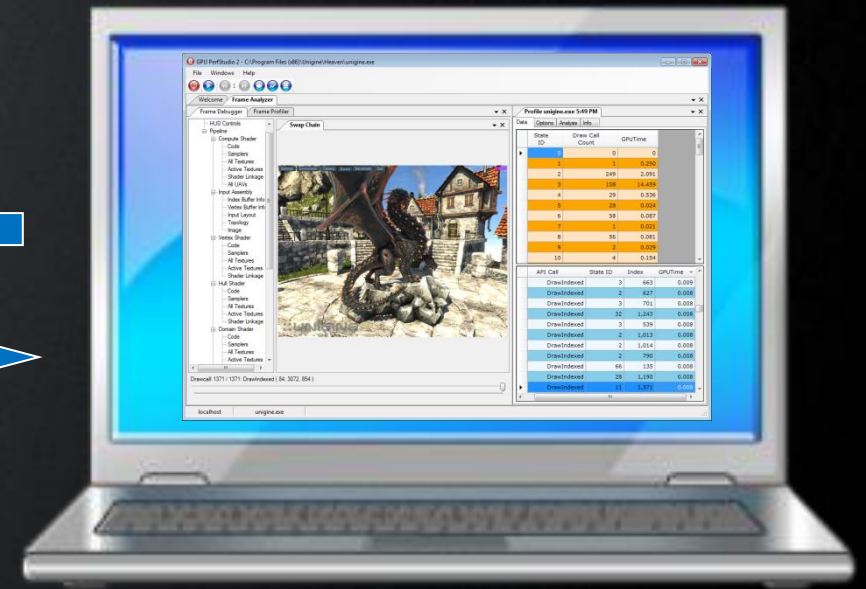
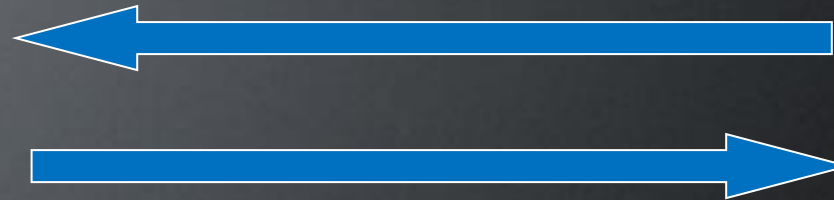


GPU PerfStudio 2.2 - How it works



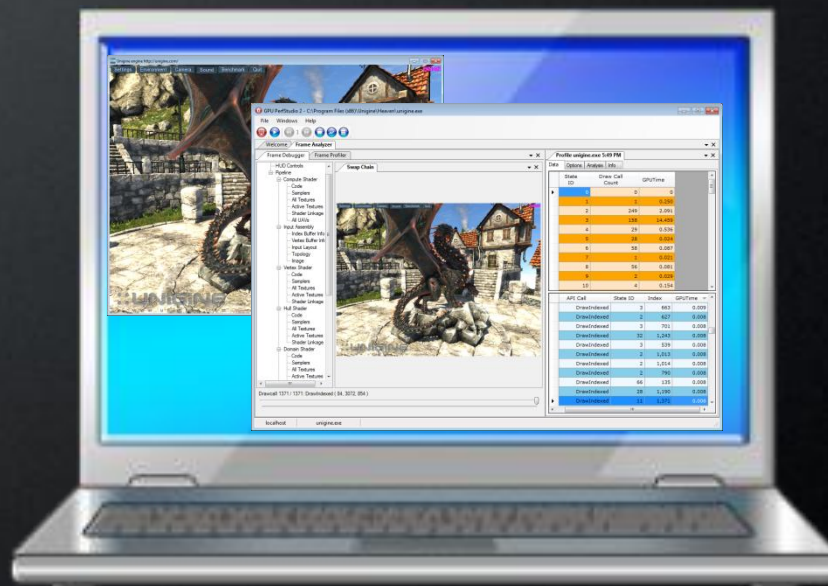
Server

Remote debugging



Client

Local debugging
server and client on
same computer



GPU PerfStudio 2.2 and Unigine Heaven

The image shows a screenshot of the Unigine engine running the Heaven benchmark, with the GPU PerfStudio 2.2 performance analysis tool overlaid. The Unigine window displays a 3D scene with a dragon and a stone building. The GPU PerfStudio window shows the 'Frame Analyzer' tab, which includes a 'Swap Chain' view and a 'Profile unigine.exe 5:49 PM' table.

GPU PerfStudio 2.2 - C:\Program Files (x86)\Unigine\Heaven\unigine.exe

Frame Analyzer

Swap Chain

Profile unigine.exe 5:49 PM

State ID	Draw Call Count	GPUPTime
0	0	0
1	1	0.250
2	249	2.091
3	158	14.459
4	29	0.536
5	28	0.024
6	58	0.087
7	1	0.021
8	56	0.081
9	2	0.029
10	4	0.154

API Call	State ID	Index	GPUPTime
DrawIndexed	3	663	0.009
DrawIndexed	2	627	0.008
DrawIndexed	3	701	0.008
DrawIndexed	32	1,243	0.008
DrawIndexed	3	539	0.008
DrawIndexed	2	1,013	0.008
DrawIndexed	2	1,014	0.008
DrawIndexed	2	790	0.008
DrawIndexed	66	135	0.008
DrawIndexed	28	1,190	0.008
DrawIndexed	11	1,371	0.008

Drawcall 1371 / 1371: DrawIndexed (84, 3072, 854)

localhost | unigine.exe

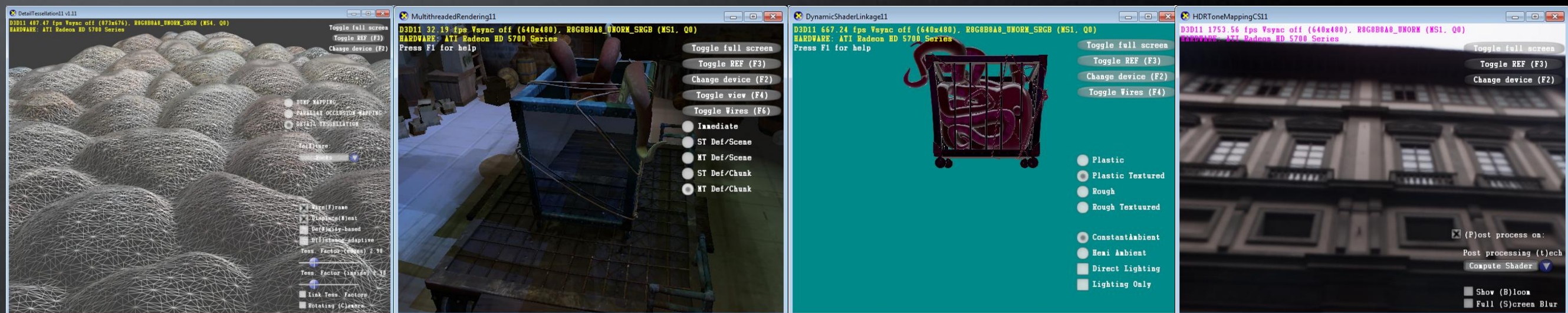


GPU PerfStudio 2.2 - Demo



GPU PerfStudio 2.2 –DirectX®11

- Compute Shader debugger
- Tessellation - view HS, DS shader code
- View UAV descriptions (OM and CS)
- View Dynamic Shader Linkages
- Multi-threaded DirectX® 11 application support



GPU PerfStudio 2.2 – OpenGL

The screenshot displays the GPU PerfStudio 2.2 interface, which is used for profiling and debugging GPU applications. The main window is titled "GPU PerfStudio 2 - C:\Program Files (x86)\Unigine\Heaven\unigine.exe".

Frame Debugger: The left sidebar shows the "Frame Debugger" with a tree view of the rendering pipeline, including Input Assembly, Render States, Index Buffer, Vertex Buffer, Image, Vertex Shader, Code, Samplers, Active Textures, Render States, Pixel Shader, Code, Samplers, Active Textures, Render States, Frame Buffer, Render States, Render Targets, Stencil Buffer, and Depth Buffer.

PS Code: The "PS Code" panel shows the shader code for the selected frame. The code is written in GLSL and includes functions for calculating light direction, camera direction, and attenuation. The code is as follows:

```
vec3 getDiffuseSpecularFresnelNormalizeAttenuate(vec3 diffuse, vec4 specular, vec3 light_dir, vec3 camera_dir)
{
    float light_dir_length2 = dot(light_dir.xyz, light_dir.xyz);
    float light_dir_length = inversesqrt(light_dir_length2);
    float camera_dir_length = inversesqrt(dot(camera_dir, camera_dir));
    float fresnel = clamp(s_material_fresnel.x + pow(clamp(1.0f - abs(dot(camera_dir, light_dir)), 0.5f), 2.0f), 0.0f, 1.0f);

    float attenuation = clamp(light_dir.w - light_dir_length2, 0.0f, 1.0f);
    attenuation *= attenuation;

    return diffuse * (clamp(dot(light_dir.xyz, normal) * light_dir_length, 0.0f, 1.0) * specular.xyz * (pow(clamp(dot(reflect(-light_dir.xyz, normal), camera_dir), 0.5f), 2.0f), 0.0f, 1.0f)) + (1.0f - attenuation) * diffuse;
}

vec3 getDiffuseRimSpecularNormalizeAttenuate(vec3 diffuse, vec4 specular, vec3 light_dir, vec3 camera_dir)
{
    float light_dir_length2 = dot(light_dir.xyz, light_dir.xyz);
    float light_dir_length = inversesqrt(light_dir_length2);
    float camera_dir_length = inversesqrt(dot(camera_dir, camera_dir));
    float fresnel = clamp(s_material_fresnel.x + pow(clamp(1.0f - abs(dot(camera_dir, light_dir)), 0.5f), 2.0f), 0.0f, 1.0f);

    float attenuation = clamp(light_dir.w - light_dir_length2, 0.0f, 1.0f);
    attenuation *= attenuation;

    return diffuse * (clamp(dot(light_dir.xyz, normal) * light_dir_length, 0.0f, 1.0) * specular.xyz * (pow(clamp(dot(reflect(-light_dir.xyz, normal), camera_dir), 0.5f), 2.0f), 0.0f, 1.0f)) + (1.0f - attenuation) * diffuse;
}
```

Draw Call Table: The "Data" panel shows a table of draw calls. The table has columns for State ID, Draw Call Count, and GPU Time. The data is as follows:

State ID	Draw Call Count	GPU Time
0	0	0
1	14	0.007
2	1	0.064
3	1	0.019
4	1	0.020
5	5	0.079
6	3	0.004
7	138	11.146
8	24	14.207
9	1	0.005
10	3	0.029
11	6	0.000

API Call Table: The "API Call" panel shows a table of API calls. The table has columns for API Call, State ID, Index, and GPU Time. The data is as follows:

API Call	State ID	Index	GPU Time
glDrawRangeElements	61	1,293	
glDrawElementsInstancedARB	32	1,053	
glDrawElementsInstancedARB	30	1,047	
glDrawElementsInstancedARB	50	1,232	
glDrawElementsInstancedARB	79	618	
glDrawElementsInstancedARB	50	1,027	
glDrawElementsInstancedARB	46	1,187	
glDrawElementsInstancedARB	79	1,011	
glDrawElementsInstancedARB	5	168	
glDrawElementsInstancedARB	32	1,050	
glDrawRangeElements	60	1,295	
glDrawElementsInstancedARB	42	1,088	

Render States: The "PS Render States" panel shows the current render states. The states are as follows:

- RenderStates
- GL_CURRENT_PROGRAM: 706
- Polygon
- GL_POLYGON_MODE
- GL_FRONT: GL_FILL
- GL_BACK: GL_FILL
- GL_POINT_SIZE: 1.000000
- GL_POINT_SIZE_MIN: 0.000000
- GL_POINT_SIZE_MAX: 8192.000000
- GL_POINT_SMOOTH: FALSE
- GL_POINT_FADE_THRESHOLD_SIZE: 1.000000
- GL_POINT_DISTANCE_ATTENUATION: 1.000000, 0.0000
- GL_POINT_SPRITE: FALSE
- GL_POINT_SPRITE_COORD_ORIGIN: GL_UPPER_LEFT
- GL_LINE_WIDTH: 1.000000
- GL_LINE_SMOOTH: FALSE
- GL_LINE_STIPPLE: FALSE
- GL_LINE_STIPPLE_PATTERN: 65535
- GL_LINE_STIPPLE_REPEAT: 1
- GL_POLYGON_SMOOTH: FALSE
- GL_POLYGON_STIPPLE: FALSE
- GL_POLYGON_OFFSET_POINT: FALSE
- GL_POLYGON_OFFSET_LINE: FALSE
- GL_POLYGON_OFFSET_FILL: FALSE
- GL_POLYGON_OFFSET_FACTOR: 0.000000
- GL_POLYGON_OFFSET_UNITS: -8.368523
- Lighting
- GL_SHADE_MODEL: GL_SMOOTH
- GL_LIGHTING: FALSE
- GL_COLOR_MATERIAL: FALSE

PS Active Textures: The "PS Active Textures" panel shows the active textures. The textures are as follows:

- FS[0]: Mips: 1, Size: 32 x 32, Samples: 1, Type: 3D, Usage: Unknown, Format: GL_RGBA8
- FS[1]: Mips: 7, Size: 64 x 8, Samples: 1, Type: 2D, Usage: Unknown, Format: GL_COMPRESSED_SRGB_ALPHA_S3TC_DXT5_EXT

OM Depth Buffer: The "OM Depth Buffer" panel shows the depth buffer. The depth buffer is a grayscale image of the scene.

OM RenderTarget 0: The "OM RenderTarget 0" panel shows the render target. The render target is a color image of the scene.

API Trace: The "API Trace" panel shows the API trace. The trace is a list of API calls and their parameters. The trace is as follows:

Tree	Index	Interface	Call	Parameters
	9009	OpenGL_1.2	glDrawRangeElements	GL_TRIANGLES 0 216 324 GL_UNSIGNED_SHORT 0x00000000
	9010	OpenGL_3.0	glBindVertexArray	0
	9011	OpenGL_1.0	glDisable	GL_BLEND
	9012	OpenGL_1.4	glBlendFuncSeparate	GL_ONE GL_ZERO GL_ONE GL_ZERO
	9013	OpenGL_1.4	glBlendFuncSeparate	GL_SRC_ALPHA GL_ONE GL_ONE GL_ONE
	9014	OpenGL_1.0	glEnable	GL_BLEND
	9015	OpenGL_2.0	glUseProgram	706
	9016	OpenGL_2.0	glUniformMatrix4fv	6 1 0 0x07963320
	9017	OpenGL_3.0	glBindVertexArray	37
	9018	OpenGL_1.5	glBindBuffer	GL_ARRAY_BUFFER 76
	9019	OpenGL_1.5	glBindBuffer	GL_ELEMENT_ARRAY_BUFFER 77
	9020	OpenGL_1.5	glBufferData	GL_ARRAY_BUFFER 3456 0x0ABDAFF0 GL_STREAM_DRAW
	9021	OpenGL_1.2	glDrawRangeElements	GL_TRIANGLES 0 216 324 GL_UNSIGNED_SHORT 0x00000000
	9022	OpenGL_3.0	glBindVertexArray	0
	9023	OpenGL_1.0	glCullFace	GL_FRONT
	9024	OpenGL_1.0	glDisable	GL_BLEND



GPU PerfStudio 2.2 Availability

- Download GPU PerfStudio now from <http://developer.amd.com/gpu/PerfStudio/>
- PerfStudio 2.2 available end of March 2010
- Free to all developers



Other AMD GPU Tools

Available at: **<http://developer.amd.com/gpu>**

- GPU MeshMapper
- GPU ShaderAnalyzer
- The Compressor \ ATI Compress
- ATI Stream Profiler for OpenCL
- Stream Kernel Analyzer
- And more...

And CPU tools too - **<http://developer.amd.com/cpu>**

- AMD CodeAnalyst Performance Analyzer
- AMD Performance Libraries
- And more...



GPU PerfStudio 2

- Thank you
 - Unigine
 - ATI ISV partners
 - AMD GPU Developer Tools Team
 - You
- Download
 - <http://developer.amd.com/gpu/PerfStudio/>



Contact

- Email
 - gputools.support@AMD.com
- Developer forums
 - <http://forums.amd.com/devforum/>
- And we're hiring!
 - <http://www.amd.com/us/aboutamd/careers>

