ShaderPerf 2.0 Quick Tutorial

DU-01231-001_v01
May 2008

Table of Contents

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Chapter 1. Introduction

What is NVIDIA ShaderPerf?

NVIDIA ShaderPerf 2.0 allows you to determine how expensive a given vertex or pixel shader might be across a range of GPUs and drivers. This can be useful in many ways.

- **Modify-and-Profile Cycle.** While tuning a shader, you can get immediate feedback on the resulting cycle count and throughput to determine if the performance improvement scheme has succeeded. This rapid feedback cycle can cut the performance tuning time significantly.
- **Profile GPUs Virtually.** Because the target GPU doesn't need to be in the system, you can look at tuning and optimizing a version of your effects for a consumer level GPU with your development system intact. No more replacing graphics cards and drivers just to test out new optimizations.
- **Daily Regressions.** ShaderPerf can be used in conjunction with source code control to continually test shader source code against established budgets to make sure that no performance regressions are allowed to slip into the master source base.
- **Pixel Shader Differencing.** For GeForce 6 and 7 Series GPUs, ShaderPerf has the ability to detect differences between two pixel shaders. It accomplishes this by running the shaders through a simulator with multiple random input vectors to determine if the resulting output is the same. This is useful when trying new approaches to solving a problem since to see if there is a change in the results from the shader.

Using ShaderPerf

ShaderPerf comes with both a command line executable and an API. This means you can integrate the same performance tuning functionality into your own tools and use the data in a way that makes sense for your environment.

In addition, ShaderPerf is integrated into FX Composer, which is available at www.fxcomposer.com.

Chapter 2. Getting Started

Sample Command Lines

The command line interface for ShaderPerf offers a wide range of options, from basic analysis to complex simulation options. A complete list of supported options is below, but here are some sample command lines to get you started:

1. Standard GeForce 7 Series analysis of the stdPS function using the ps_3_0 profile:

```
NVShaderPerf –gpu G70 –type hlsl_ps –function stdPS –profile ps_3_0 phong.fx
```

2. GeForce 8 Series analysis (with profile ps_4_0) of the same function/file but adding the /O0 hlsl option for optimizations and defining a light count via a macro:

```
NVShaderPerf –gpu G80 –type hlsl_ps –function stdPS – profile ps_4_0 –hlslopt
"/O0" –defines "/DLIGHTCOUNT=3" phong.fx
```

Shader Differencing

Sometimes, changes you make to a shader to improve its performance may inadvertently introduce inaccuracies in the shader's results for certain inputs. To catch this case, ShaderPerf 2 offers "shader differencing".

When ShaderPerf does shader differencing, it tests both your original shader and your modified shader with a variety of random inputs and compares the results. This allows you to get a feel for whether the modified shader is still functionally equivalent to the original.

Please see the "Shader Differencing" section in the list of supported options to see various ways to adjust the shader differencing testing process to meet your needs.

Complete List of Supported Options

INFORMATION

-?	Print help message
-listgpus	List the available GPUs
-listdrivers	List the available drivers
-o/-output outputfilename	Location for results with outputfilename, default is stdout.
-e/-error errorfilename	Location for errors, default is stderr

BASIC ANALYSIS

-type filetype	Type of shader input, glsl_vp, glsl_fp, hlsl_vs, hlsl_ps, d3d_vs, d3d_ps, d3d_bin, cg_vp, cg_fp, cg_bin
-v/-verbose level	Verbosity level (0-1, default is 0)
-t/-technique techid	Technique in a given fx file to run performance on
-p/-pass passid	Pass in a given technique to run performance on

-f/-function funcname	Function name for Cg shaders (or instead of using -technique & -pass with -profile for HLSL shaders)
-g/-gpu gpuname	Specify GPU target
-d/-driver drivername	Specify driver target
-profile targetprofile	Target profile for the -f/-function (vs/ps_1_1, vs/ps_1_2, vs/ps_1_3, vs/ps_1_4, vs/ps_2_0, vs/ps_2_a, vs/ps_2_sw, vs/ps_3_0, vs/ps_4_0)
-minprec	Compile shader with all fp16 registers and operations, then run performance, mutually exclusive with maxprec
-maxprec	Compile shader with all fp32 registers and operations, then run performance, mutually exclusive with minprec

ADVANCED ANALYSIS

-z/-zreplace	Enable depth replace (default is off)
-m/-mrt count	Enable MRT programs and set the MRT surface count
-color16	Set destination color register to 16bit bpp instead of 32bpp
-minbranch	Default computes the max branch cost (ie cost of longest path), this flag changes to compute shortest path cost]
-texdim hexbitfield	Set texture dimensions. 2 bits per texture, 0: unused, 1: 2D, 2: 3D, 3: Cubemap, default is 0x55555555 or all 2D
-texwidth hexbitfield	Set texture width info. 2 bits per texture, 0: Texture <= 8 bits per channel 1: semi-fat (ex FLOAT_RGBA16) texture 2: fat texture (ex FLOAT_RGBA32)]
-texsplit hexbitfield	[Set texture split info, 0 = unsplit, 1 =

	split, also need to set -texwidth]
-compile filename	Save results from high level compile as filename
-c/-cgcopt optionstring	Specify an option string to be sent to the Cg compiler, use quotes if multiple options or options with arguments (both requiring spaces) are needed
-hlslopt optionstring	Specify an option string to be sent to the D3D HLSL compiler, use quotes if multiple options or options with arguments (both requiring spaces) are needed
-d3dx version	Specify the version of the D3DX dll to use, defaults to highest available
-cgcpath path	Specify a path to the Cg compiler, must be full path, including cgc.exe
-asm	Print assembly language result from compiling Cg/HLSL/GLSL
-vpcond hexbitfield	Preset conditionals for vertex programs (1 = take branch, 0 = don't), first->last conditionals LSB->MSB
-include includepath	Specify an include path for higher level languages, quote the entire string, paths are semicolon delimited
-defines string	Specify preprocessor commands, use a quoted string in the format /D(id)=(text), ex: -defines "/DLIGHTS=5 /DSHADER=PHONG /DNEWVERSION"

SHADER DIFFERENCING

-fpdiff	Perform a functional diff between the first 2 shaders specified in the file list
-dtype filetype	File type for the difference file
-dfunction funcname	For the difference file, the function name to profile (instead of using technique/pass), must specify -[d]profile
-dtechnique techid	For the difference file, the technique in a given fx file to run performance on
-dpass passid	For the difference file, the pass in a given technique to run performance on
-dprofile targetprofile	For the difference file, the target profile for the -f/-function (vs/ps_1_1, vs/ps_1_2, vs/ps_1_3, vs/ps_1_4, vs/ps_2_0, vs/ps_2_a, vs/ps_2_sw, vs/ps_3_0, vs/ps_4_0)

-dcgcopt option	For the difference file, the option string to be sent to the Cg compiler, use quotes if multiple options or options with arguments (both requiring spaces) are needed
-dinclude includepath	For the difference file, the include path for higher level languages, quote the entire string, paths are semicolon delimited
-ddefines string	For the difference file, the preprocessor commands, use a quoted string in the format /D(id)=(text). Ex: -ddefines "/DLIGHTS=5 /DSHADER=PHONG /DNEWVERSION"
-dhlslopt option	For the difference file, specify an option string to be sent to the D3D HLSL compiler, use quotes if multiple options or options with arguments (both requiring spaces) are needed
-minrand value	Minimum float value for all random numbers, default 0.0
-maxrand value	Maximum float value for all random numbers, default 255.0
-errthresh value	Float value for the error threshold in differencing, default 0.1
-seed hexseedvalue	Random seed for differencing (hex value only)
-consts value	Range for constant randoms, 0 = [min, max] 1 = abs([min, max]) 2 = [-1, 1] 3 = [0, 1]
-attribs hexbitfield	Range for attribute randoms, Bitfield with 2 bits per attrib, 0 = [min, max] 1 = abs([min, max]) 2 = [-1, 1] 3 = [0, 1]
-textures hexbitfield	Range for textures, bitfield with 2 bits per texture, 0 = [min, max] 1 = abs([min, max]) 2 = [-1, 1] 3 = [0, 1]
-chvalid hexbitfield	Bitfield to enable testing for a color channel, RGBA MSB->LSB, default is 0x0F, ex. 0x0E will disable Alpha
-printdiff passindex	Index for a difference pass to print for detailed results

Chapter 3. The ShaderPerf API

The ShaderPerf API gives you the ability to add ShaderPerf functionality directly into your production pipeline and custom tools.

To use the API, link your program with NVShaderPerf.lib, and then follow the sample code shown below:

```
#include "NVShaderPerf.h"

// Somewhere in init
NVSPInit();

// When you want to run a program
SetValue(SVINDEX_SHADERTYPE, SVST_HLSL_PIXEL); // See the other values for this in .h file
SetValuePtr(SVINDEX_FUNCTIONNAME_PTR, "MyEntryPoint"); // Specify the entry point to compile
SetValuePtr(SVINDEX_DRIVERNAME_PTR, "174.74"); // Specify the driver to compile to
SetValuePtr(SVINDEX_GPUNAME_PTR, "G70"); // Specify the GPU to run performance for

// Run performance
FragmentProgramPerformance("filename", 0, &pResults, &nCount);
// Return code is NVSP_OK if all combinations compiled and ran
// (you can specify NV4* for the gpu and it will run perf on all NV4* GPUs),
// or the first error if there was one.

// Then, loop through nCount to look at all pResults
FreeFragmentResults(pResults);
```

You can also use the `EnumerateGPUs()` and `EnumerateDrivers()` functions to determine the list of GPUs and drivers that are currently installed.