AMD Accelerated Parallel Processing
TECHNOLOGY

SAMPLE

Uniform Random Noise Generator with GL Interoperability

1 Overview

1.1 Location

`$(AMDAPPSDKSAMPLESROOT)\samples\opencl\cl\app`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$(AMDAPPSDKSAMPLESROOT)\samples\opencl\bin\x86\` for 32-bit builds, and `$(AMDAPPSDKSAMPLESROOT)\samples\opencl\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `URNGNoiseGL`
   This generates uniform noise in the input image.
2. `URNGNoiseGL -h`
   This prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
-h	--help	Shows all command options and their respective meaning.
	--device	Devices on which the program is to be run. Acceptable values are cpu or gpu.
-q	--quiet	Quiet mode. Suppresses all text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing.
	--dump	Dump binary image for all devices.
	--load	Load binary image and execute on device.
	--flags	Specify compiler flags to build the kernel.
-p	--platformId	Select platformId to be used (0 to N-1, where N is the number of available platforms).
-d	--deviceId	Select deviceId to be used (0 to N-1, where N is the number of available devices).
-v	--version	AMD APP SDK version string.
-f	--factor	Noise factor.
-i	--iterations	Number of iterations for kernel execution.

2 Implementation Details

This sample generates noise in an image by using a *linear congruential generator* which generates a uniform deviation in the range (0, 1) and which is multiplied by a *noise factor* to produce the final noise.

A minimal standard linear congruential generator proposed by Park and Miller (see reference [1]) is:

$I_j+1 = a\ I_j \bmod m$

where $a = 16807\ (7^5)$ and $m = 2^{31} - 1$

We use Schrage's method (see [2]), which is based on an approximate factorization of m, to implement this.

m = aq + r, that is: q = [m/a], r = m mod a.

We then apply a shuffling algorithm by Bays and Durham, as described in Knuth (see reference [3]), to remove low-order serial correlations.

We calculate the uniform deviation from the seed, which is generated by averaging four components of a pixel, and apply that deviation (multiplied by the noise factor) to all the components of the pixel. Thus, each global thread computes a uniform deviation and applies it to a pixel.

3 References

1. Park, S.K., and Miller, K.W 1988, *Communications of the ACM*, vol 31, pp, 1192-1201.
2. Schrage, L. 1979, *ACM transactions on Mathematical Software*, vol. 5, pp. 132-138.
3. Knuth, D.E, 1981, Seminumerical Algorithms, 2nd ed., vol. 2 of *The art of computer programming*, 3.2-3.3.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport
Forum: developer.amd.com/openclforum