AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Perlin Noise

1 Overview

1.1 Location

`$<AMDAPPSDKSamplesInstallPath>\samples\bolt\`

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 `$<AMDAPPSDKSamplesInstallPath>\samples\bolt\bin\x86\` for 32-bit builds, and `$<AMDAPPSDKSamplesInstallPath>\samples\bolt\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `PerlinNoise`
   This command generates the cloud texture and writes to a 2D Image.
2. `PerlinNoise -h`
   This command prints the help file.
3. `PerlinNoise_TBB -h`
   This command generates a build with the multiCoreCpu path (the Thread Building Block library), enabled.

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 meanings.
	`--device`	Explicit device selection for BOLT.
`-q`	`--quiet`	Quiet mode. Suppresses all text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing-related statistics.
`-v`	`--version`	Bolt and AMD APP SDK version string.
`-i`	`--iterations`	Number of iterations.
`-r`	`--red`	Red component of the image [0 - 255].
`-g`	`--green`	Green component of the image [0 - 255].
`-b`	`--blue`	Blue component of the image [0 - 255].
`-z`	`--zoom`	Zoom-in/zoom-out.
	`-per`	Persistence: controls roughness of the image.
	`-octaves`	Number of iterations of coherent noise functions per pixel.

Note: The `--device multiCoreCpu` option becomes available when the sample is compiled with `ENABLE_TBB` defined. Microsoft Visual Studio build configurations `Debug_TBB` and `Release_TBB` are created for this purpose. These configurations have `ENABLE_TBB` defined to enable the TBB path (multiCoreCpu) for all the AMD BOLT functions used in the sample.

2 Introduction

Perlin noise is a computer-generated visual effect developed by Ken Perlin. It is a procedural texture primitive, a type of gradient noise used by visual effects artists to increase the appearance of realism in computer graphics. The function has a pseudo-random appearance, yet all of its visual details are the same size. This property allows it to be readily controllable; multiple scaled copies of Perlin noise can be inserted into mathematical expressions to create a great variety of procedural textures.

This sample generates noise similar to a cloud, which gets stored as a 2D Image. By varying the values of the RGB, persistence, and zoom components, various textures such as wood and grass can also be generated.

2.1 Perlin Noise Algorithm

Perlin Noise begins by creating a grid of vectors or gradients. Each grid point has a gradient pointing away from it in a random direction.

Now, for any given point (or pixel), interpolate the value from the surrounding four gradients. First, each grid-point has a value of 0, and every other point is found by interpolating between gradients instead of value-points. There are a few steps to this:

1. Get the four closest gradient values.
2. For each of the above, get the dot-product using the relative distance between it and the pixel.
3. Use a fade function to skew the interpolation value.
4. Use cosine-interpolate to find the value.

3 References

1. http://freespace.virgin.net/hugo.elias/models/m_perlin.html.
2. http://en.wikipedia.org/wiki/Perlin_noise.
3. http://mrl.nyu.edu/~perlin/doc/oscar.html.

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