15-468/668 Project 1: Fractals

Due Tuesday, 2/01/11 at 11:59pm

This project is meant to help you become familiar with CUDA by having you implement a fractal viewer on the GPU. For this assignment, we'll focus on the Mandelbrot set, which is very easy to compute per-pixel without having to worry about memory bandwidth or synchronization.

Requirements:

You must implement a Mandelbrot set viewer using CUDA, where each pixel will represent a point in the 2D complex plane. You may tune the aesthetics however you wish, as long as it clearly generates the image of a fractal.

Specifically, in order to write this code, you must fill in the `__global__ void kernel` function. This requires using each thread's unique ID to compute x and y coordinates. Then, you must iteratively determine how close this point is to the Mandelbrot set and write the resulting color to shared memory. One simple algorithm for plotting the Mandelbrot set is called the "escape time" algorithm, but you may use others as you see fit.

In addition to plotting the Mandelbrot set, your visualization must be dynamic in some way. Whether you interactively zoom/pan or vary your visualization with time is up to you. One method would be to create a timer with CUDA (see cutCreateTimer, cutStartTimer, and cutGetTimerValue), but you can also use Unix time facilities or GLUT's input callbacks.

Skeleton Code:

The skeleton code is written using GLUT and OpenGL to render the contents of the shared memory that CUDA has access to. The files `main.cc` and `window.cc` take care of this, so you don't have to modify them. The only file you are responsible for is `simple_kernel.cu`, which contains the kernel function for you to fill in. The vast majority of CUDA functions have already been laid out for you, so this is primarily to become familiar with writing thread-specific code using some of CUDA's facilities.

Setup:

The C++ runtime libraries and CUDA binaries and libraries aren't visible by default, so before you can use the makefile to build the project, you'll need to add some folders to your PATH and LD_LIBRARY_PATH.

If you are using the shell cshrc (check by running `echo $SHELL`), then add the following lines to your `.cshrc` file:

```
setenv PATH $PATH":/usr/local/cs/cuda/open64/bin"
setenv LD_LIBRARY_PATH
$LD_LIBRARY_PATH":/usr/local/lib64:/usr/local/cs/cuda/lib:/usr/local/c
s/cuda/lib64"
```

If you're using bash, then add the following lines to `.bashrc:`

```
export PATH=$PATH":/usr/local/cs/cuda/open64/bin"
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH":/usr/local/lib64:/usr/local/cs/cuda/
lib:/usr/local/cs/cuda/lib64"
```

Also, in case you use VIM and would like syntax highlighting (for the C++ portions of your code), add the following line to your `.vimrc`:

```
au BufNewFile,BufRead *.cu set ft=cpp
```

Hardware:

CUDA is proprietary and only works on NVIDIA hardware, so if you only have Intel or ATI graphics available to you, you'll have to work using one of the Gates machines in the 5th floor or 3000 clusters (to be specific, the ghcXX.ghc.andrew.cmu.edu machines, where XX is from 01 to 82). Makefiles for Mac OSX and Linux are also provided, so if you do have modern NVIDIA hardware, you can try using them with some minor adjustments.

Unfortunately, you won't be able to use VNC or X-forwarding to visualize this project remotely. The fast shared access between OpenGL and CUDA is accomplished using a Pixel Buffer Object, which is an extension in OpenGL 2.0 – GLX doesn't support extensions, so OpenGL won't be able to initialize over the X-tunnel.

Handin and grading:

By 11:59:59pm on Tuesday, February 1st (2/01/11), you must email a tarball of the code that builds on the Gates machines to the TA at **dboehle@andrew.cmu.edu** with the subject "15-468 Project 1 Handin".

Because this is the first assignment and doesn't have a lot of depth, it will be graded based mostly on completion of the basic requirements, as well as correct use of CUDA functions.

Bonus:

There will be a competition for having the prettiest Mandelbrot viewer, so try to make a smooth or interesting visualization! You may display related fractals of the Mandelbrot set, such as a Julia Set, as long as it behaves like a fractal.

Reference materials:

For writing code for CUDA threads, this overview will help serve as a refresher to the basic concepts, particularly pages 37 through 40.

http://www.nvidia.com/content/cudazone/download/Getting_Started_w_CUDA_Training_NVISION08.pdf

For convenience, the official CUDA documentation is currently being hosted on the course website at http://www.cs.cmu.edu/afs/cs/academic/class/15668-s11/www/cuda-doc/, where you can access both the PDF manuals and the html documentation. This has documentation for the version of CUDA installed on the Gates machines.

For conceptual and algorithmic understanding of plotting Mandelbrot sets, check these articles:

http://en.wikipedia.org/wiki/Mandelbrot_set#For_programmers

http://en.wikibooks.org/wiki/Fractals/Iterations_in_the_complex_plane/Mandelbrot_set