COMS W4995
Parallel Functional Programming
Project Proposal
dwo2102

The project proposal is to render the Mandelbrot set using the parallel facilities provided by the Haskell ecosystem. When working in the continuous domain, the 'Mandelbrot set is the set of complex numbers *c* for which the function $f_c(z) = z^2 + c$ does not diverge when iterated from $z = 0$' [1]. In other words, if the number of iterations *n* goes to infinity and $f_c(z)$ is not bounded, the point does not belong in the set. It has been proven that set inclusion implies that $|f_c(z)| \le 2$ ('Radius of escape') for all $n \ge 0$ [1].

To calculate the Mandelbrot set on a computer, i.e. in the discrete domain, the values of the complex numbers c are restricted to this discrete domain. It is the discrete domain of c and the fact that we are restricted to a finite number of iterations that distinguishes the discrete problem from its continuous counterpart.

This discrete domain of c makes calculating Mandelbrot set an ideal candidate for parallel processing. Each pixel placement in the image is the source for a complex number c. Using a function of the row and column numbers of these pixels, we can source *x, y* coordinates which map to the 2d representation of the complex numbers, where the *x* coordinate represents the real component of *c* and the *y* coordinate represents the imaginary component. The mapping of a pixel to the question of set inclusion is independent of all other pixels in the image. Therefore, since the number of pixels *n* is equal to the image row count multiplied by the column count, we have *n* possible parallel tasks. In his book 'Parallel and Concurrent Programming in Haskell' [2], Simon Marlow uses the power of the GPU to process this volume of tasks at one time. Since this project will be using the idea of parallel 'Strategies' in Haskell, processing this number of tasks simultaneously would cause more time in overhead than could be gained by the parallel processing itself, so an alternative strategy will be devised.

As is typical when rendering the Mandelbrot set, the project will show set inclusion points in black and gradiate the colors of points not in the set by how unstable they are, measured by how quickly they were excluded from the set. We determine how quickly a point is excluded from the set by using the iteration number that drove the point outside the circle of radius 2, i.e. the 'Radius of Escape'. When the final set is calculated an image will be created which follows these rules for each pixel. At this time, the choice of Haskell image library is an open question. Though the Haskell Image Processing (HIP) library, the JuicyPixels library and the Friday library would seem to be three of the most popular image libraries on Hackage, the Haskell package archive.

References

1. Wikipedia contributors. "Mandelbrot set." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 21 Nov. 2020. Web. 22 Nov. 2020.
2. Marlow, Simon. "'Parallel and Concurrent Programming in Haskell." O'Reilly Media, 2013, chapter 6.