Fractals GPU

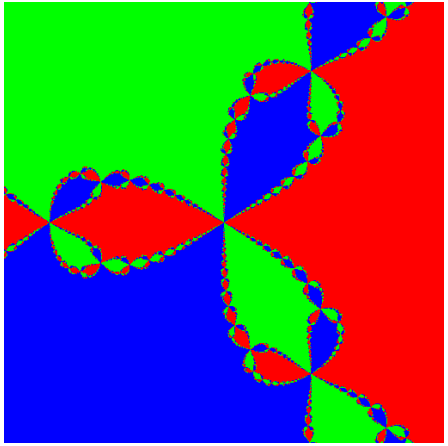
Contributed by:
Alvin Tan
Kenneth Toh
Yong Kiat
Cheng Jiang

Overview

To accelerate generation of Fractal patterns using GPGPU using CUDA C/C++ programming techniques.

Hénon Map - Alvin

Newton Fractal - Alvin



Thrust::complex or cuComplex

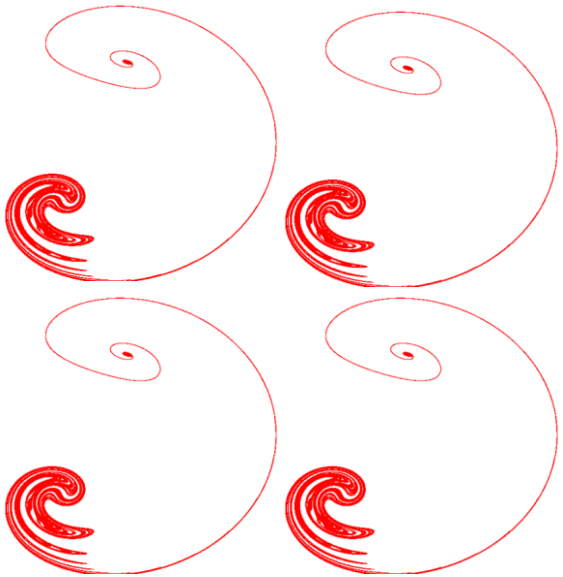
- Both are similar!
- Results only differs by **0 to 2%**

Ikeda Map - Alvin

Single Precision v.s. Double Precision for GPU

- How much faster?
    - Single Precision is faster by **15-16x**!

Ikeda Map - Alvin



All four version are different but same to the human eye!

Brownian Tree - Kenneth

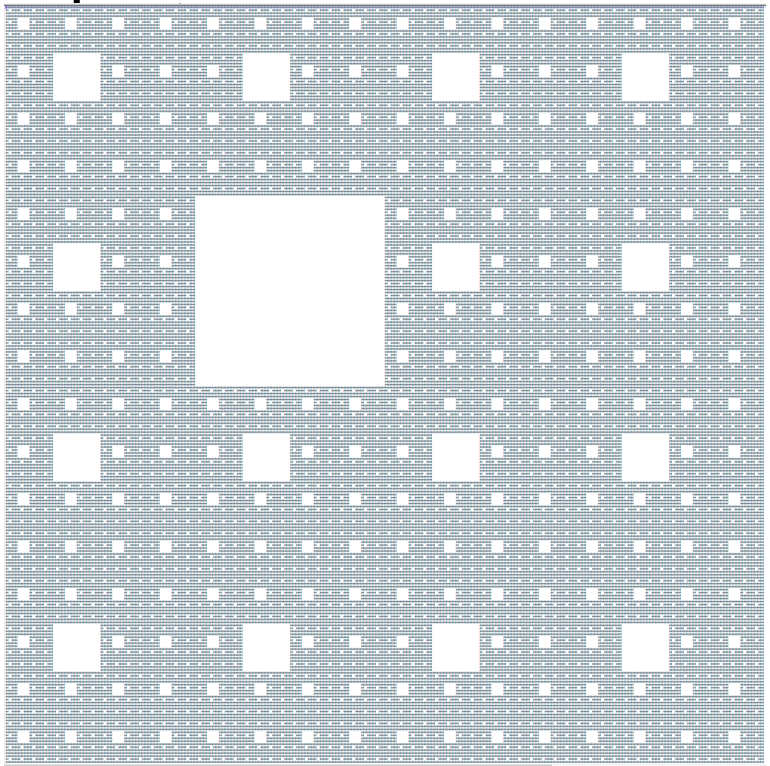
CUDA Conversion

- Failed due to unforeseen circumstances
- Eventually dropped the experiment

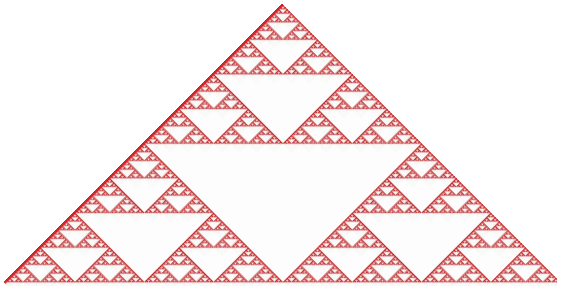
Sierpinski Carpet - Kenneth

Performance Increase

- Est. 1% ~ 1.3%
- Memory Management does help depending on the size of the fractal

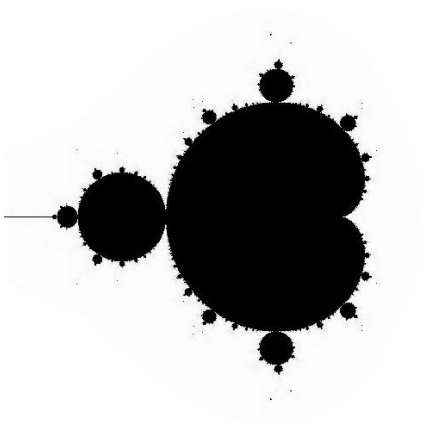


Sierpinski Triangle - Yong Kiat



Tried to parallelize but failed

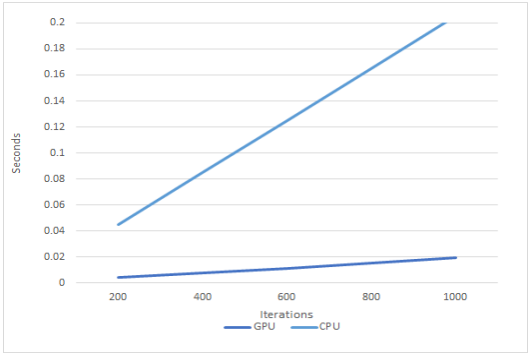
- Unable to speed up the computation of the algorithm using parallelism.



Mandelbrot Set - Yong Kiat

Accomplishment 1

- Managed to speed up the process and achieve the necessary speed up of 9.6%

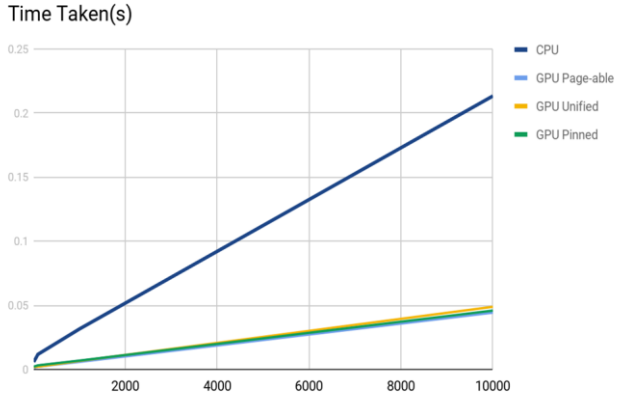


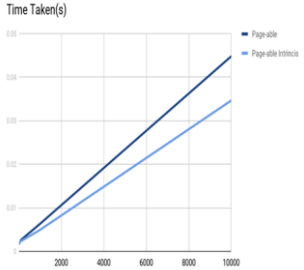
Burning Ship Fractals - Cheng Jiang

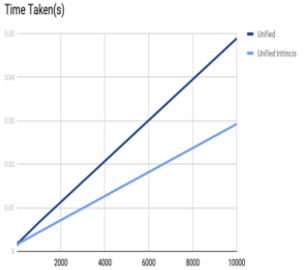
- **Optimizing**

Main focus would be using intrinsic to speed up the process of the generating the burning ship image.

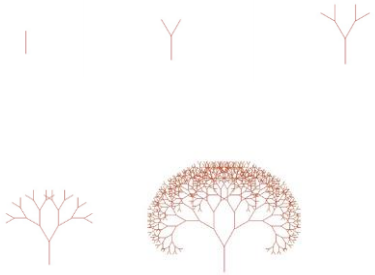
Additional, test of page-able, unified and pinned memory check on speed up(roughly 5 times)

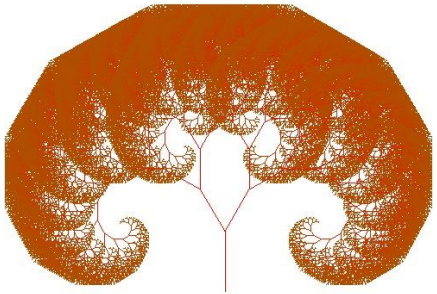






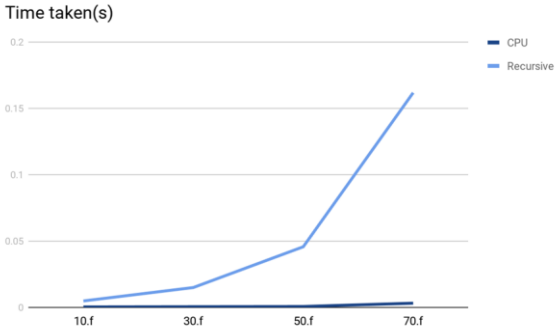
Fractal Tree - Cheng Jiang





Recursive

- Zero parallelism
- Unknown stack amount for a single thread
- Slow and inefficient(50 times slower)



Iterative

- Need a greater amount of memory to allocated compare to Recursive
- Breadth first compare to depth first of Recursive
- Efficient and capable of parallelism (1.6 times faster)