GPU Sorting Algorithms in OpenCL

- I coded the sorting algorithms mergesort, bitonic merge sort and a bubble sort sorting network for OpenCL- Also there is an implementation of merge sort on CPU for comparison. See Sort.cl for Kernel-Code, CSortTask.cpp for most of the host code and CSortingMain.cpp for changing local work size and array size. Generally GPU's are inefficient for sorting algorithms implementations but I wanted to try it and see for myself. For the most part there is no benefit in writing a sorting algorithm on a GPU because it simply cannot use the parallel features present in GPU computing. However Merge Sort does preform pretty decently is possibly one of the most efficient sorting algorithm for GPU's

Standard Mergesort for GPU:

- Mostly not recommended because it does not benefit that well from parallelization. With bigger array size the graphics card will eventually run out of memory. Much better handled on the CPU

Bubble Sort Sorting Network:

- Called Simple Sorting Network (SSN) in Code. Not recommended because it is inefficient and slow. Much better handled on the CPU.

Bitonic Merge Sort

- Recommended sorting variant of the three implemented algorithms. It Is very fast and utilizes parallelization very well.

How to Build

- Best way is to use cmake with the Code folder as source folder. Use a 64-Bit compiler (I'm pretty sure everyone does but just as a security or else big array sizes will not work.)
- > cmake CMakeLists.txt
- > make

Debug

- This application is designed to run on MacOS. It will not work on windows, and possibly will run into issues on Linux OS. The only issues that will occur on linux are header #include statements not aligning. I believe linux headers are “CL/cl.h”. If any compilation errors occur it should be resolved by this
- Cmake sometimes will say the directory does not match. This is easily resolved by deleting a specific file ”Code/CMakeCache.txt”

- All measurements with randomly generated arrays of the given size. Times are in milliseconds. The CPU variant for comparison is a self written mergesort implementation that generally runs faster than std::sort.

Size	CPU	Merge Sort		SSN		Bitonic Merge Sort	
.	Time	Time	Speedup	Time	Speedup	Time	Speedup
4 * 1024	0.312	2.364	0.133	0.997	0.315	0.435	0.722
64 * 1024	6.905	36.998	0.188	48.629	0.143	0.873	7.711
256 * 1024	28.497	177.688	0.161	703.29	0.042	2.019	14.123
2 * 1024 *1024	253.283	1450.661	0.176	-	-	24.681	10.263
64 *1024 *1024	8861.331	-	-	-	-	1192.821	7.439

Resources:
https://cs.wmich.edu/gupta/teaching/cs5260/5260Sp15web/lectureNotes/parallel%20sorting%20from%20Fernando%20Silva.pdf