Ice Floe Simulator

Steven Chaulk, Shadi Alawneh, Dennis Peters, Haochen Zhang, Claude Daley, and Gary Blades

Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John’s, NL, Canada
{steven.chaulk, shadi.alawneh, dpeters, hz3030, cdaley, gblades}@mun.ca

Abstract

The ice floe simulator, built as part of the StePS project, models the movement of floating ice floes as they interact with each other and around impeding structures. The application utilizes General Purpose Graphical Processing Unit (GPGPU) computing using Nvidia’s CUDA libraries to take advantage of the high number of cores in CUDA enabled GPUs and to compute the movement of ice floes in hyper-real-time. This poster and demo illustrates current work in this project and highlights what is being done to increase both the speed and size of ice field simulations.

1. Hyper-Real-Time Simulation

The Ice Simulation Viewer is an attempt to provide accurate simulation results for realistic ice fields. In order to be effective at providing useful information quickly enough to be acted on, these simulations need to be computed in hyper-real-time. To accomplish this, the Ice Floe Simulator has taken advantage of General Purpose Graphical Processing Unit (GPGPU) computing to utilize the large number of processing cores on current Graphics Processing Unit (GPU) devices to perform large amounts of calculations in parallel. The Ice Floe Simulator uses NVIDIA’s CUDA libraries to get optimum performance using CUDA enabled GPUs.

2. Simulator Structure

The Ice Floe Simulator has been built using the Qt framework in order to take advantage of its various libraries and cross-platform capabilities. The simulator has an interface that a user can interact with in order to load in files, load simulations, load images, edit properties, or create simulations. These functions are completed by separate processes which use inter process communication via TCP sockets to communicate with the simulator interface. The entire Ice Flow Simulator is composed of three distinct processes in a client server model that communicate in this way and can be labeled as the Simulato, Server, and Engine. Fig. 1 shows the three processes and how they relate to each other in the Ice Floe Simulator.

3. Features

The Ice Simulation Viewer can be used to play and generate 2D simulations of ice pan movement and interaction with structures. These simulations take a variety of variable parameters, such as the water drag coefficient, wind direction and force, and simulation time step that the user can specify for their own simulation. Each simulation can be exported to an xml based .ice file that contains all the information for the simulation, and which can be imported back into the simulator to be played again. Fig. 3 shows a sample of an .ice file. Each object tag represents one ice floe, and contains its location, velocity, and thickness.

4. Ice Floe Breaking

A new mechanic that has been implemented into the physical model is the breaking of any ice floe into two separate ice floes. This event is caused by the application of force to the ice floe, such as a moving structure, and is triggered when the force applied is large enough. This works by creating a new ice floe within the shape of the original, and reshaping the original ice floe to fill the remaining area. One ice floe receives a new index while the other will retain the index of the original ice floe. See Fig. 6 for an example.

5. Multi Platform Use

Currently the Ice Floe Simulator is being expanded to work on different platforms using varying CUDA enabled GPUs and operating systems. Supported operating systems for the Ice Simulation Viewer include Windows 7 and OSX.

6. Results

We have used several systems to successfully run the simulator including:
1. Intel(R) Xeon(R) CPU @2.27GHz (2 processors) and a GPU Tesla C2050. This card has 448 processor cores, 1.15 GHz processor core clock and 144 GB/sec memory bandwidth.
2. Intel(R) Core(TM) i5-2500k CPU @3.30GHz (4 processors) with a NVIDIA GTX 580 GPU that has 512 processor cores, 1.54 GHz processor core clock and 192.4 GB/sec memory bandwidth.
3. Intel(R) with a NVIDIA GTX 650M GPU that has 384 processing cores and 80 GB/sec memory bandwidth.

7. Future Work

Investigation is being done into utilizing multiple GPUs to perform simulation calculations [2]. Also, minor adjustments need to be made in order to make the user interface more intuitive and less redundant.

8. ACKNOWLEDGMENTS

This research has been done under StePS project and was supported by: ABS, Atlantic Canada Opportunities Agency, BMT Fleet Technology, Husky Oil Operations Ltd, Research and Development Council, Newfoundland and Labrador and Samsung Heavy Industries.

References