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^2 project[1]*, models the movement of floating ice floes as they interact with each other and around impeding structures. The application utilizes General Purpose Graphical Processor Unit (GPGPU) computing using Nvidias 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 Sructure

The Ice Simulation Viewer 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 files, load simulations, load images, edit properties, or create simulations. These functions are completed by seperate processes which use inter process communication via TCP sockets to comunicate with the simulator interface. The entire Ice Flow Simulator is composed of three distinct processes in a client server model that comminicate in this way and can be labeled as the Simulator, Server, and Engine. Fig. 1 show the three processes and how they relate to each other in the Ice Floe Simulator.

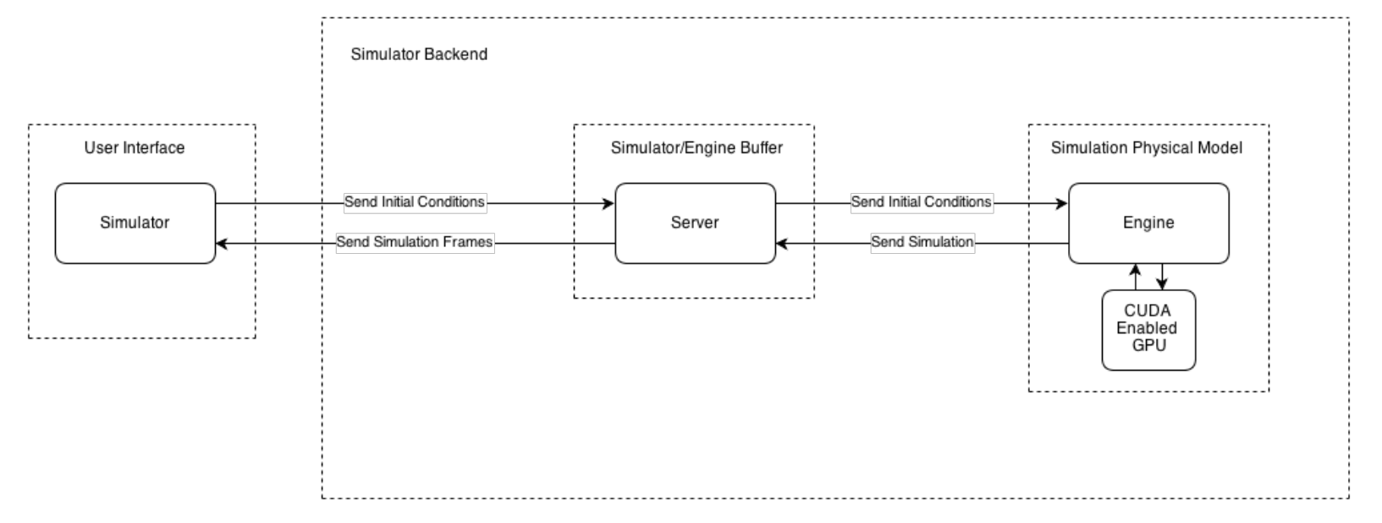


Figure 1: *Internal Structure of Simulator*

The Engine process [1] in the Ice Simulation Viewer is solely responsible for creating the simulation based on the parameters it recieves from the Simulator. To do this, the engine uses the NVIDIA developed CUDA libraries to use GPGPU computing on a CUDA enabled GPU. These GPU's contain hundreds of CUDA cores with the upper range models containg as much has a couple thousand. Fig. 2 shows the internal flow of the Engine process and how it relates to the GPU device.

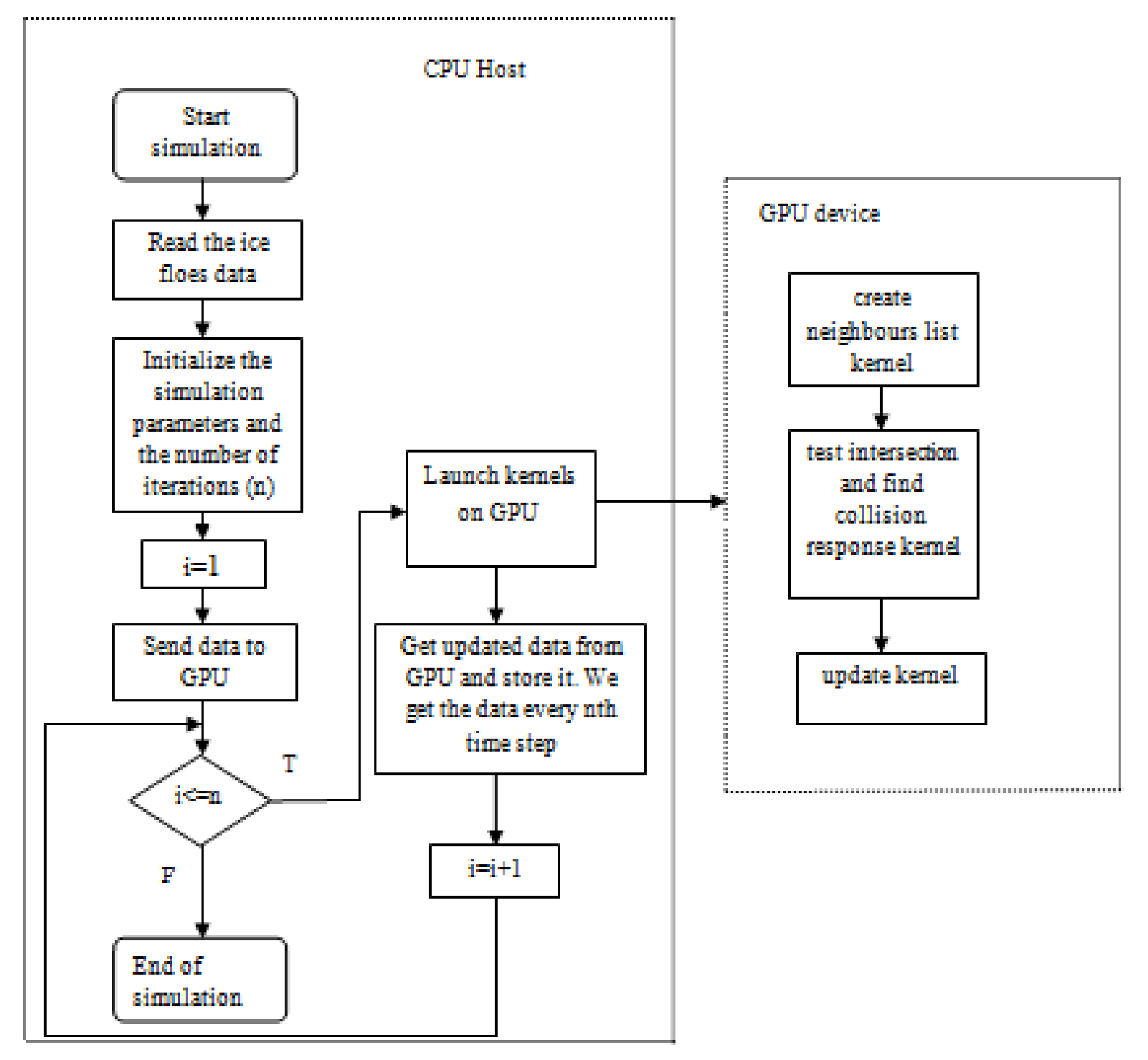
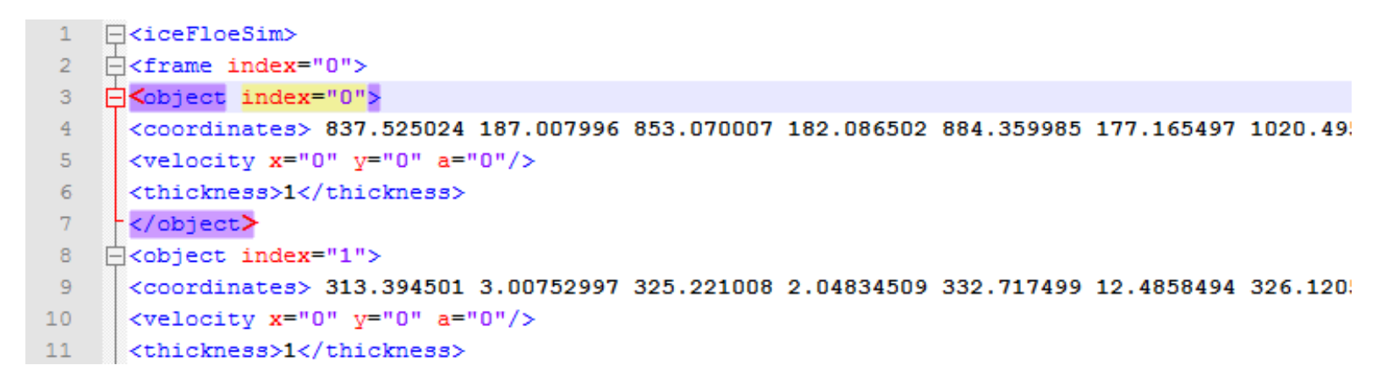

CPU Host
Start simulation
Read the ice floes data
Initialize the simulation parameters and the number of iterations (n)
i=1
Send data to GPU
i<=n
T
F
End of simulation
Launch kernels on GPU
Get updated data from GPU and store it. We get the data every nth time step
i=i+1
GPU device
create neighbours list kernel
test intersection and find collision response kernel
update kernel

Figure 2: *Engine Flow Chart[1]*

The Server process acts as a buffer for the communication that is being sent from the Engine process and the Simulator process. This means that the Simulator can play a partial simulation from the engine while stile continuing to load more of the simulation from the Engine. The Server process is also responsible for importing files.

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.



```
<iceFloeSim>
<frame index="0">
<object index="0">
<coordinates> 837.525024 187.007996 853.070007 182.086502 884.359985 177.165497 1020.49
<velocity x="0" y="0" a="0"/>
<thickness>1</thickness>
</object>
<object index="1">
<coordinates> 313.394501 3.00752997 325.221008 2.04834509 332.717499 12.4858494 326.120
<velocity x="0" y="0" a="0"/>
<thickness>1</thickness>
```

Figure 3: *Sample ice File Code*

The play back of a simulation can be paused and played, and the view of the ice field rotated or panned. Ice floes can be selected from the ice field and the floe properties are displayed for the user to see. Fig. 4 shows the ship being selected in the middle of a paused simulation; its properties are displayed in the top left of the screen.

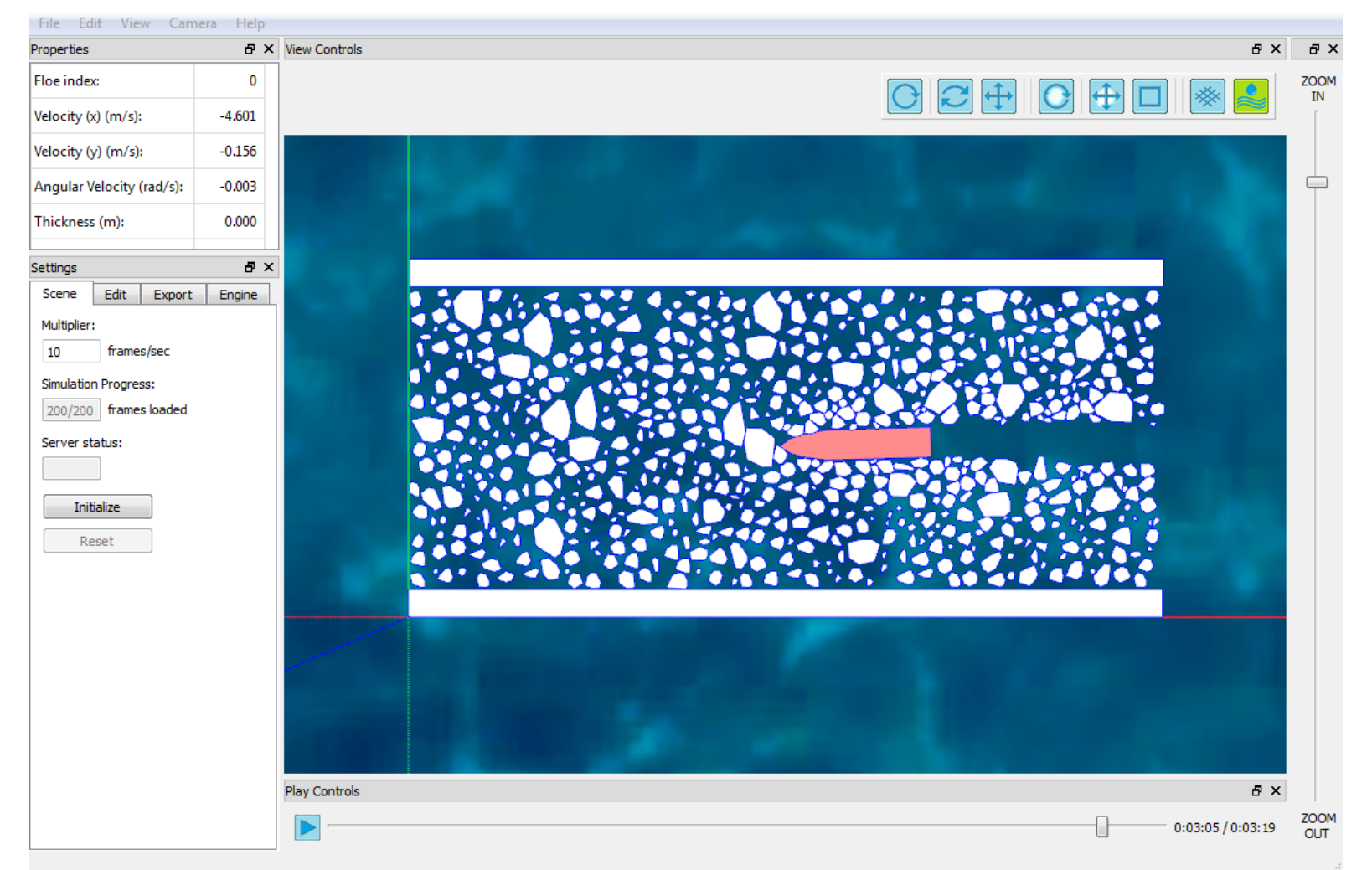


Figure 4: *Running an Ice Floe Simulation*

4. Ice Floe Breaking

A new mechanic that has been implemented into the physical model is the breaking of any ice floe into two seperate ice floes. This event is cause 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 form 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

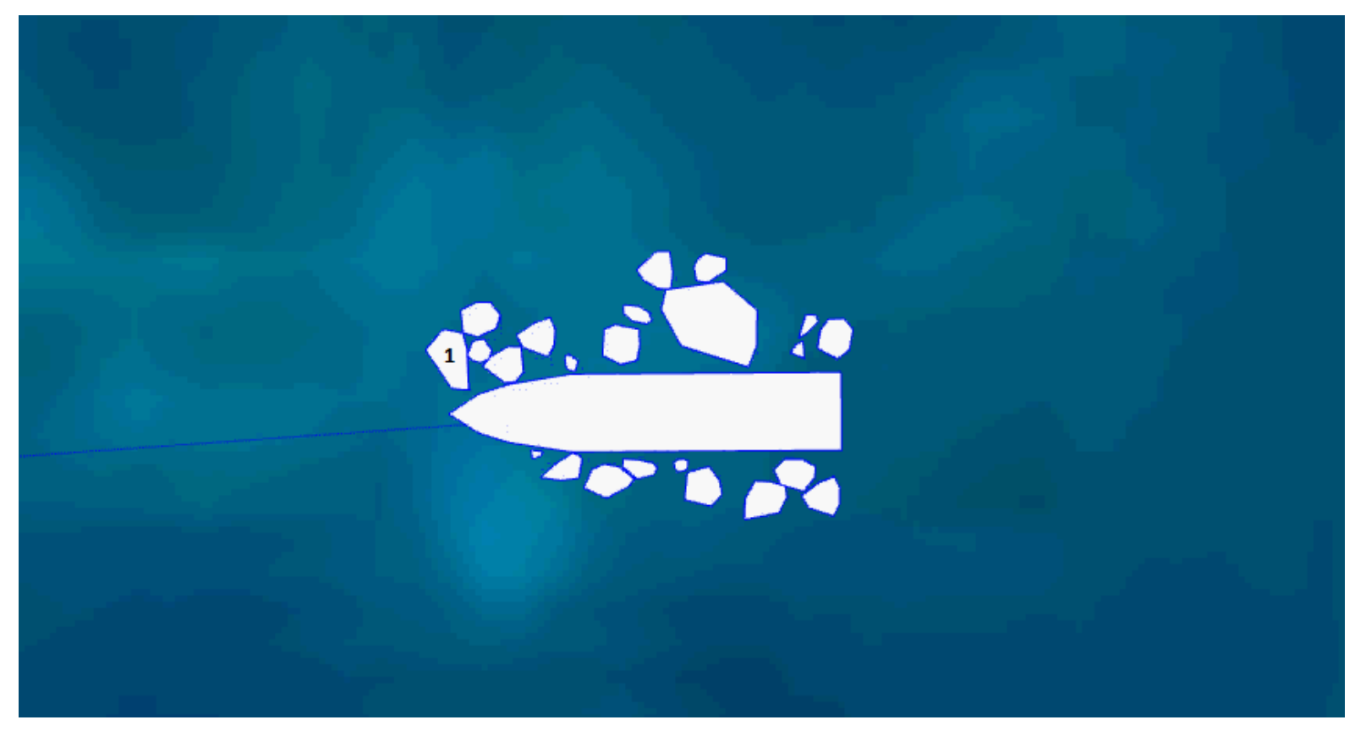


Figure 5: *Ice Floes Before Breaking: 22*

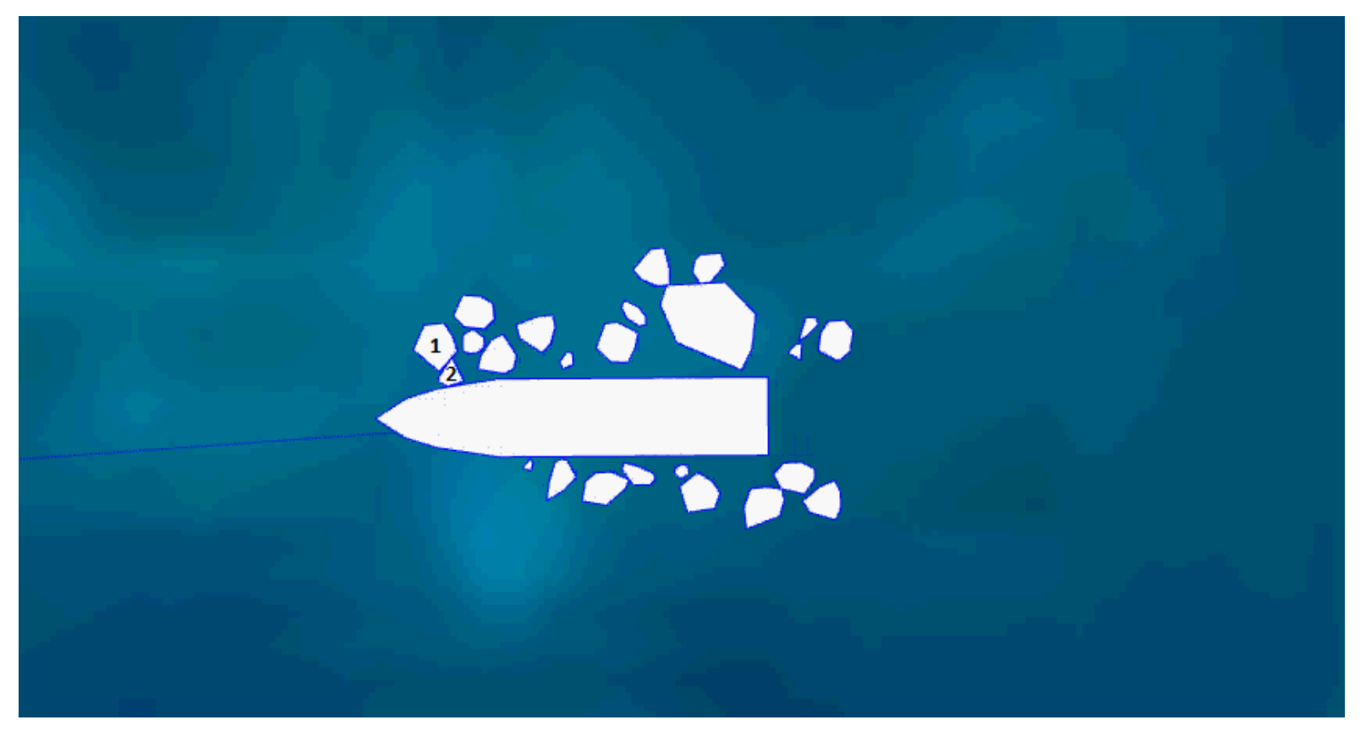


Figure 6: *Ice Floes After Breaking: 24*

Figure 6 above shows the ice field after a structure has moved through it with sufficient force to cause an ice floe to break apart.

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 multple GPU's to perform simulation calculations [2]. Also, minor adjustments need to be made in order to make the user interface more intuative and less redundant.

8. ACKNOWLEDGMENTS

This research has been done under STePS^2 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

[1] Shadi Alawneh and Dennis Peters, "Ice simulation using gpgpu," in *Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, Washington, DC, USA, 2012, HPCC '12, pp. 425–431, IEEE Computer Society.

[2] Xiaoqian Men and Dennis Peters, "Particle simulation using serial,gpu and distributed approaches," in *In Proceedings of Newfoundland Electrical and Computer Engineering Conference (NECEC 2013), IEEE*, St. John's, NL, Canada, Nov 2013.

[1] http://www.engr.mun.ca/research/Steps2.php