REAL-TIME VISUALIZATION OF SHIP AND WAVY SURFACE MOTIONS BASED ON GPGPU COMPUTATIONS

Anton Gavrikov[a], Andrei Ivashchenko, Ivan Gankevich, Nataliia Kulabukhova, Alexander Bogdanov, Alexander Degtyarev, Vladimir Rukovchuk

Saint Petersburg State University

E-mail: [a] gavrikovantonkapi@gmail.com

One of the key stages in ship design process is the modeling of its behavior on the wavy sea surface, carried out with the expected operational characteristics taken into account. Similar modeling process could be done within real conditions at Virtual testbed, which allows to monitor the influence of external disturbances on ship's running characteristics in real time with sensors installed on-board. Visualization of the results for such modeling process allows the researcher to correctly and holistically perceive occurring events, as well as to predict and timely respond to emerging dangerous situations. If we are using GPGPU technology for computation purposes, results of modeling will be already placed in GPU memory after process completion. This fact can be regarded as an opportunity to optimize the visualization process by converting the raw simulation data into graphic objects directly on the GPU, and interaction mechanisms between OpenGL and OpenCL could be used here. In this article we demonstrate the effectiveness of this technique on the example of ship behavior visualization on a wavy sea surface, as well as forces acting on the ship's hull.

Keywords: GPGPU, OpenCL, OpenGL, visualization, ocean waves, virtual testbed

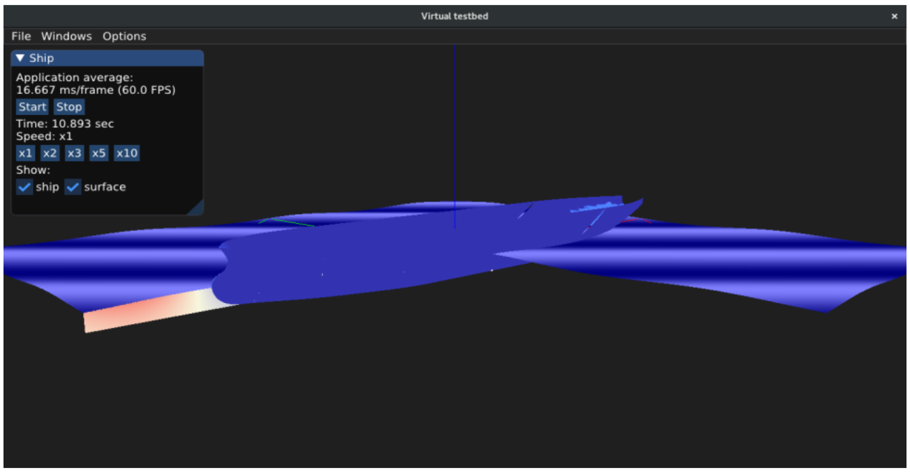
1. Introduction

Our problem includes a lot of different calculations, but CPU performs poorly in many of the algorithms, that we have used. Also we have done visualization based on our results. So our goal is to use GPU in such part of our program to speed up it and unite visualization with some computations. The same approach is used in [1], [2]. In the first article, the authors develop a system for panorama image stitching in real time with OpenCL. They use an OpenCL for fast computing parallel tasks and OpenCL/OpenGL sharing to avoid data copying, they compare their results with a coherent version on the CPU. In the second paper Ukidave et. all describe ways to optimize the OpenCL/OpenGL interoperability and presents Slot Rendering Mechanism for better performance. Various tests are conducted with and without OpenGL sharing and with different parameters their algorithm. In both cases, acceleration is obtained when using the OpenGL sharing, so we also decided to use it.

2. Virtual testbed

Virtual testbed is a program for modeling wavy surface and its influence on ship hull. First, we compute wavy surface, after that we determine velocity potential field and pressure field. All of this we compute using CPU. This part of our program is already fast enough, so we do not need to use GPU. Second, we find translational and angular motion of our ship using previous data. For that purpose we use GPU. Finally, we visualize all of this with Magnum, which is based on OpenGL. We achieve two goal with such approach. The first one is optimization of slow ship motion calculation on GPU using OpenCL. The second one is calculation and visualization of ship motion on GPU, that provide an opportunity to store this data in GPU memory and do not use inefficient data copying between CPU and GPU memory.

Projects with similar functionality already exist, but our Virtual testbed have some advantages. Fast algorithm for wave computing and using GPU allow us to do all of this in real-time. Also graphical user interface and visualization help to change a lot of options for fast and convenient modeling of different situation. Also we use IGES format for creating ship hull. It is format for engineer, which stores data in analytic format.

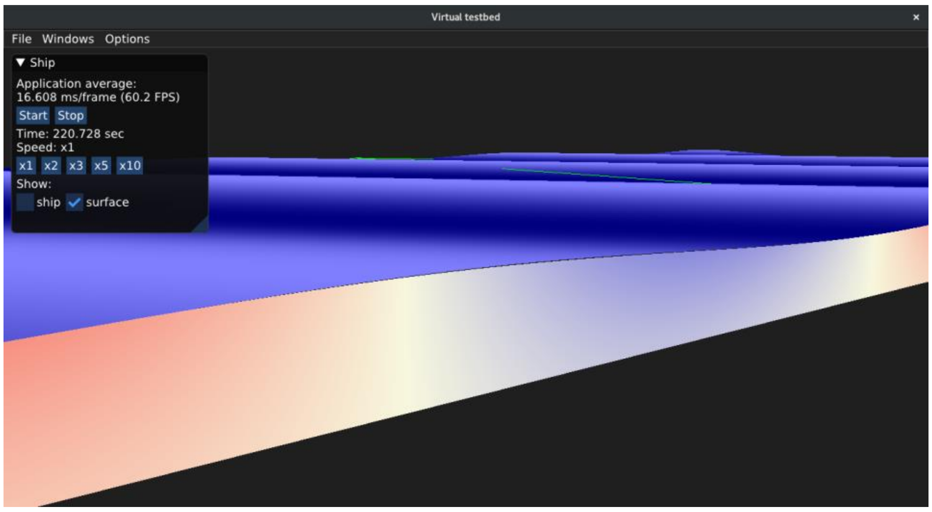


Application average:
16.667 ms/frame (60.0 FPS)
Time: 10.893 sec
Speed: x1

Figure 1 . Ship motion visualization

3. Visualization

The picture demonstrates the surface of the water that changes over time. Specifically, this slide shows a wavy surface a three hundred and a three hundred meters in size with an amplitude of one meter. To represent the velocity potential, we use the slice of our surface at the required point. On these panels we show the potential value using colors. The red color for values greater than zero, blue for values less than zero and white at zero. Velocity potential field and pressure field look alike, so it is enough to show only one of them. Also there is ship, created from IGES format. There were some difficulties with the hull normals. Because the format analytically describes the ship, it was necessary to calculate the normals additionally. It takes about 20 seconds, so after the first calculation we write it in the cache, so as not to repeat it every time. Moreover, GUI allow configure settings of modeling such as step, type of wave, it's amplitude and much more.



Application average:
16.608 ms/frame (60.2 FPS)
Time: 220.728 sec
Speed: x1
Show:
☐ ship ✓ surface

Figure 2 . Velocity potential field visualization

4. Determination of wetted panels

Our function have to determine which panels is under water. Since there are quite a lot panels, we use OpenMP for accelerating computation. Also for all panels under water we have to find their normals, area, volume and center. All of this operations easily can be transferred to GPU. In OpenMP version we have breakpoints for panels above water and for panels with zero area. Actually, we were surprised when he have discovered panels with zero area, it was caused by data format and we ignore these panels. Finally, real ship hulls have hundreds of thousands of panels, thus we expected significant acceleration on GPU.

5. Evaluation

For testing results of using OpenCL with OpenGL sharing we have measured time of computation of three version of our function: OpenMP, OpenCL and OpenCL with OpenGL sharing. As parameter we have used no. of panels, which was 1280, 5120, 20480 and 81920. For our tests we used computer with AMD FX-8370 CPU and GeForce GTX 1060 6GB GPU. With time of observing is about 90 second we have got mean execution time of our function, that is demonstrated in Table 1.

Table 1 . Wetted panels benchmark results

	1280 panels	5120 panels	20480 panels	81920 panels
OpenMP	1.987 ms	3.587 ms	10.287 ms	38.498 ms
OpenCL	0.560 ms	0.646 ms	1.721 ms	5.379 ms
OpenCL/OpenGL sharing	0.540 ms	0.615 ms	1.530 ms	4.639 ms

Version with OpenCL/OpenGL sharing shows best results. OpenMP version computes approximately six times as long in comparison with OpenCL/OpenGL sharing. Each panel can be processed separately, so this task is suitable for GPU and acceleration is quite good. Finally, OpenCL version was a little slower than the OpenCL/OpenGL sharing from 3.7% on 1280 panels to 15.9% on 81920. Actually, for now this function uses the same data every time, so we copy our buffer from host to device only once. Due to this, we do not see any reasons for acceleration with sharing.

Also there is a graph, which indicates the dependence of time on no. of panels under water (Figure 3). Our function has a loop with some continue breakpoints on OpenMP, so it can be noticed, that execution time on OpenMP depends on no. of panels under water linearly, while OpenCL does not depend on it.

Additionally, 30% of execution OpenCL/OpenGL version is copying result from device to host, but soon we will rewrite code, which needs this result to avoid this copying, so we will get additionally 30% of acceleration.

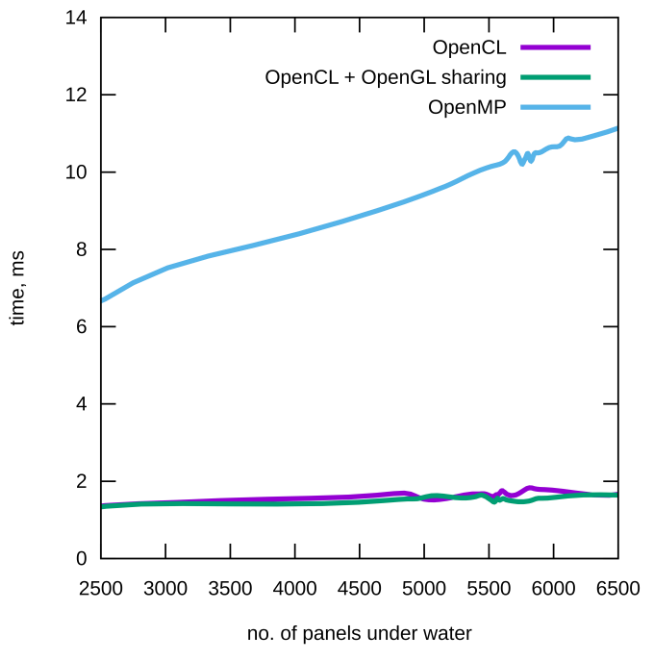


Figure 3. Wetted panels benchmark results

6. Conclusion and future work

A new version of the detecting wetted panels was developed, which executes about 6 times faster than the OpenMP version, depending on the count of input data. The version with OpenGL sharing, which was developed for the further transfer of calculations on the GPU, turned out to be faster than the version with OpenCL only. So far, we have not found a reason for this, so this is a topic for future research.

Acknowledgments

The research was carried out within framework of grant of Saint Petersburg State University no. 26520170.

References

[1] Liao W. S. et al. Real-time spherical panorama image stitching using OpenCL //Proceedings of the International Conference on Computer Graphics and Virtual Reality (CGVR). – The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011. – С. 1.

[2] Ukidave Y., Gong X., Kaeli D. Performance evaluation and optimization mechanisms for inter-operable graphics and computation on GPUs //Proceedings of Workshop on General Purpose Processing Using GPUs. – ACM, 2014. – С. 37.