ACCELERATING REAL-TIME SHIP MOTION SIMULATIONS USING GENERAL PURPOSE GPU COMPUTATIONS

Ivan Petriakov [a], Ivan Gankevich [b], Vladimir Korkhov [c], Degtyarev Alexander [d]

Saint Petersburg State University

E-mail: [a] st049350@student.spbu.ru, [b] i.gankevich@spbu.ru, [c] v.korkhov@spbu.ru, [d] a.degtyarev@spbu.ru

Software suites for ship simulations are typically used for statistical studies of ship dynamics, but also as a simulator for training ship crew in dangerous situations. One problem that arises during training is speeding-up a part of the session which does not involve actions from the crew. The aim of the study reported here is to accelerate solution of ship motion equations using general purpose computations on GPU. These equations describe dynamics of ship manoeuvring in wavy sea surface, and are central to the simulator programme. The equations are solved numerically via Runge—Kutta—Fehlberg method. Due to high number of floating point operations, computation on GPU achieves considerable speed-up over CPU. High performance solution allows to shorten training sessions and make them more efficient, but also beneficial for statistical studies as it reduces simulation time.

Keywords: GPGPU, OpenCL, ship dynamics, ocean waves, maritime simulator, virtual testbed

1. Introduction

This work is carried out within framework of the Virtual testbed project which aims to create a decision support system for simulating, predicting and preventing dangerous situations caused by physical phenomena that may occur with the ship in sea way. The situations include: the flooding of the compartment, the fire in the compartment, the loss of ship stability, large ocean waves and many others. The main feature of the project is the usage of general purpose GPU computations to improve performance. Other key feature is realistic marine objects. These are provided by IGES format, which is vendor-neutral standard for exchanging object models between CAD systems. Finally, novel wavy surface model used to simulate ocean waves [7], but at this stage it is not included in the project yet.

On-the-shore decision support system receives data in real-time from ships in the sea, simulates ship motions; to be enable to make decisions the system have to simulate ship motions in real-time. The technology that provides real-time performance is general purpose computations on GPUs. Most of the computations in the programme involve large number of transcendental mathematical functions, linear access to dense arrays and matrices, and no complex information dependencies between computation stages. It allows us to use OpenCL framework as acceleration tool.

Simply rewriting source code in OpenCL is often not enough. We need to adapt algorithms and transform mathematical formulae to make them efficient for GPU. The new mathematical formulae for velocity potential computation was derived specifically for this use case. It uses fast Fourier transforms to achieve high performance on GPU and is numerically equivalent to the known formula from linear wave theory. Multidimensional derivatives is another algorithm which requires completely different OpenCL implementation. We plan to write code generator that produces optimised OpenCL programme for each particular derivative dimension and array shape. Finally, geometric transformations (rotation and translation) are easy to do in OpenCL with lots of built-in vector functions.

2. Related work

The feature that distinguishes the Virtual testbed from other similar projects is that it is not only a ship motion simulator, but also a decision support system. This results in hard requirements on performance which are not present in other simulators. Often they are either research tools with no no performance requirements , or simulators for computer games or movies which may trade accuracy of describing physical phenomena to improve performance. The Virtual testbed needs both accuracy and performance, and this is the main difference that distinguishes it from similar programmes.

In [1] the authors develop a method for performing ship motion simulations based on physical theories with simplifications that are necessary for real-time performance. These simplifications include decomposition of the waves into head and transverse to speed-up calculation of wave excitation forces. The authors mention that using velocity potential theory to compute these forces is impractical to implement in real-time applications. Contrary to this statement we compute wave excitation forces by computing velocity potentials. The reason that this approach works in real-time is twofold. First, we use explicit formula based on Fourier transforms which is fast to compute compared to implicit formulae that slower and needs numerical methods to compute. Second, we use GPU to accelerate computations of Fourier transforms.

In [2] the authors use response amplitude operators to simulate ship motions in real-time. To increase performance they also use fast Fourier transforms, but compute everything on CPU. Finally, in [3] the authors study how GPU can be used to simulate ocean surface. In the Virtual testbed GPU is used to improve performance of wavy surface generation, velocity potential and wave pressure calculation.

3. Computation of external forces on GPU

The governing system of equations of ship motion is derived from the conservation laws for mass and moment of inertia (second Newton's law), and are described in detail in various books on ship dynamics [4,5]. The system of equations is solved numerically using fourth-order Runge—Kutta—Fehlberg method. It is not the numerical method that consumes a lot of resources, but calculation of external forces acting on the ship hull: buoyant force (due to pressure of water displaced by the ship) and wave pressure force (due to ocean wave motion). Since the calculation of these forces consumes most of the time required to render the frame, the code for computing them was rewritten for GPU.

3.1. Velocity potential

Assuming that the fluid is incompressible and inviscid, we compute wave pressure from velocity potentials. Velocity potential is determined from the following system of equations.

$$\nabla^2\phi = 0$$

$$\phi_t + \tfrac{1}{2}|\vec{v}|^2 + g\zeta = -\tfrac{p}{\rho}$$

$$\mathrm{D}\zeta = \nabla\phi \cdot \vec{n}$$

The first is the continuity (or Laplace) equation, the second is the equation of motion (or dynamic boundary condition), the last one is a kinematic boundary condition on the free surface. Since the free surface is known, the second equation becomes an explicit formula for wave pressure, and the problem reduces to finding the velocity potential φ.

$$\phi_{xx} + \phi_{yy} + \phi_{zz} = 0$$

$$\zeta_t = \underbrace{\left(\frac{\zeta_x}{\sqrt{1+\zeta_x^2+\zeta_y^2}} - \zeta_x \right)}_{f_1} \phi_x + \underbrace{\left(\frac{\zeta_y}{\sqrt{1+\zeta_x^2+\zeta_y^2}} - \zeta_y \right)}_{f_2} \phi_y - \underbrace{\frac{1}{\sqrt{1+\zeta_x^2+\zeta_y^2}}}_{f_2} \phi_z$$

The system is solved using the Fourier method with some physical and mathematical simplifications. The full solution is written as a convolution of some window function with superposition of wavy surface derivatives. If we use the assumption of small-amplitude waves, the solution is reduced to the solution from the linear wave theory.

$$\phi(x,y,z,t) = \mathcal{W}_2(x,y,z) * \frac{\zeta_t(x,y,t)}{F\left(f_1,f_2,f_3\right)}$$

Convolution is implemented as three Fourier transforms which makes it efficient to compute on GPUs. The formula is suitable for any wavy surface regardless of the mathematical model used to generate it as long as it is physically feasible: the same formula can be used for both plain and irregular waves of arbitrary amplitude. Full derivation of the formula is given in [7].

3.2. Buoyant force

The buoyant force is proportional to the mass of the water displaced by the ship. In order to compute it we decompose ship hull into triangular panels and determine underwater panels. For partially submerged panels we compute their intersection with the water level (assuming that it is a straight line) and consider only submerged part in the later computations. For each panel we compute pressure due to water displacement and pressure due to ocean wave motion.

$$p(x,y,z,t) = -\rho\phi_t - \rho\frac{1}{2}\left(\phi_x^2+\phi_y^2+\phi_z^2\right) - \rho g z$$

Then the force is simply a vector in the opposite direction of the surface normal to the panel with length proportional to the pressure and panel area:

$$F(x,y,z,t) = -pS\vec{n}$$

The total buoyant force is the sum of all individual forces acting on each submerged panel. If one omits velocity potential terms, the formula becomes the Archimedes law for a single material point. This algorithm has linear memory access pattern and involves a lot of floating point operations, as well as vector arithmetic, which makes it efficient to implement on GPU.

4. Evaluation

The algorithms described in the previous section were implemented using OpenMP and OpenCL, and using both CPU and GPU significantly improves performance compared to using only CPU. We can see the performance difference between CPU and GPU implementation on the next graphs. All tests were run on computer with AMD FX-8370 CPU and GeForce GTX 1060 6GB GPU.

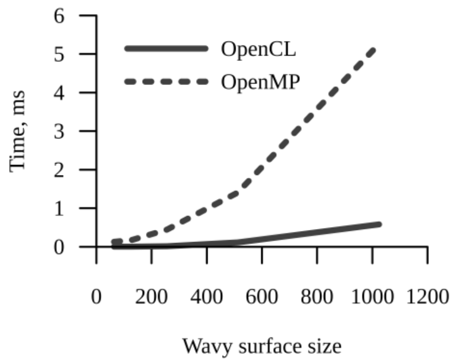

Figure 1. Velocity potential computation performance

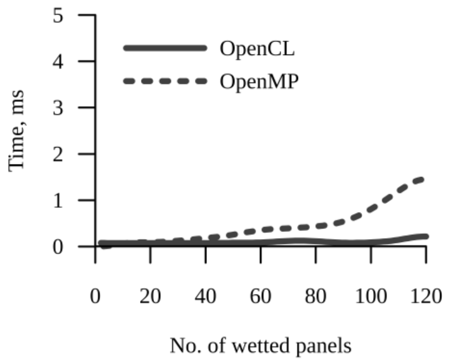

Figure 2. External force computation performance

The first experiment was to measure OpenMP implementation of velocity computation and compare execution time with OpenCL implementation. As was mentioned, it uses three fast Fourier transforms, that can be done very effectively on GPU [6]. So, thanks to this ability, OpenCL has better performance. This fact is illustrated in Figure 1 that shows how performance scales with the number of grid points.

The next test is similar to the previous one. We run the testbed with different frameworks to compare time to compute pressures on ship hull. The wave pressure formula is easy to be parallelized with OpenCL. Performance measurements are presented in Figure 2.

5. Discussion

We implement and test everything on CPU with OpenMP, and rewrite the programme hot-spots in OpenCL. This allowed us to achieve approximately 30 frames per second on the target machine. However, there is a room for improvement. We compute velocity potential derivatives on the CPU which involves a lot of data copying between CPU and GPU memory. Also, we store ship hull vertices and indices in CPU memory which causes even more data copying. In the future, we plan to eliminate data copying overhead by storing everything in GPU memory and computing derivatives using optimised OpenCL kernels.

In similar studies the authors often apply simplifications to the ship motion equations or pressure computations formulae to achieve real-time performance. We choose to optimise only computational part of the problem by offloading programme hot-spots to GPU. To optimise mathematical part we use explicit formula for velocity potentials which works for any physically feasible wavy surface. Thus, there is no restriction on mathematical model for ocean waves that can be used in our programme.

6. Conclusion and future work

New implementation involves a lot of data copying between CPU and GPU memory, which slowed the application a little bit. The next goal is to rewrite everything in OpenCL. This will further speed-up computations and remove expensive data transfers.

Acknowledgements

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

References

[1] Chen X. et al. Real-time simulation of ship motions in waves //International Symposium on Visual Computing. – Springer, Berlin, Heidelberg, 2012. – C. 71-80.

[2] Varela J. M., Soares C. G. Interactive Simulation of Ship Motions in Random Seas based on Real Wave Spectra //GRAPP. – 2011. – C. 235-244.

[3] Ma X., Chen Z., Shi G. Real-time ocean wave motion simulation based on statistic model and GPU programming //Information Science and Engineering (ICISE), 2010 2nd International Conference on. – IEEE, 2010. – C. 3876-3880.

[4] Matusiak, J. Dynamics of a rigid ship //Helsinki: Aalto University, 2013

[5] Kornev N. Ship dynamics in waves //Rostock: University of Rostock. – 2011.

[6] Volkov V., Kazian B. Fitting FFT onto the G80 architecture // University of California, Berkeley. – 2008. – T. 40.

[7] Gankevich I., Degtyarev A. (2018) Simulation of Standing and Propagating Sea Waves with Three-Dimensional ARMA Model. In: Velarde M., Tarakanov R., Marchenko A. (eds) The Ocean in Motion. Springer Oceanography. Springer, Cham.