Realtime GPU-Accelerated Smoothed Particle Hydrodynamics with Rigid Body Interaction

Shaun Silson

April 5, 2013

1 Overview

Basic fluid simulation is a thoroughly saturated research area, but the two forefronts are solid-fluid coupling and realtime performance. The current state of the art in solid-fluid coupling with SPH is [2]. This is a *non-GPU* implementation and the simulation times range from 1 second to 1.5 minutes per frame for 100k to 200k particles. On the other hand there is [3] which is GPU-accelerated and achieves 17fps for 130k particles *without rendering* and 11fps with GPU accelerated marching cubes rendering (the paper focuses largely on rendering).

So my aim will be to combine the two elements and achieve realtime solid-fluid coupling. to do this I will bootstrap my implementation off of [4]. Currently this implements a GPU accelerated framework that achieves 23fps for 200k particles and 4.2fps for 1M particles. both of these are *without* rendering, and that does not seem to be a major focus of the project (although it is on the feature wishlist).

	Akinci 2012	Goswami 2010	Hoetzlein 2012
Focus	CPU based Solid/Fluid SPH	GPU accelerated SPH	Realtime GPU SPH
Issues	Not GPU	Focused on rendering, no rigids	No Solid-Fluid
Performance	1 second/1 minute per frame (100-200k particles)	17fps for (130k particles)	23fps (200k particles)

A potential extension will be animated fluid control, the best paper I could find for this is [6] which describes a system of control particles for LBM but says it can be applied to SPH. The demo [1] shows fluids being morphed into shapes and then released to splash. All simulation seems to be done offline so getting this to occur in realtime might be over-ambitious, but using the control particle idea might be interesting.

2 Details

2.1 Solid-Fluid Coupling

As described in [2] the mesh will be pre-processed into a cloud of particles. This step is not explicitly described in the paper so I assume it is completely separately to the simulation so I will consider it out of scope for my research. I will need to find some way of transforming a triangle mesh into a point cloud.

The simple case will be to get a body to float on top of the water and displace an appropriate amount of fluid. To get the body to float I will calculate the buoyancy and divide it by the number of fluid particles in contact, thus the sum of all contacting fluid particles will be the correct amount of force to keep the body afloat. I will use Bullet to handle the rigid body simulation and update the fluid forces between fluids simulation steps. Fluid displacement should be occur naturally because the particles making up the solid will be the same as the fluid particles.

Once the body floats the next step is to apply viscosity forces to make objects able to tumble as they are splashed. This is described in some detail in [2] so I will base my implementation on theirs.

2.2 Realtime Performance

As mentioned before I will be basing my work on an existing GPU based system so the porting itself will not be part of the scope of my work. That said there remains more scope for speedup in two key areas: improving the neighbour-search algorithm and allowing larger timesteps by implementing imcompressibility correction with PCISPH.

2.2.1 Neighbour-Search

The most expensive part of SPH is determining which other particles lie within a particular particles sphere of influence. Z-indexing as described in [3] is a more efficient memory layout which ensures that particles which are close in 3D space are also close in linear memory. This is as opposed to row-major multidimensional arrays where memory access is scattered. Because of the locality of the interacting particles we should see improved performance because interacting clusters can fit into shared memory (instead of global memory) which is faster.

2.2.2 Incompressibility

The size of the simulation timestep affects water compressbility, with a larger timestep allowing more artefacts to occur. PCISPH [5] uses a solver between simulation steps to correct compressibility artefacts allowing larger timesteps, which should improve simulation performance.

References

[1] Blender magic fluid simulation - http://www.youtube.com/watch?v=WruTNnF6Ztg.

[2] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics*, 31(4):1–8, July 2012.

[3] Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. Interactive SPH Simulation and Rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010.

[4] Rama Hoetzlein. Fluids v.3: A Large-Scale, Open Source Fluid Simulator (http://www.rchoetzlein.com/fluids3/).

[5] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. *ACM Transactions on Graphics*, 28(3):1, July 2009.

[6] N Thuerey, R Keiser, M Pauly, and Ruede U. Detail-preserving fluid control.