Ship in a Bottle

Saket Patkar and Bo Zhu

1 Modeling and Rendering the Water

We decided to get the basic ocean surface through a particle level set fluid simulation. The fluid simulator can only handle simple manifold meshes. Unfortunately the ship was a complex mesh consisting of 600 sub meshes (as shown in Figure 1).

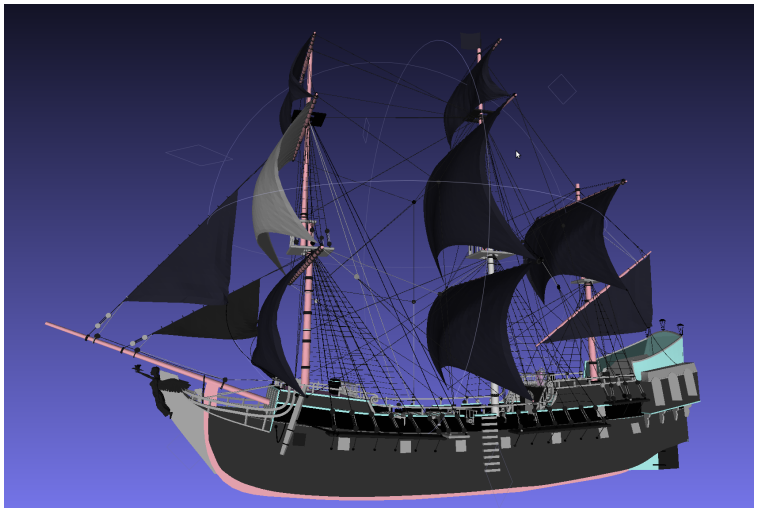

Figure 1: Ship mesh as viewed in meshlab.

Hence I first imported the mesh in 3ds Max and deleted most of the components other than the hull of the ship. The hull was still not a single piece and hence could not be imported into the simulation. After trying a lot of things, taking the convex hull of the hull mesh seemed to do the trick (as shown in Figure 2).

Figure 2: Simplified ship mesh.

We ran a fluid simulation [4] using the simplified mesh. At first we ran a very high resolution simulation that gave us nice waves but we realized that the big waves did not suit the scene. Moreover it was giving us small scale features while we were more looking to get large scale waves. Hence we ran a course simulation where a source seed water in a sine wave pattern. This gave us nice bulk waves. Figure 3 shows the results of both the simulations.

To make the surface of the water look like ocean we added a displacement based on Phillips spectrum see [1] for more details. The basic idea is to superpose many sine waves of different amplitudes and frequencies to get a displacement map. The Phillips spectrum gives the amplitude as a function of frequencies and is used to create a time dependant height field in frequency space. A FFT of these amplitudes converts it from

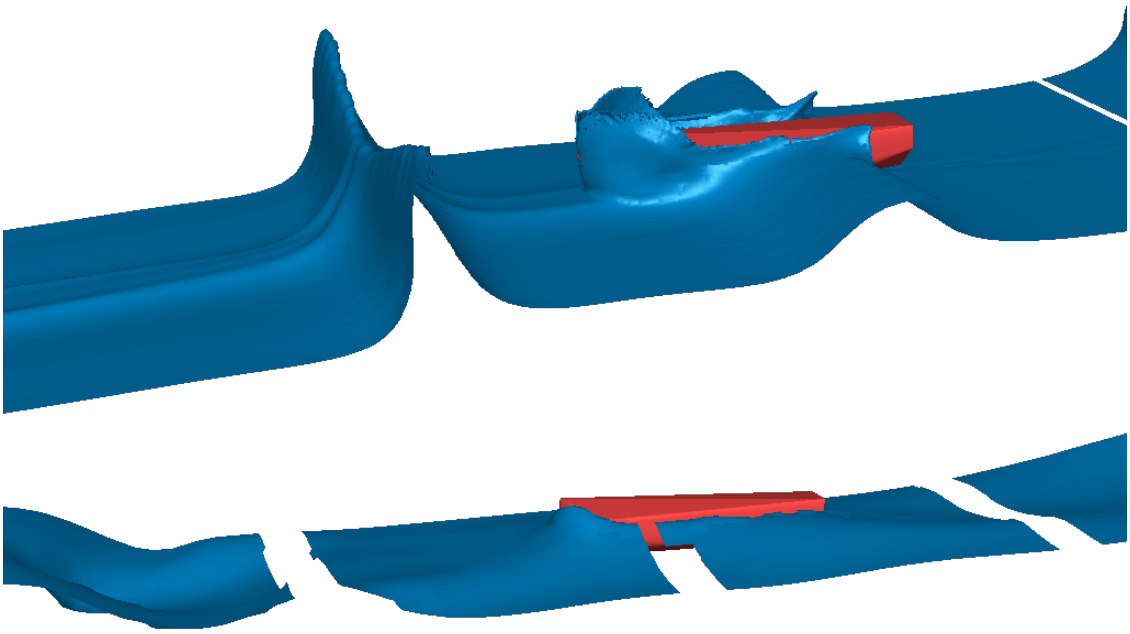

Figure 3: (Top) A high resolution fluid simulation. (Bottom) Lower resolution simulation that has better bulk waves.

frequency space to the position space. Now we have a $2D$ displacement map (more like a height field). This map can be used to advance the fluid level set using Semi Lagrangian advection to get the bumpy ocean like surface as shown in Figure 4.

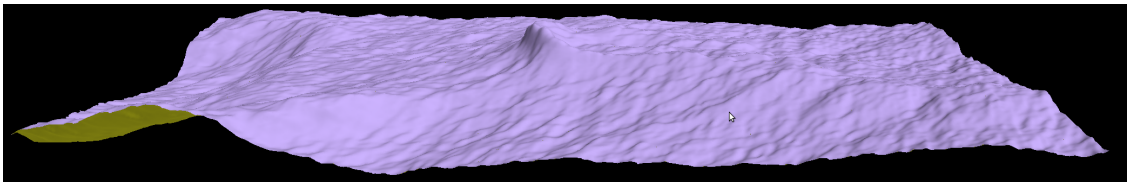

Figure 4: Fluid surface after applying displacement based on Phillips spectrum.

Finally this surface needed to be clipped w.r.t. the bottle. To do this we created a signed distance field for the bottle. We also had the bottle triangle mesh. Then we transformed the simulation level set to the same coordinate system. Now for every point on the level set grid we shot rays in two opposite directions to determinng whether the voxel is inside the bottle. We also used the bottle's level set to determine this. We had to initialize the cells near the bottle with correct distance values to avoid getting a jagged surface. We ran a fast marching on this surface in the end to get a signed distance field. Finally we ran a dual contouring algorithm to get a triangulated surface from the level set. Figure 5 shows the final result.

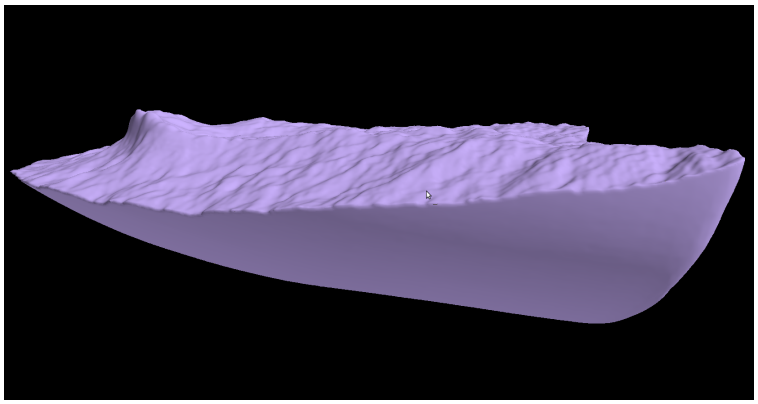

Figure 5: Fluid surface transforming, interpolating and clipping by the bottle.

2 Sand and Spray Particle Rendering

In order to get high resolution sand in a complex scene, we model sand as particles and run a physics simulation for the sand and the environment. The fine sand grains are generated based on the coarse simulation results adaptively in an view-dependent way. We show the rendering results with different sampling parameters.

2.1 Sand Modeling

Sand is modeled as a collection of particles. Each particle is treated as a rigid sphere, and the interaction between two rigid spheres is modeled in a impulse-based way considering the repulse force and damping force as in [3]. A hybrid particle-height field model [5] is used to accelerate the simulation by fixing the positions of particles under the surface flow relying on the fact that the flow of granular materials only happens on the surface.

More specifically for the scene setup, we first model the terrain and the book as a triangle mesh. Then we convert these two meshes into surfaces of dense enough particles in order to incorporate them into the particle-based sand simulation algorithm. Then we seed a thin layer of particles from a square source covering the entire visible part in the scene, and run simulations for a number of time steps until all the particles are fallen and static on the ground (as shown in Figure 6). These particles are then used as the input for the point cloud render process.

2.2 Sand Rendering

For the sake of rendering high resolution granular material, particles are upsampled procedurally based on the results we have got from the physics simulation. A number of sub particles are seeded from each simulation particle, and placed randomly within a certain radius to the particle center. This upsampling process can produce realistic rendering results for the sand grains near the camera, however, it is a waste to sample the same number of particles for those particles far from the view occupying merely one or several pixels. In order to perform this upsampling process efficiently, we implemented an adaptive view-dependent point sampling algorithm for sand rendering. This algorithm is motivated by [2], in which sand particles are treated in a LoD way based on its distance to the camera. Similarly, we define four parameters for the adaptive sampling process: R – the sampling radius for each simulation particle, r – the radius for each

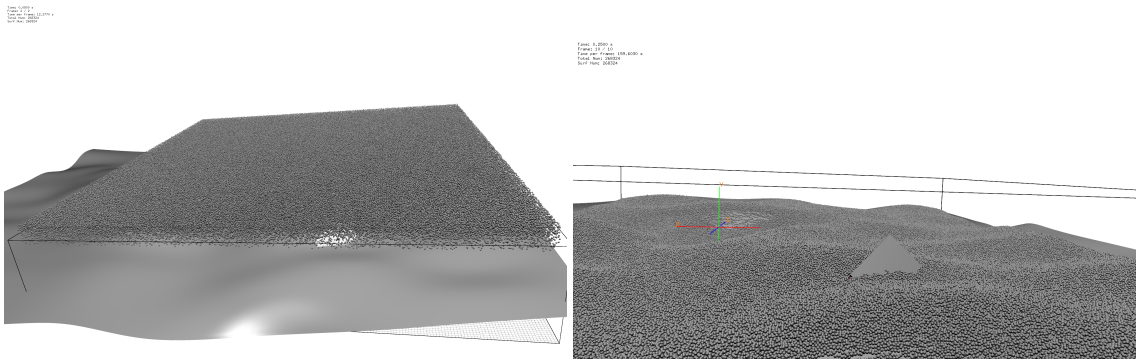

Figure 6: Particle-based sand simulation. The left image shows the initial configuration of our simulation. The right image shows the results after 10 timesteps. 22k particles are used in this simulation.

upsampled particle, N – the maximum number of upsampled particles, and n – the minimum number of upsampled particles. N and n are calculated as a quadratic function of the distance from the particle to the camera.

As shown in Figure 7, different parameters are used to show the different levels of details for sand rendering. The top-left image shows the direct rendering of the simulation particles and the bottom-right image shows the rendering of the upsampled fine grains that is used in our final image.

2.3 Foam Rendering

Foam particles are seeded around the ship near the level set water. These particles are rendered in the same way as the sand particles but with different materials.

3 Team Member Contribution

Bo: sand modeling and rendering, foam rendering, rope modeling and rendering, and other miscellaneous rendering and modeling tasks (e.g., modeling and texturing the cork, book, scene setup, etc.)
Saket: Modeling and rendering water and other miscellaneous rendering and modeling tasks such as the octopus and barrels.

4 Results

References

[1] Simulating ocean waves with fft on gpu. `http://www.edxgraphics.com/blog/simulating-ocean-waves-with-fft-on-gpu`.

[2] Chris Allen, Doug Bloom, Jonathan M Cohen, and Laurence Treweek. Rendering tons of sand. In *ACM SIGGRAPH 2007 sketches*, page 27. ACM, 2007.

[3] Nathan Bell, Yizhou Yu, and Peter J Mucha. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86. ACM, 2005.

Figure 7: Adaptive granular material rendering. Parameters for the 4 images are (from left to right, top to bottom): R=.05, r=.05, N=1, n=1; R=.05, r=.02, N=10, n=2; R=.08, r=.01, N=30, n=10; R=.1, r=.005, N=80, n=50.

[4] R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 85–94. ACM, 2013.

[5] Bo Zhu and Xubo Yang. Animating sand as a surface flow. *Eurographics 2010, Short Papers*, 2010.

Figure 8: Ship in a bottle: final rending image.