Final Animation Project
Fluid Simulation

Antoine Racine-Gingras
Manel Guay-Montserrat
Thanasis Barbouras
Patrick Chan

Presented to Mr. Tiberu Popa
Concordia University
Animation in Computer Games
COMP 477
December 3rd, 2018

Introduction:

After exploring a variety of topics in this course, we decided to proceed with our final project on the topic of Fluid Simulation. Our goal for this project was to attempt to create a realistic and accurate fluid simulation, using efficient algorithms which can allow for a large number of particles. Though we could have implemented the project through a grid based system, we decided to use the approach of a particle system to implement smoothed particle hydrodynamics (SPH) interpolation. It is a well-researched algorithm based on laws of physics, notably the Navier-Stokes equation, Newton's Second Law adapted for fluids. As we have limited time and no experience on the subject, we all agreed that exploring particle fluid simulation is most suited for our current capabilities. The project will be developed using C++. The library used to render the 3D models will be in OpenGL and Eigen for computing the mathematics necessary to give us the desired results.

Motivation

Fluid simulation is a topic that has been researched extensively and there are a variety of resources to draw information from. We chose this as our topic because it feels like a challenging topic to explore, as the laws of physics, when applied to fluids, because complex. Indeed, fluid dynamics are still a subject of research. Originally, we were curious to explore this topic by simulating the physics property of oobleck (a particular mixing ratio with cornstarch and water mixture) solution. After some discussion, and because of the complexity of the topic, we concluded that it is strategically better to study basic fluid simulation through a conducted research, attempt to recreate the basic implementation and only at the end will we try to add more if time permits.

The Problem

Throughout the project, we knew we wanted to implement SPH to simulate the fluid simulation. Despite covering the topic of fluid simulation in class, we knew it wasn't quite enough to accomplish what we intended to do.

In our research process, we discovered numerous reports from a variety of researchers on the topic. What was interesting from our discoveries were the context of the explanations that different authors used based on their domain of expertise. This admittedly led to some conclusions, because equations like the Navier-Stokes equations were presented differently. Among the different researches, we found fluid simulation using mathematics from numerical methods, physics and differential equations.

Aside from deciding the method we would use to implement the calculations, we need the option to programmatically solve differential equations. Without attempting to “reinvent the wheel”, we went to look for C++ libraries that may be useful for the task. At first, we were thinking of using the Boost library. Unfortunately, the size of the library was too big to transfer all at once (4GB). Instead, we decided to use the Eigen library for its mathematical formula. We manually coded everything directly related to the water physics by ourselves.

Algorithms and Coding

To obtain a realistic fluid simulation, the following variables need to be taken into account: density, pressure, time, viscosity, acceleration (usually simply replaced by gravitational acceleration). All the variables required were calculated using the formulas shown in class, with slight variations to some of the numbers, to better fit our setup. Furthermore, an Octree data structure was used to get the neighbors of a given particle, using the kernel radius. Some damping was also included to deal with collisions with the boundaries. The integration method implemented was explicit Euler, hence giving some unwanted results due to inaccuracy. We used an older version of Open GL (2.1), the Eigen library and Intel’s TBB, to parallelize some of the code and improve performance.

Retrospection (what worked, what did not)

While all of the physics were implemented, some of them are not working properly. The variables that influence the simulation correctly are gravity, viscosity and forces. However, we were not able to make the fluid incompressible, which means that the density within the WaterParticle does not remain constant. This is why we can observe that the water particles fall flat and collide into each other, ending up at the corners rapidly.

Though we have implemented our base code for fluid simulation, we didn’t have enough time to include any properties that will simulate the physics property of oobleck.

How to compile and run the program

Requirements: you need Microsoft Visual Studio. We cannot guarantee versions prior to 2010 work. (It has to be at least compatible with Open GL 2.1)
Simply click on the Visual Studio solution FluidSimulation477.sln to open the solution of the program.Then, since the program is computationally heavy, we strongly recommend running in release mode (**not in debug mode**), which is done by pressing Ctrl + F5.

Controls

While the program is running, press 'R' to reset the simulation. The particles’ initial position are generated randomly, so the results will be different every time.

Workload distribution

- Everyone contributed to finding resources and research documents to get an idea of how to implement the physics.
- Manel Guay-Montserrat worked on implementing the water physics.
- Thanasis Barbouras worked on the implementation of the OctTree. He also helped on implementing the water physics with Manel.
- Patrick Chan worked on implementing the WaterParticle class, linking the project with the Eigen library and the website.
- Antoine Racine-Gingras was the source control expert throughout the project. This includes merging Patrick’s WaterParticle class with Thanasis’s OctTree. Furthermore, he was responsible for the Open GL implementation and the visual representation of the simulation.

External Resources used

- https://users.encs.concordia.ca/~grogono/Graphics/fluid-5.pdf
- https://www.3dgep.com/simulating-particle-effects-using-opengl/
- https://pdfs.semanticscholar.org/e008/6d3fcdfae7bbb3c3162271e5cfd2c5fe1117.pdf
- https://cg.informatik.uni-freiburg.de/course_notes/sim_10_sph.pdf
- https://pdfs.semanticscholar.org/fbc1/b9c2a3438c79a1ef8684bc87ac5e2149643e.pdf
- http://matthias-mueller-fischer.ch/publications/sca03.pdf
- https://bigtheta.io/2017/07/08/implementing-sph-in-2d.html
- https://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc16/13/thesis.pdf
- https://github.com/Maarten-vd-Sande/smoothed-particle-hydrodynamics
- https://www.youtube.com/watch?v=SQPCXzqH610
- https://www.youtube.com/watch?v=tAXHCAEgSuE
- https://www.youtube.com/watch?v=l-f16KjR9Bw
- https://www.youtube.com/watch?v=eSuV3ocghSI&list=PLRVC3pHRToNHywni60N8qhd_NAoUbjF0
- https://www.youtube.com/watch?v=3BGKn8PyIHo
- https://www.youtube.com/watch?v=f5jKknVBflM
- https://www.youtube.com/watch?v=2nXkHcuQAqM
- https://www.youtube.com/watch?v=n-WXjuF5iyI