Computer Simulation of Fluid Dynamics

Brian Malsan

Undergraduate Computer Science Capstone

Capstone Advisor: Dr. Michael Reale

Spring 2018

**Introduction:**

We set out to make a fluid simulation program that acts with objects based on density. I wanted to create a fluid simulation due to my interest in science, where a large area of study is how fluids interact with solids and other fluids. Fluid simulation is a very important field in computer graphics, used in video games, films, and scientific work. Since they are in such high demand, there is a need for fast and accurate mathematical equations and display of fluids. We initially planned for a simulation in three-dimensions, but eventually settled for a two-dimensional fluid simulation using the Smoothed-Particle Hydrodynamics method.

**Related Work:**

Fluid simulation has been a heavily studied field in computer graphics. The first instance of using computers to model fluid flow was in the early 1960s by a group led by Francis H. Harlow (Harlow, 2004). This was the first instance of a two-dimension representation. The first paper describing three-dimension fluid simulation was published in 1967 by John Hess and A.M.O Smith. In the paper they described a way to discretize the surface of the fluid with panels (Hess, 1967).

There are two most common methods to represent fluids. These methods are Lagrangian and Eulerian. The Lagrangian methods represents fluids as a particle system, while the Eulerian methods look at the fluid through a grid structure that store’s information about the fluid (Strantzi, 2016). The most used Lagrangian method is Smoothed-Particle Hydrodynamics (SPH), which was discovered in 1977 to initially solve astrophysical problems. The most popular form of Eulerian method is the Marker-and-Cell method.

Many of these methods are derived from the Navier-Stokes equations for incompressible flow. Using the equations, you solve for the position, velocity, pressure, and forces acting on each particle or on the fluid grid depending on the method. The equations include:

Equation 1 states that the divergence of the velocity field(u) is equal to 0. This allows for the conservation of mass in the fluid system, derived from the Helmholtz-Hodge Decomposition. The second equation solves for the velocity field over time. The terms included in the equation are advection (), pressure (), diffusion () and the external forces (). Solving for all these terms will give you the velocity field (Nvidia, 2007).

Smoothed-Particle Hydrodynamics is a mesh-free Lagrangian method developed by Gingold, Monaghan and Lucy. The method splits the fluid up into particles. These particles have a spatial distance, known as the smoothing length, over which the particles are “smoothed” by a kernel function (Wikipedia, 2018). This means that the properties of each particle can be found by summing the relevant properties of all the other particles within a specified range. This is the neighborhood of particles around each particle. Using this method provides a sufficient way to approximate fluid motion.

**Method:**

The program was written in C++ using OpenGL, GLM, GLFW, GLEW, and GLSL. The program is made up of a main file, three classes, helper functions to set up OpenGL and different type of shaders. I used the “SPH simulation in OpenGL compute shader” project by Samuel Gunadi on GitHub as a base for my project, and then built upon it (Gunadi, 2017). This project provided insight on how to set up the compute shaders and math to get everything to work. I split his project into different classes to allow for future developments, for example different types of fluids. Also, added functionality for the objects and mouse, while adjusting some of the variables.

**C++ Classes:**

* **Fluid**- Handles the initialization of all the particles, update and drawing of all the particles. The particles are placed in GL\_SHADER\_STORAGE\_BUFFER that is sent to the compute shaders. The buffer holds each particle’s position, velocity, force applied, density and pressure.
* **Mouse**- Handles all mouse interaction with the screen. Using the left mouse button will apply an external force to all the particles that are in a box around the mouse position. The mouse is given to the fluid class, so the shaders can do the detection of whether a particle is being affected.
* **Object**- Handles the creation and drawing of on object to the screen. The object can be given three different starting positions based on what command line argument is given. The object is given to the fluid class, so the shaders can do collision detection. The collision detection only works if the object is square or rectangular.

The use of compute shaders was a big part of my program. Compute shaders have been part of OpenGL core since version 4.3. Compute shaders are very different from other types of shaders. The other types of shaders have a well-defined set of inputs and the frequency at which they execute is specified by the nature of the shader (Khronos Group, 2018). Compute shaders have no pre-defined inputs and outputs and the frequency that the shader executes is defined by the user. Compute shaders are a tool well suited for linear arrays of particle systems, which is used in my project.

In my project I use three different compute shaders. They are used to do all the mathematic calculations that allow for Smoothed-particle hydrodynamic to work. The equations use the 6th polynomial smoothing kernel function.

**Compute Shader Code:**

* **compute\_density\_pressure.comp** – calculates the density and pressure of each particle. It does it using this equation:

* **compute\_force.comp** – calculates the viscosity, pressure and external forces acting on each particle. It uses the following equations

Then the total force acting on a particle is the summation of the particle’s pressure, viscosity and external forces.

* **integrate.comp** – calculates the new velocity and position of each particle. This is also where collisions and boundary conditions checked.

**Results:**

The result of the finished project is a two-dimension fluid simulation using smoothed-particle hydrodynamics. The simulation allows for three different starting positions of the particles and three different positions for an object in the scene. This object acts as an obstacle that interacts with the fluid but can not move. The simulation also allows for mouse interaction. By pressing the left mouse button, an upward force will be applied to the particles around the mouse position.

I ran the program using two different graphics cards. The first one is an Intel HD Graphics 620 and the second card is a NVIDIA GeForce 930MX. Running the program with 15,000 particles. The Intel graphic card ran at about 20 frames per second, while the Nvidia car ran at about 18 frames per second. I thought that the Nvidia card would run faster. However, from looking at the two results the Nvidia card allowed for more precise and quality of particle movements. Despite the slight slowdown, the use of the Nvidia graphics card allow for better fluid simulation.

The following are some images to showcase different parts of the project.

Figure 1: Particle starting point with no object

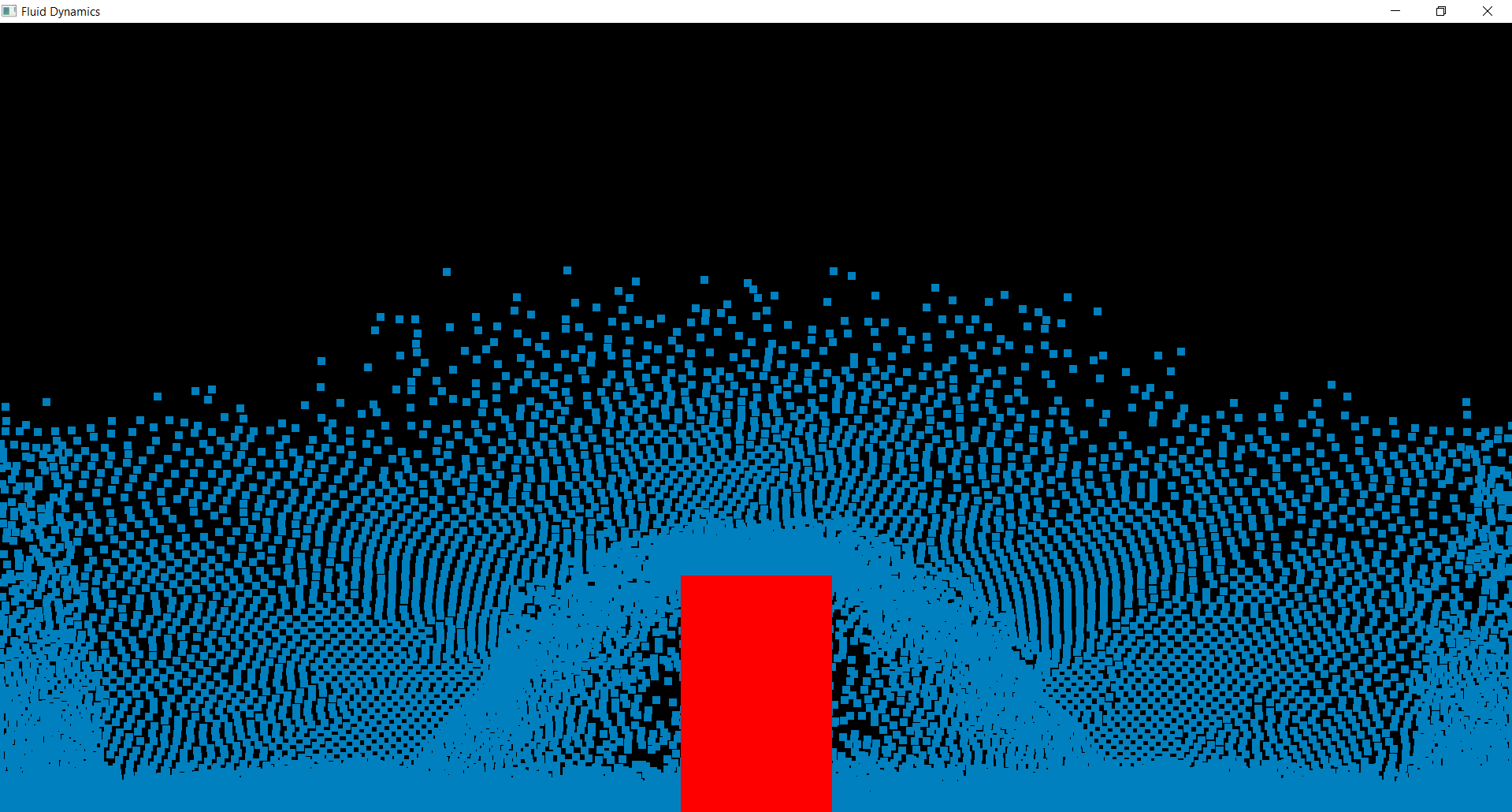


Figure 2: Particles started in center and interacted with object on floor



Figure 3: Particles started on the left side of the screen with a stationary object floating off the floor



Figure 4: Particles moving freely underneath an object

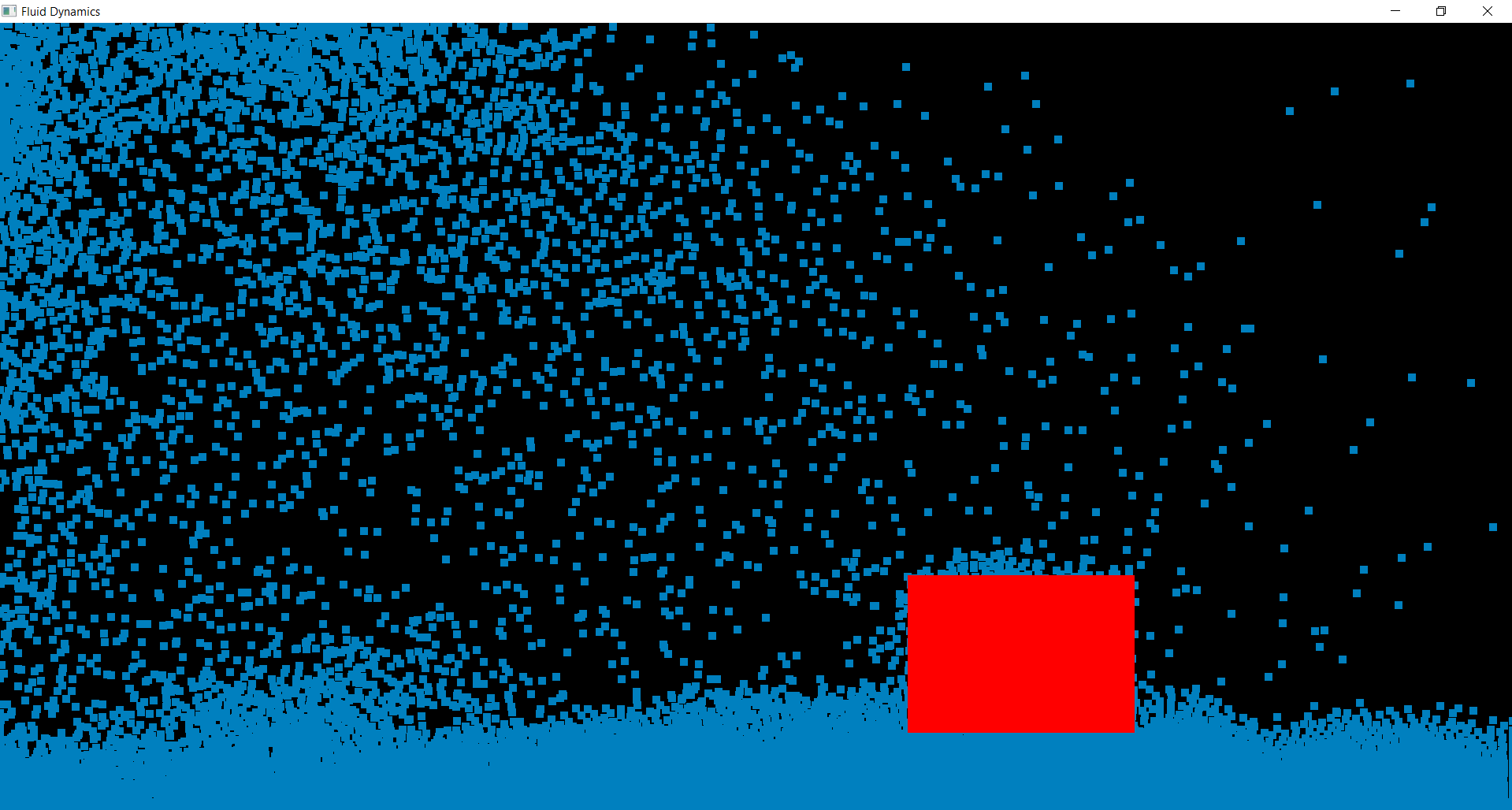


Figure 5: Mouse interaction

Figure 1 shows one of the three different starting locations for the particles with no object in the scene. In Figure 2 the particles started in the center of the screen and over time moved downward. This figure shows a clear depiction of how the particles interact with an object. Figures 3-5 all take place during the same run. Figure 3 shows the particles starting on the left side of the screen and a stationary object floating off the floor. Figure 4 takes place a few seconds later. This shows that the particles can move freely underneath and around the object. This proves that the object can be moved anywhere on the screen and the collisions only occur where the object is. Lastly, Figure 5 shows off what happens after the user applies an upward force to some of the particles by using the mouse.

**Discussion:**

After six months of working on my capstone project I had to deal with many roadblocks while also learning a lot. One thing I learned was how helpful prototyping is. Throughout my project I created many prototypes, for example to get the compute shaders to work, this allowed me to work on specific parts of my project while not accidentally breaking my original project. I was also able to learn a lot about time management, especially how to schedule out my time to be able to work on multiple projects at once while getting all my school work done for other classes.

Some things that did not work for me was the use of OpenGL textures. I spent many weeks trying to get them to work however I never really made any progress. Especially getting the textures to work within the shader code. That’s why I changed my approach to use a smoothed-particle hydrodynamics technique. This approach made a lot more sense to me and in my mind was easier to implement the math behind it.

The biggest road block that I faced was being able to communicate information from the CPU to the GPU and back using compute shaders. Working on this set me back about 3 weeks, putting in hours of work trying different things and getting some strange results. I fixed this problem by using a single buffer and offsets to point to get the correct position in memory. This was a negative experience, but I was able to work through it.

If I was to do the project all over again, I think I would start out by aiming to do a two-dimensional fluid simulator and not a three-dimensional simulator. This way I would haven been able to add some cool features that would make the project better. Having a clear vision for what I wanted to complete would have made the project more enjoyable.

**Conclusion and Future Work:**

Fluid simulations have been a useful and widely studied topic. The two main types of methods to represent fluid dynamics are Lagrangian and Eulerian. For my project I use the Smoothed-Particle Hydrodynamics method to create a two-dimension fluid simulator. Since no project in computer science is truly finished, future work on the project includes allowing the objects to move freely, the object would interact with fluid and float or sink depending on density of both the fluid and object. A second thing I would like to implement is allowing for different fluids to be simulated not just water. This would happen by changing some constant values. Lastly, since I couldn’t get the simulation to work in three dimensions, which I set out to do, I would work to add the third dimension. I have been able to learn a lot and gain valuable experience that I can take with me in future education and jobs.

**Appendix:**

Dependencies:

* OpenGL version 4.3 or newer
* GLM
* GLFW 3.2.1
* GLEW 2.1.0
* Assimp 3.3.2

Install and build:

1. Download source files
2. Use Cmake to build project
   1. Give where the source code is located
   2. “Where to build the binaries” create a folder called build inside of your project folder and select that.
   3. Click configure
   4. Change MASTER\_DEPEND to location of installed dependencies
   5. Click configure again, then generate
   6. Open project
3. If using Microsoft Visual Studio make sure to set your project as “StartUp project”
4. Give the command line arguments you want, or none
5. Click run

Command line arguments-

Give the program two numbers (0-2). The first number states the starting position of the particles and the second states the location of the object.

First argument:

* 0- Particles start in the center of the screen
* 1- Particles start on the entire left side of screen
* 2- Particles start in the top right corner of the screen

Second argument:

* 0- Box is in the middle of the screen on the floor
* 1- Box is in a position off the floor
* 2- Box is in a position off the screen

No command line arguments are the same as giving 0 0.

References

Harlow, F. H. (2004). Fluid dynamics in Group T-3 Los Alamos National Laboratory:(LA-UR-03-

3852). *Journal of Computational Physics*. Elsevier.

Hess, J.L.; A.M.O. Smith (1967). Calculation of Potential Flow About Arbitrary Bodies. *Progress in*

*Aerospace Sciences*. **8**: 1–138. Bibcode:1967PrAeS...8....1H. doi:10.1016/0376-0421(67)90003-6.

Gunadi, S. (November 2017). *SPH simulation in OpenGL compute shader*. Retrieved from

[https://github.com/multiprecision/sph\_OpenGL](https://github.com/multiprecision/sph_opengl)

Khronos Group (15 January 2018). *Compute Shader*. Oregon, USA: Khronos Group. Retrieved from

<https://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc16/15/thesis.pdf>

Nvidia (2007). GPU Gems. *NVIDIA CUDA Library: Main Page*. Retrieved form

developer.download.nvidia.com/books/HTML/gpugems/gpugems\_ch38.html.

Strantzi, A. (2016). *Fluid Simulation Using Smoothed Particle Hydrodynamics (SPH)*. Retrieved

from <https://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc16/15/thesis.pdf>

Wikipedia (2018). Smoothed-Particle Hydrodynamics. *Wikimedia Foundation*. Retrieved from

en.wikipedia.org/wiki/Smoothed-particle\_hydrodynamics