Creating and Optimising a Fluid Simulation using Smoothed Particle Hydrodynamics

BSc Computer Science

Sakir Azimkar
200823588

Supervised by: Richard Davison

April 2025

Abstract

This dissertation explores the Navier-Stokes equations, which are used to describe the motion of fluids. It briefly looks at an Eulerian implementation introduced by Jos Stam [1], before shifting focus on to Lagrangian methods - specifically Smoothed Particle Hydrodynamics (SPH). Furthermore, this paper describes an implementation of this technique.
The main goal of this project is to produce a real-time fluid simulation that can be integrated into a video game, with minimal performance loss. It is an optimisation problem, motivated by an interest in the GPU and shader languages. The final project is a Smoothed Particle Hydrodynamics simulation that can reach 400 frames per second with over thirty-two thousand particles on modern hardware.

Declaration

"I declare that this dissertation represents my own work, unless explicitly stated otherwise."

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Richard Davison, for being reliable and helpful throughout the entire project. I'd also like to thank Gary Ushaw, my "second" supervisor, who was always encouraging and present at our meetings. Lastly, I'd like to thank my friends and family for supporting me during this time. This dissertation is dedicated to these people.

Contents

List of Figures

1 Introduction

This section highlights the motivations, aims and objectives of this project. It talks about the overall dissertation structure and compares the differences between this project and the initial proposal.

1.1 Motivation

Fluid dynamics are observed in all aspects of daily life and in research. It is crucial to be able to simulate and understand the behaviour of all fluids, including liquids and gases. There already exists a number of different techniques to achieve this, through utilisation of the Navier-Stokes equations. These methods have been crucial in research and in industry. For example in aerospace, which utilises Computational Fluid Dynamics (CFD) for aerodynamic analysis, aircraft design optimisation and thermal management, to name just a few. As our knowledge of fluids and CFD techniques improve, the global market is expected to increase by nearly triple its current value by 2033 [6]. Simulations are also essential for cost reduction, as they reduce the need for physical prototypes, which are significantly more expensive to produce.

In the gaming industry, realistic fluid dynamics are essential to enhance immersion; being able to interact with the ocean in a beach environment is a lot more engaging than the common use of invisible barriers that prevent interaction with the water in many games. A fantastic example of excellent water physics is Far Cry 6 [7], as it looks very realistic and is affected by external forces from objects such as vehicles and people swimming.

However, affordable options for advanced fluid simulation are quite difficult to come by for indie developers, who do not have the same funding as triple A companies like Ubisoft. As a result, the motivation for this project is to produce a free, lightweight simulation for all types of fluid (not just water). It will also aim to maintain a high performance, so it may be seamlessly implemented into an existing video game with minimal performance loss. This is essential in the gaming industry as lag and low frame rates hinder the overall gameplay experience.

1.2 Aim

The aim of this dissertation is to utilise the Navier-Stokes equations and Smoothed Particle Hydrodynamics to produce an accurate fluid simulation. The performance (specifically frame rate) of the simulation will be measured and there will be attempts to optimise it. The resulting code should be a fluid flow video game asset that works out-of-the-box for Unity.

1.3 Objectives

1. **Explore Existing Fluid Simulations** - This dissertation will test and experiment with free and accessible simulations available on the internet. It will compare and identify issues with them from the perspective of a game developer searching for a tool to implement into their own game.

2. **Navier-Stokes and SPH** - This dissertation will explain what the Navier-Stokes equations are and describe two ways to utilise them in a fluid simulation.

3. **OpenGL and C++/GLFW** - This dissertation will briefly mention a graphics API that was used initially in this project and explain why it was chosen. It will then talk about the issues experienced with this API and why it was not utilised in the final project.

4. **Unity and Compute Shaders** - This dissertation will talk about the use of Unity and compute shaders to produce a fluid simulation that runs on the GPU.

5. **Frame Rate and Performance Investigations** - Throughout this dissertation, the main focus will be on improving the frame rate of the simulation. It will talk about what methods were used to improve the frame rate, such as reprogramming the simulation in shader languages. The overall goal will be to improve the performance of the simulation.

1.4 Changes

In the initial project proposal, it was stated that the simulation would be programmed in C++, using GLFW [8] to create windows and display the graphics created by the graphics API that I would be using, OpenGL [9].

However, later in development this decision was changed to utilising the Unity game engine [10] and compute shaders. Reasoning will be provided in the What was Done and How section.

Additionally, the project proposal stated that the simulation would be first created in two dimensions and later upgraded to three. It was decided to skip this step as the development process was mostly the same, the only difference being whether the vectors contained two or three values. This was to save development time and allow more focus on optimisation.

1.5 Dissertation Structure

1.5.1 Time Structure

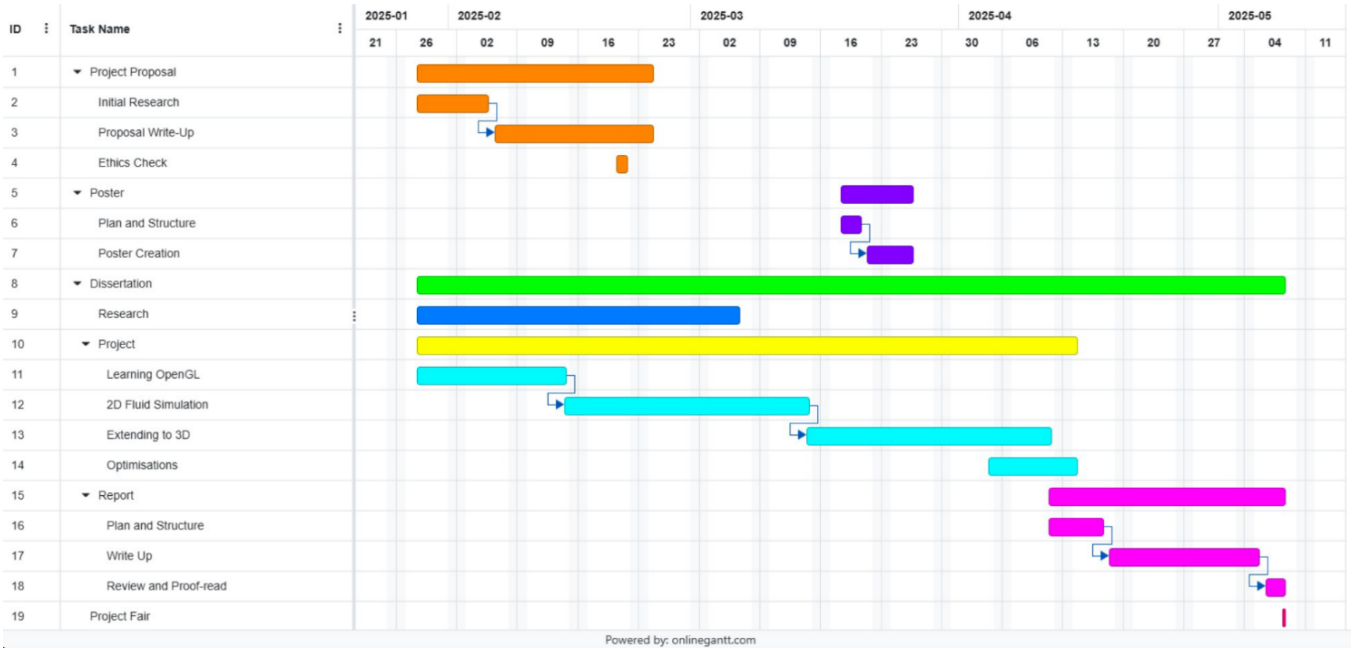

Figure 1: Gantt chart of project timeline [2]

1.5.2 Dissertation Outline

- **Introduction**
  An introduction to the dissertation, outlining the motivation, aim, objectives and overall project structure.

  - Motivation
  - Aim
  - Objectives
  - Changes
  - Structure

- **Background Review**
  A breakdown of research performed before and during the project.
  - Software Tools
  - Mathematics and Physics
  - Shader Language
  - Optimisation Tools

- **What was Done and How**
  An explanation of how the simulation was implemented, including the methods learned and processes involved.
  - OpenGL
  - Unity
  - Compute Shaders
  - GPU Instancing
  - Optimisations

- **Results and Evaluation**
  A complete analysis of frame rate improvements over the course of the project and results obtained.
  - Frame Rate
  - Testing

- **Conclusion**
  A conclusion that describes fulfilment of objectives, what was learned and future work.
  - Aims and Objectives

- Achievements
- What was Learned
- Future Work

2 Background Review

This section shows the research made on this dissertation. It talks about relevant background resources that were used for different parts of the development process.

2.1 Strategy

There are four main components of this project:

- Exploring existing fluid simulations
- Which software to use
- How fluid simulations work
- How to optimise this project

All four of these sections are integral in achieving a functional, lightweight simulation and thus needed to be researched before beginning. Below is an outline of the papers, videos and other resources that were used to understand these topics.

2.2 Exploring existing fluid simulations

It is important to understand current existing options that are available and describe their benefits and limitations in the context of video games.

2.2.1 WebGL Implementations

"WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. WebGL is fully integrated with other web standards, allowing GPU-accelerated usage of physics, image processing, and effects in the HTML canvas". [11]

There are a lot of fantastic WebGL implementations of fluid motion that are easily accessible due to their availability in a web browser. An example is the one developed by PavelDoGreat on GitHub.

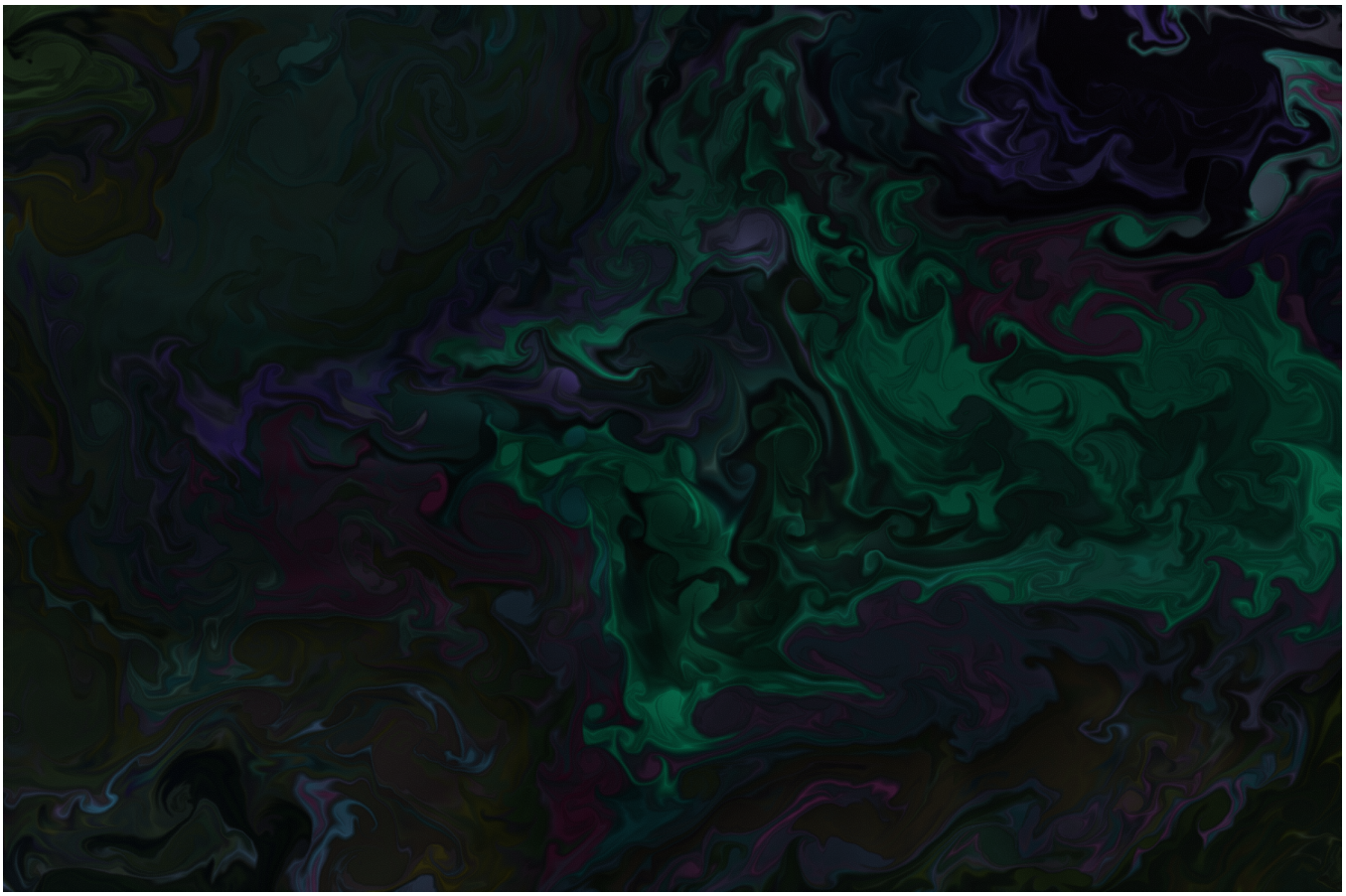

Figure 2: WebGL Fluid Simulation by PavelDoGreat on GitHub [3]

This simulation showcases a gaseous fluid made up of different, randomly-generated colours. They appear to mostly be for aesthetic purposes. The screen begins completely black and requires the user to input an external force by left-clicking and moving the cursor across the screen. There is a control panel on the top right corner, which allows the user to change various settings, such as density diffusion and vorticity. This is a great visual aid for educational purposes as it allows the user to see the visual impacts that these variables have on the fluid in real time. As an educational tool, this is

an excellent visual aid; the immediate feedback and colorful representation make it easy to understand complex fluid dynamics concepts. In the context of video games, however, the colours are a bit unnecessary. This can be easily rectified if someone were to use it in their game, but they would need to make a number of modifications to the existing code to make it independent of user input (for the more common uses of fluids in games).

Here is another example of a WebGL-based fluid simulation available for free online, developed by Grant Kot.

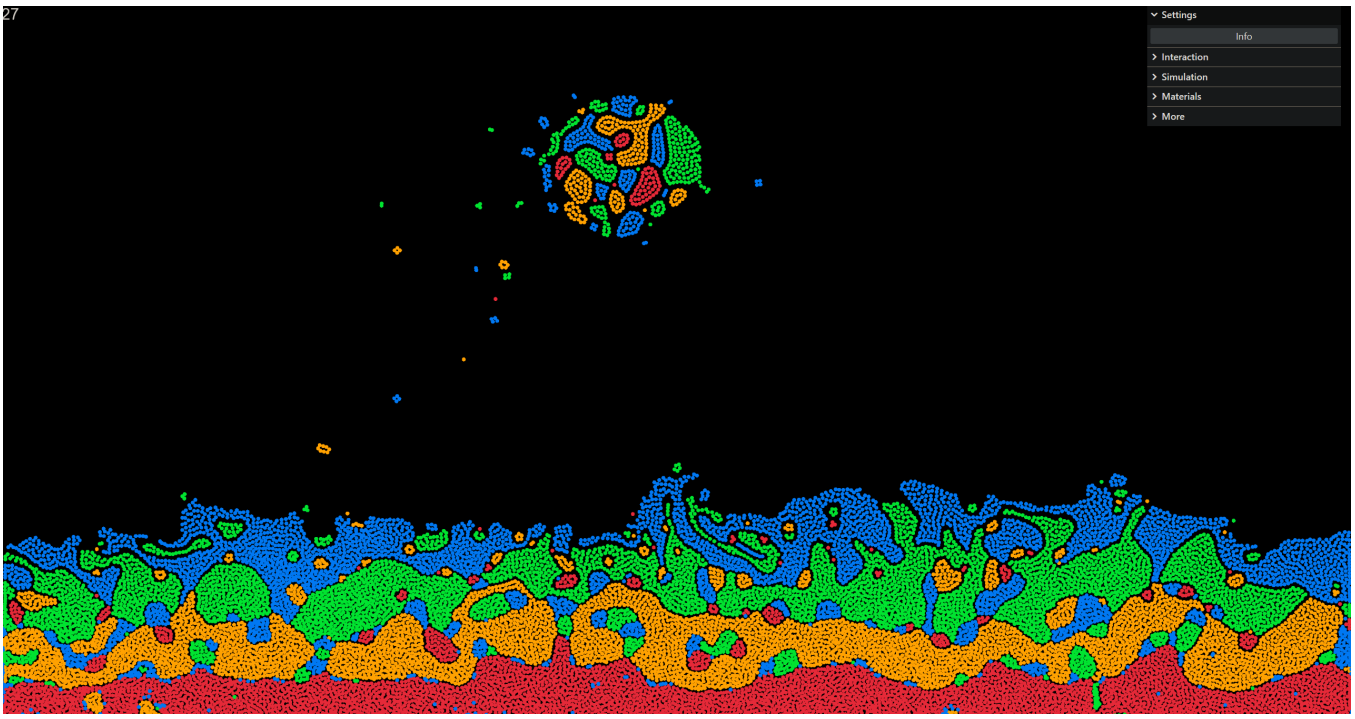

Figure 3: WebGL Fluid Simulation by Grant Kot. [4]

This simulation excels in visualising fluid interactions, particularly between liquids of varying masses - represented by four distinct colors. You can use the mouse and different keybinds to manipulate the fluid with a range of forces, such as lifting a portion of it up (as shown in figure 3), or creating dynamic collision objects that follow the cursor. It behaves with realistic motion that is driven by gravity and internal pressure and user interaction is only optional.

Conversely, there is a performance issue with this project. During testing, the simulation peaked at around fifty frames per second. This is completely fine when exploring the fluid in a web browser or for educational demos, but becomes a much larger issue when using it as a video game asset, as video games prefer to be consistently above 60 frames per second - to maintain responsiveness and overall gameplay fluidity. Further optimisations would be required of this simulation in order to utilise it in a video game environment.

WebGL simulations, however, work exceptionally in very simple browser games. Here is an example called "Interplanetary Postal Service" [5], which was created by Sebastian Macke, for a coding competition with a strict 13KB size limit.

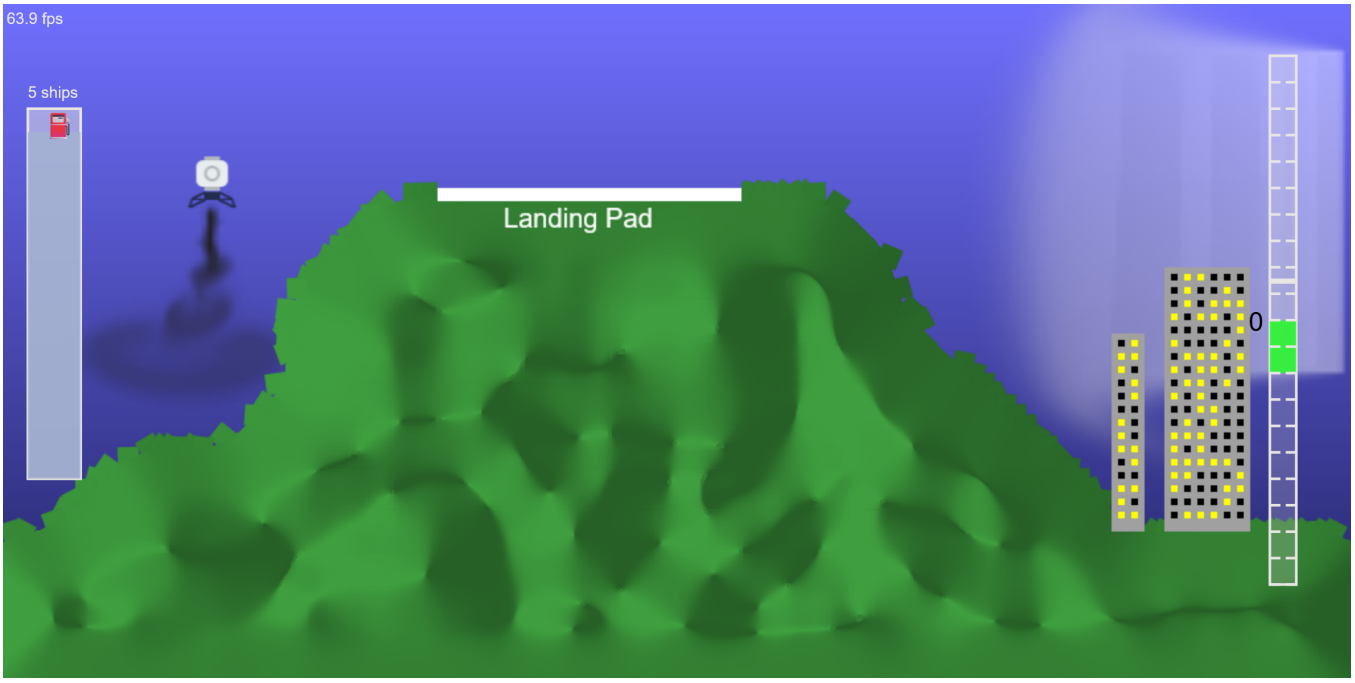

Figure 4: Interplanetary Postal Service by Sebastian Macke [5]

As the game's core mechanics are very simple (use WASD to power the post lander on to the landing pads), there is very little need for optimisations. This allows it to run consistently at over 60 frames per second. This demonstrates how WebGL-based simulations can deliver high performance and smooth interactivity when used for lightweight, focused experiences - which most video games aren't.

While all of these examples are incredibly impressive, they would not function well in more complicated, non-web-based video games. The first two are built as standalone browser-based visualisations for the purposes of aesthetics and education. Although all of these can be rewritten and ported to OpenGL-based games, there is a lot of upfront work required to do so; they do not work out-of-the-box. They are also only two-dimensional, which means further modifications would be required necessary to adapt them for 3D environments. Furthermore, for games written using pre-existing game engines such as Unity, most of the existing JavaScript simulation code would have to be rewritten into a language that is compatible with the engine, such as C#. One of the aims of this dissertation is to produce a 3D asset that is ready to use in Unity without requiring extensive modifications.

3 What Was Done and How

4 Results and Evaluation

4.1 Results

4.2 Evaluation

5 Conclusion

5.1 Aims and Objectives

5.2 What Was Learned

5.3 Future Work

6 References

References

[1] J. Stam, "Real-time fluid dynamics for games," https://damassets.autodesk.net/content/dam/autodesk/www/autodesk-reasearch/Publications/pdf/realtime-fluid-dynamics-for.pdf, 2003, accessed on February 18, 2025.

[2] T. Sze and Y. Liu, "Online gantt," Online Software: https://www.onlinegantt.com/#/gantt, accessed on February 19, 2025.

[3] PavelDoGreat, https://paveldogreat.github.io/WebGL-Fluid-Simulation/, accessed on January 28, 2025.

[4] G. Kot, "Viscoelastic webgl fluid simulation," https://grantkot.com/ll/, accessed on January 29, 2025.

[5] S. Macke, "Interplanetary postal service," https://play.js13kgames.com/interplanetary-postal-service/, accessed on April 26, 2025.

[6] B. R. Insights, "Computational fluid dynamics (CFD) market size, share, growth, and industry analysis, by type (personal and commercial), by application (aerospace and defense, automotive industry, electrical and electronics, and others), regional insights and forecast from 2025 to 2033," https://www.businessresearchinsights.com/market-reports/computational-fluid-dynamics-cfd-market-111787, 2025, accessed on February 18, 2025.

[7] Ubisoft, "Far cry 6," Video Game: https://www.ubisoft.com/en-gb/game/far-cry/far-cry-6, 2021, accessed on April 25, 2025.

[8] "GLFW," Window creation API: https://www.glfw.org/, accessed on February 1, 2025.

[9] K. Group, "OpenGL," Graphics API: https://www.opengl.org/, accessed on February 1, 2025.

[10] U. Technologies, “Unity,” Game Engine: https://unity.com/, accessed on March 1, 2025.

[11] Wikipedia, “Webgl,” https://en.wikipedia.org/wiki/WebGL, accessed on April 25, 2025.

7 Appendices