<u>Creating and Optimising a Fluid Simulation Using the Navier-Stokes Equations For Use in Video Games</u>

<u>Context</u>

Introduction

Fluid dynamics are observed in all aspects of daily life and in research. As a result, it is crucial to be able to simulate and understand the behaviour of all fluids, including liquids and gases. There are many industries that rely on Computational Fluid Dynamics (CFD), such as aerospace, which utilizes CFD for aerodynamic analysis, aircraft design optimization and thermal management, to name a few examples. Due to the large amount of interest in CFD, the global market is expected to increase by nearly triple its current value by 2033 [1]. Furthermore, they are essential for reducing costs in these industries as they reduce the need for physical prototypes, which can be significantly more expensive to produce [2].

The Problem

In the gaming industry, fluid simulations are essential for creating realistic water in most outdoor-based games. Although there are a lot of tutorials for water shaders online [3], most only create visually realistic water without simulating fluid flow or interactivity. Realistic fluid dynamics enhance immersion, allowing players to interact with water, such as picking up a bucket or splashing in the sea. This increases player engagement, as opposed to the common use of invisible barriers that prevent interaction with water in many games. A fantastic example of excellent water physics is Far Cry 6 [4], as the water is incredibly realistic and interacts with objects around it as well as forces applied to it by vehicles [5]. However, for most indie game developers, affordable options for advanced fluid simulation are hard to come by, as the cheaper options are still quite expensive [6]. Additionally, most of the options available do not consider fluid viscosity, which is important considering the variety of fluids used in games beyond water.

The Rationale

In my project, I aim to create a free, accessible simulation for smaller game companies to utilize, that can simulate fluids with high viscosity such as slime or syrup, using the Navier-Stokes equations [7] to do so. It will be an extension of previous research conducted by Jos Stam [8] that considers the third dimension. I will also aim to achieve a consistent high performance while the simulation is running, as it is essential for video game assets to not hinder the gameplay experience with lag or low frame rates.

<u>Key Background Sources</u>

<u>An Introduction to the Mathematical Theory of the Navier-Stokes Equations, Galdi, G.P. (2016) [7]</u>

Description

This book goes into detail about the Navier-Stokes equations and provides a sufficient introduction to understanding and using them.

Relevance

This book is essential for understanding exactly how the Navier-Stokes equations work on a fundamental level. It is incredibly important to know exactly how they work, instead of relying on code from other sources. This is in order to improve and optimize my own implementation to the best of my ability.

<u>Real-Time Fluid Dynamics for Games, Jos Stam (2003) [8]</u>

Description

This paper presents a simple and fast implementation of a fluid dynamics simulation. It utilizes the Navier-Stokes equations and provides a complete C code implementation for two dimensions.

Relevance

As my project will also be utilizing the Navier-Stokes equations, this paper is a great starting point for my own fluid simulation implementation. It helps bridge the distance between the mathematical knowledge required to understand the Navier-Stokes equations and how I would actually implement it graphically and programmatically.

Real-Time GPU-based SPH Fluid Simulation Using Vulkan and OpenGL Compute Shaders [9]

Description

A paper that investigates the performance of OpenGL compared to Vulkan, using smoothed particle hydrodynamics. An alternative method for simulating a fluid using the Navier-Stokes equations.

Relevance

As a primary concern for this project is performance, it is important to know which graphics API has better frame rates for my intended goals. Additionally, it uses an alternative method to simulating fluids which will be useful to compare with and use ideas from for my own implementation.

Advances in Fluid Mechanics, Rahman, M. and Bebbia, C.A. (2022) [10]

Description

This book contains a lot of different papers about fluid modelling and simulations in multiple different branches. It "aims to publish new research work that enhances the prediction and understanding of fluid mechanics and balances from academic theory to practical applications".

Relevance

These papers emphasize the importance of fluid simulations and their accessibility to all fields of research. They all provide different alternative methods to simulations and give real-world examples of how important and useful the Navier-Stokes equations are for viscosity calculations and thick fluid flows.

Learn OpenGL website [11]

Description

A website that gives an in-depth tutorial for getting started with OpenGL. It teaches about the rendering pipeline, basic vertex and fragment shaders by demonstrating the steps to produce a triangle on to the screen using OpenGL.

Relevance

As I will be using OpenGL in my coursework, it is important to understand how the graphics API works and to be able to produce graphics on to the screen with it.

Aim and Objectives

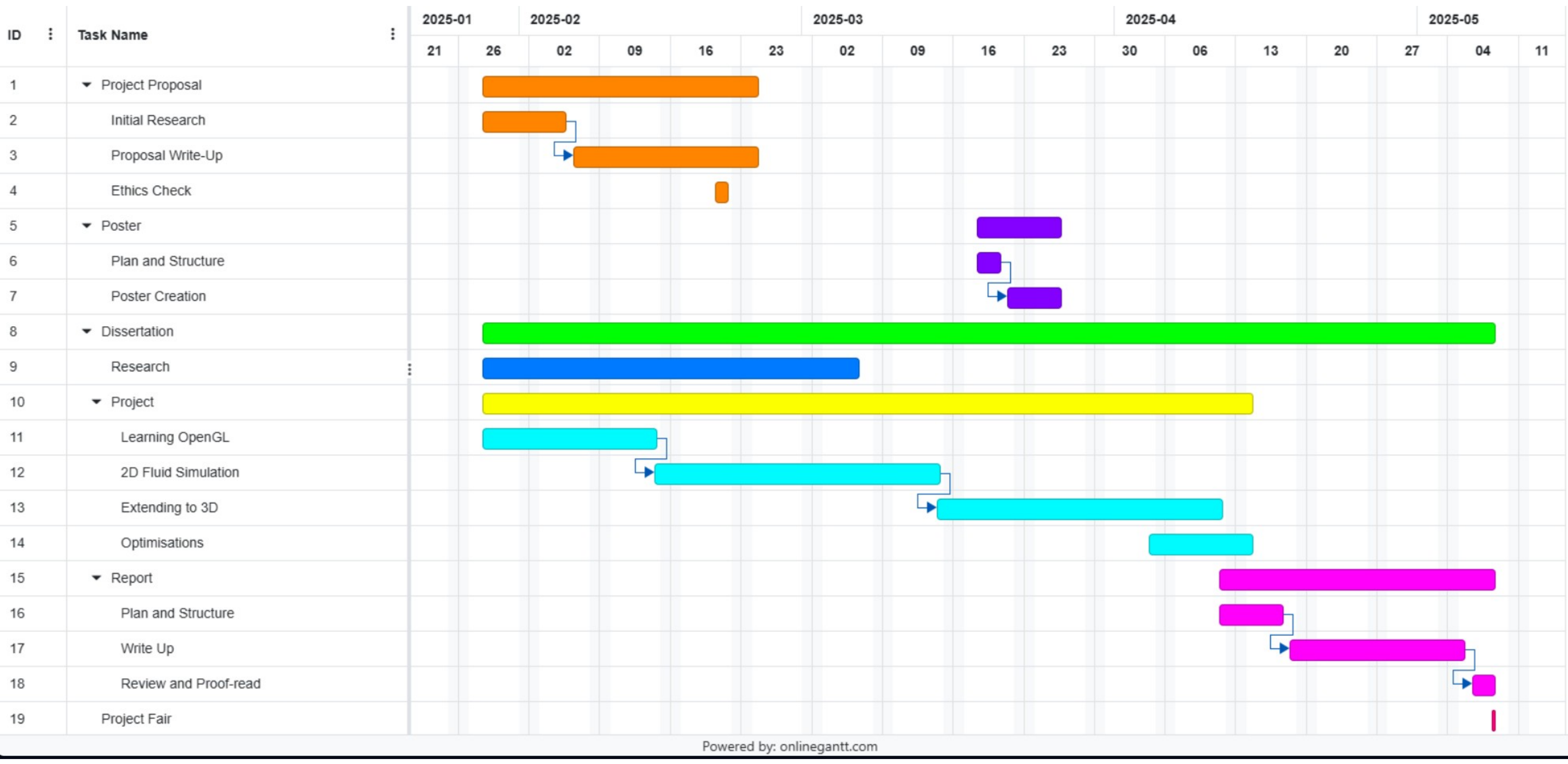
Aim

The aim of this project is to use the Navier-Stokes equations and OpenGL to produce an accurate fluid simulation for viscous fluids. I intend to measure the performance (specifically frame rate) of the simulation and attempt to optimize it. I will test how the viscosity and number of particles of the simulation affects the frame rate and attempt to reduce their impact on the performance. Furthermore, I will test my simulation in a game engine and assess qualitatively how well it works in a game environment.

Objectives

1. Explore existing fluid simulations – In my background reading, I will test and experiment with free and accessible simulations available on the internet. I 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. Understand and explain the importance of the Navier-Stokes equations for fluid simulations and viscosity - There are a lot of fluid simulations that use the Navier-Stokes equations for their calculations. I will explain why they're so fundamental in fluid mechanics, as well as describe and explain what the calculations involving them actually represent and produce.
3. Learn and utilize OpenGL with C++ and GLFW – A graphics API is required to produce graphics, such as a fluid flow simulation. I will use OpenGL and discuss the performance differences shown in a paper [9] to justify my API choice. I will also explain and justify my language choice.
4. Develop a basic 2D fluid simulation – A basic simulation is an important starting point; I will create my own using knowledge I have acquired from research as well as from existing fluid simulation code. I will use papers I have read in my initial research [8] to do this. I will measure how the frame rate is affected over time when more detail is added over time.
5. Extend the fluid simulation to three dimensions – In order to be used in 3D games, the simulation must work in 3D. As a result, I will use previous research and my 2D solution to produce a three-dimensional simulation.
6. Test what parts of the simulation impact the frame rate the most – Instead of testing across different hardware, I will specifically check what elements within the simulation produce different frame times and how much they affect the overall frame rate compared to the rest of the program. I will also develop the simulation with these tests in mind and make sure they satisfy the criteria to be useful in a video game.
7. Improve performance of the simulation – The simulation must not be too heavy on the GPU when used in a video game in order to ensure that it runs smoothly. As a result, I will be constantly tracking the frame rate of my simulation and attempting to increase it wherever possible. I will research optimization techniques and refine my own code when necessary to make sure it runs smoothly in a 3D video game. This is an extension of objective 6 as I will be using the information learned from that and focus on the most frame intensive elements of the program.

<u>Planning</u>



ID	Task Name
1	Project Proposal
2	Initial Research
3	Proposal Write-Up
4	Ethics Check
5	Poster
6	Plan and Structure
7	Poster Creation
8	Dissertation
9	Research
10	Project
11	Learning OpenGL
12	2D Fluid Simulation
13	Extending to 3D
14	Optimisations
15	Report
16	Plan and Structure
17	Write Up
18	Review and Proof-read
19	Project Fair

Gantt chart created using OnlineGantt.com [12]

Brief Explanation of the Plan

Each section of the dissertation ends on their deadline, and thus I have not included the deadlines explicitly in the diagram. This is because it is important to use all the time I have for each section to the best of my ability. The arrows represent which sections are dependent on each other; I cannot start one without having done the section that the arrow is connected to. It is important to be constantly working on the project during the shorter deadlines such as the proposal and the poster, as it is a hefty task that will require a lot of time; the poster will not need an entire month to complete. I have included the project proposal and initial research on the diagram as, although most of this has already been completed, I felt it necessary to show as they are just as important as the rest of the dissertation. I plan on doing the write-up after the simulation is made, as this will be the point where I will understand everything the best and can fully focus on making sure that it is as coherent and well-explained as possible (as it will be the only thing left to focus on). I will be using an agile development approach.

Risks

1. Scope may be too large – Learning an entire new API and advanced graphics from scratch may be too difficult in the span of time we have been given. In the worst case scenario, I will mitigate this issue by swapping to an already established game engine if I find myself to be too far behind to feasibly continue with OpenGL. In order to manage the scope, I will be using an agile development approach with frequent supervisory meetings. This will help me as my supervisor will be able to set realistic goals every week so I can manage my time better and see if the scope is too large (by being unable to complete the tasks set for that week).
2. Fully understanding the mathematics required – The Navier-Stokes equations may seem simple initially, but there is a lot to consider in order to fully understand them. As I have a background in mathematics, I believe this won't be too much of an issue for me. I believe the worst case scenario for this problem is that I may end up spending longer time than planned to fully understand. This can be mitigated by learning how to use OpenGL in tandem with learning the mathematics involved. This will allow me to switch topics when I feel overwhelmed and come back later with a refreshed outlook.
3. Full write-up at the end of the project – Although I will understand everything that will be explained in the report up to this point, I may forget key pieces of information that I would have remembered if I wrote it at the time of learning. I will mitigate this by consistently taking notes about what the report will include while I am creating the project.
4. Balancing realism and performance – It is difficult to maintain realism while keeping performance high. Since this will be a persistent simulation in a game, it is important to prioritize performance. As a result, I will focus on the most important interactions and optimize the calculations for these. Additionally, optimizations may take a lot more than the currently allocated time. I will attempt to do the most that I can in the permitted time and may take a few more days than planned, but it is important to acknowledge the significance of the report and may leave further work until the end of the write-up if time allows it.
5. Loss of data – In order to prevent data being lost from my computer or another device, it is important to make sure I have an external backup located elsewhere. To achieve this, all of my work will be saved to a private GitHub repository, using git.

References

[1] www.businessresearchinsights.com. (n.d.). _Computational Fluid Dynamics (CFD) Market Size - Forecast To 2031_. [online] Available at: https://www.businessresearchinsights.com/market-reports/computational-fluid-dynamics-cfd-market-111787 [Accessed 18 Feb. 2025].
[2] Quadco Engineering (n.d.). _Advantages and Disadvantages of CFD | Quadco Engineering_. [online] www.quadco.engineering. Available at: https://www.quadco.engineering/en/know-how/cfd-advantages-and-disadvantages.htm [Accessed 18 Feb. 2025].
[3] Pirani Dev (2024). _Creating Realistic Water and Oceans in Unity 6_. [online] YouTube. Available at: https://www.youtube.com/watch?v=-rxqKHb_4gc [Accessed 18 Feb. 2025].
[4] _Far Cry 6._ (2021). PC [game]. Toronto, USA: Ubisoft.
[5] Dan Allen Gaming (2021). _Incredible Water Physics and Details in Far Cry 6_. [online] YouTube. Available at: https://www.youtube.com/watch?v=BNispn46eAw [Accessed 18 Feb. 2025].
[6] @UnityAssetStore. (2019). _Obi Fluid_. [online] Available at: https://assetstore.unity.com/packages/tools/physics/obi-fluid-63067 [Accessed 18 Feb. 2025].
[7] Galdi, G.P. (2016). _An Introduction to the Mathematical Theory of the Navier-Stokes Equations_. Springer.
[8] Stam, J. (2003). Real-Time Fluid Dynamics for Games. [online] Available at: https://www.dgp.toronto.edu/public_user/stam/reality/Research/pdf/GDC03.pdf [Accessed 18 Feb. 2025].
[9] Gunadi, S.I. and Pujianto Yugopuspito (2018). Real-Time GPU-based SPH Fluid Simulation Using Vulkan and OpenGL Compute Shaders. _2019 5th International Conference on Science and Technology (ICST)_, pp.1–6. doi:https://doi.org/10.1109/icstc.2018.8528699.
[10] Rahman, M. and Brebbia, C.A. (2022). _Advances in Fluid Mechanics_. _Forum for interdisciplinary mathematics_. Springer Nature. doi:https://doi.org/10.1007/978-981-19-1438-6.
[11] de Vries, J. (2019). _Learn OpenGL, extensive tutorial resource for learning Modern OpenGL_. [online] Learnopengl.com. Available at: https://learnopengl.com/ [Accessed 1 Feb. 2025]
[12] Online Gantt (2024). _Free Online Gantt Chart Software_. [online] www.onlinegantt.com. Available at: https://www.onlinegantt.com/#/gantt [Accessed 19 Feb. 2025]