1 Evaluation and Final Remarks

1.1 Final artefact outcome

The first success criteria was to research SPH and popular alternative methods of fluid simulation. From my research review, I think it is evident that this criteria was fulfilled. My initial introduction to Lague [?] and his sources led me to discover SPH and how CFD is handled within industry. I initially used Youtube to gain a high-level understanding of SPH but quickly realised the amount of detail which videos glossed over when explaining SPH. Sebastian Lague's sources on his github page prompted me to use Google Scholar to search for Müller's [?] introductory work. This was naturally the most useful source in terms of theoretical understanding of SPH and therefore extremely useful when developing the theoretical model. Koschier [?] and Clavet [?] provided more algorithmic approaches and were more useful in the development part of the project. Per my Research review, I have also considered the alternative Eulerian approach to simulating fluids. Although more precise, Bridson [?] stated it is less suitable for interactive implementations as the resizing the window would reshuffle the grids which this grid-based approach relies upon.

The second objective on my sucess criteria is to develop a simulation which solves the Navier-Stokes equations and can be classed as a fluid. Referring to the figures in Appendix ??, it is evident the simulation solves every term of the Navier-Stokes equation. Namely, moving down the pressure gradient, constant density, viscosity and gravity. Interestingly, the pressure multipler ends up controlling how gas-like or fluid-like the simulation is. Clavet [?] mention that the pressure multiplier acts as a stiffness constant and this is obvious in my implementation as a higher pressure multiplier makes the simulation more stiff, portraying more liquid-like behaviour in my SPH implementation as particles exert more intermolecular forces. A lower pressure multiplier has smaller intermolecular forces and therefore the particles are more gasseous. Interestingly, an extremely high pressure multiplier leads to the simulation behaving more like a soft bodied solid, like jelly. This is similar to the outcome mentioned by Clavet [?]. The simulation is highly sucessful at striving to obtain a constant density. This was best evident upon adding mouse forces to the simulation because a repulsive force creates additional space, which the particles rush to fill once the force has been removed. This is similar to the particles moving down the pressure gradient when the window is resized to add additional space. The particles end up in a stable position with no velocity in an orderly fashion.

Viscosity and gravity has a lower level of success than initially anticipated. Although viscosity does contribute to the particles velocity ending up being the same as its neighbours, this is better achieved by a larger pressure multiplier. Gravity is also not well incorporated within the simulation. The best results are achieved by imagining the simulation is looking at a sample of submerged fluid or a top-down view of a container containing this fluid with gravity set to zero. If I attempt to simulate the fluid as water within a glass with gravity, for the same settings, the fluid ends up being very chaotic as seen in the Appendix ?? Figure ??. Although still semi-realistic, the problem is the particles at the bottom seem to gain kinetic energy, as if they are boiling or are representing a heavy-gas, while particles at the top seem unphased. One way around this is to reduce the particle number. This introduces its own problem though. The simulation becomes less accurate for a lower particle number. The aim is to be able to render as many particles as possible and in order to have fluid in a glass simulation, I have to half my particle number to 800 particles. The issue created here is the surface at the top does not end up smooth, but ends up becoming jagged as seen in Figure **??**.

It is fair to say the animation runs at with minimal lag and resource wastage. During early stages of development, I did have to ensure that all my calculations were being done in as few loops as possible. Currently, my programs has 2 loops, one nested within another. This, as I found during my development, was the limit for an interactive simulation. The addition of further loops caused the program to start slowing down and become unresponsive. With this in mind, I limited my calculations to this structure. Upon adding mouse forces, the program also started to slow down and this was due to how SFML handles mouse button down events. I did find that interactive elements worked best implemented within the events loop rather than the main simulation loop, which retrospectively is expected as those events are designed to be handled within that loop.

As for interactive elements, it is evident throughout Appendix **??** that window resizing has worked flawlessly since the start. As for mouse forces and the linear colour algorithm, these were elements I was able to add over the summer due to the extension of the project deadline. Their outcome has been extremely sucessful as the mouse forces have helped me test the simulation response to user interaction. SPH is designed for interaction and this was evident upon implementing the mouse forces. Before the extension of the deadline, I struggled to change fluid settings due to the lack of sliders with key fluid variables. Over the summer, I made sure to implement sliders seen in Figure **??**. These allowed me to change variables to values whilst the simulation is running, saving me time spent debugging or recompiling otherwise. They also help demonstrate concepts such as constant density and forcing fluids to move down pressure gradients. I found this useful, especially during my presentation, as they helped visualise concepts to the audience. The colour also adds a layer of aesthetic investigation and intrigue to the user. Any additional time over the summer was spent perfecting the interactive elements to ensure the simulation responds well to any user interaction, hitting the last two points of my sucess criteria.

1.2 Improvements

A major improvement which I would make to my artefact would be adding optimisations to render more particles. This would solve most issues with my project so far. As stated in the last section, the effects of viscosity are not immediately obvious but a larger particle number would mean the smoothing radius of more particles overlap with each other, increasing the influence experienced by each particle. As per my code, this would increase the effect of viscosity overall leading to a noticeable effect on screen. When it comes to gravity, seen in Figure **??**, a larger particle number would even out the jagged surface which is unrealistic in practice. This would increase the realism of the simulation as a liquid in a glass. Attempting to improve viscosity further would add a surface tension-like effect as well, as seen in Lague's [?] implementation, leading to a layer of particles attracted to each other.

One method of optimising my code to be able to add more particles would be to develop the simulation in an existing physics engine like Unity or Unreal Engine 5. These game engines are designed to be able to run simulation physics to a very high standard and contain proprietary optimisation techniques. Moreover they would help with compute-heavy calculations such as distance calculations and standardise units into kilograms or meters. Currently, my program runs on a pixel-to-pixel basis, where one unit of distance is a pixel. This means the dimension analysis [1] of my units for pressure or density are not correct. Physics engines often ease such aspects of computer graphics to allow developers to focus on more complex simulation mechanics.

One final improvement I would make to my artefact would be to

1.3 Skills

1.4 Presentation

1.5 Final Takeaways

[1] Analysis of base units of a physical quantity