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 two 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. Interestingly, container pressure or surface tension effect is observed within the animation, refer to Figure **??**. On the simulation border, we see an unusual adhesion the particles have to the window border, causing a repelling force. In reality, this property arises from a known disadvantage of SPH when handling boundary cases. This effect will appear to be smoothed out with a larger number of particles.

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.

Another improvement would be to add the third dimension and complete the simulation in three dimensions. Although this would involve 3D linear algebra to consider the positions of objects and how they are seen at different distances on screen, it would surprisingly not change much in terms of the mathematics for SPH except change 2D vectors into 3D vectors. All of the papers I have used for SPH work perfectly in 3D. With now an intuitive understanding of SPH in 2D, a viable extension would be to implement an SPH solver in 3D. Some elements of interactivity may be sacrificed such as the window resizing, and be converted to a bounding box within the simulation to interact with instead.

One final improvement I would make to my artefact would be to turn the simulation into a compute shader. Currently, my program runs on the CPU. Turning it into a compute shader would involve major changes to the code such that it can be run on the GPU. Although this would involve its own research and would be the equivalent of undertaking another EPQ, due to my lack of familiarity with parallel processing, it is certainly an improvement I would consider making to understand more about GPU mechanics and how it can be utilised more effectively for simulation. Parallel processing power of the GPU would be put to test when using the fluid simulation for modelling fluid flow around or within objects for more accurate and realistic results, an extension worth considering for more realistic modelling.

1.3 Skills

Reading and understanding academic publishings and research papers in a Mathematics and/or Computer Science background was initially challenging. Although the formal

[1] Analysis of base units of a physical quantity

language posed a small barrier, the mathematical notation was more nuanced and took longer to parse. Most papers included a key which set out the notation and I found myself referring to it constantly when first researching SPH. This inspired me to create a definitions section within my write-up to help everybody understand the language used in this paper more thoroughly with minimal ambiguity. Reading such published work is a skill I feel much more confident with now and am proud to have learned as my work at university will undoubtedly use this to a great extent.

Being able to create my theoretical model based upon existing research is another skill I have clearly accomplished. I am able to justify every section within my theoretical model and have sucessfully applied the research to customise certain sections of SPH, such as the proposed compute-power heavy smoothing kernel versus my implemented gentle alternative.

Typing up my write-up in LaTeX, a document formatting tool which I have used to compile my pdf, has eased the process of integrating mathematical formulae into published work. From the theoretical model and development sections of this write-up, it is evident this has been very sucessful. I can say with experience this would have been arduous on alternative document formatting tools such as Word or Onenote. I plan to share this write-up on GitHub and confidently submit it to open-access journals. LaTeX has enabled me to structure the document in a format that feels on par with the research papers I've studied, enhancing both its clarity and professionalism. I have extended my knowledge of LaTeX by being able to add plots, seen in the theoretical model, code, as seen in development and images, as seen in Appendix **??**, better emphasising my points and improving legibility. Furthermore, LaTeX is preferred at university for mathematics and is used by students to take digital notes as well as submit assignments with instead of easily misplaced or misunderstood handwritten work. This skill will be invaluable for me at university.

A big motive of taking this project was also to learn C++, a new language with which I was not initially familiar. C++ is often the preferred language due to the vast amount of support it has in many software disciplines but especially due to its fast execution time which is crucial with simulation. Learning C++ was time consuming at the start, I found myself constantly referring to W3schools [**?**], a website with tutorials to learn new concepts within programming languages. This would be for simple tasks such as outputting values to the screen or to understand the cause of simple syntax errors. I also spent a lot of time on the SFML documentation site [**?**] to understand how to setup and use SFML and ImGui. Due to my experience with Python [2], I was perhaps able to draw parallels quicker than someone learning the language from scratch. My familiarity with the testing and debugging process fastracked error resolving overall. Now after finishing my artefact, I feel more confident using C++ on a day-to-day basis in my Computing A-level at school or eventually at university for projects.

In order to manage my code and allow data synchonisation between my laptop and PC, I used Github. Github is a cloud based service which allows developers to store their source code to ease with development version control. It is a framework built on

[2] Another programming language

top of git, a version control software. Github provides a graphical interface allowing me to more easily manage coding over multiple devices as well as publish code to the wider audience for everybody to use. It thoroughly streamlined synchonisation and allowed me to save time when trying to work on the project at school and work. It is a tool widely used in the IT industry and a tool I can confidently use in a corporate environment.

One of the biggest skills I have managed to develop is my time management. At the start of the EPQ, I consistently spent two hours weekly on Thursdays and Fridays. I found it was easier researching SPH at school and developing at home where I had access to my PC. Adhering to this schedule meant the development of density and pressure forces was very quick due to the well conducted research and well planned-out theoretical model which outlined the main mathematical elements to be coded. Interestingly, the half of my timescale is extremely accurate but the second half is not very accurate at all. This is likely due to additional commitments during the latter parts of Year 12 such as mocks as well as the extension of the deadline. This allowed me to refine the artefact to include viscosity and add interactive elements other than window resizing, resulting in a more polished implementation overall. Learning to adapt to sudden changes in timescales and change the theoretical model appropriately is a skill undertaking an EPQ has taught me.

1.4 Presentation

To prepare for my presentation, I created a powerpoint presentation, refer to and structured it to explain For my presentation, I decided on displaying my artefact by setting up my PC in school. Although this was time consuming and inconvenient, I felt that having the simulation on display helped attract attention especially due to its aesthetics with the linear colour algorithm and interactions with the mouse. Having the artefact present also helped illustrate my points on the inner workings of SPH when presenting with the slideshow on my laptop. Showing how different sections of SPH solve the Navier-Stokes equations, such as resizing the window to show fluids moving down the pressure gradient, provided a better intuitive sense of my artefact and many complemented me for this visualisation. Generally, I felt my pace when explaining sections of the simulation was good and I was able to field questions when asked. The most common included "Why did you choose this project", to which I responded with how CFD perfectly overlaps with my A level subjects and how it is a field I intend to explore further. Some SPH and artefact specific questions included "Why does the pressure multipler control gas or liquid behaviour?", as this slider seemed to interest most people. I responded by explaining how the pressure multiplier controls the stiffness of the fluid and quantifies intermolecular forces between these particles. Another question was "Why do the particles stick to the border and repel particles within the window" to which I honestly answered by elaborating on the known disadvantage of SPH when handling boundary and edge cases. The biggest success was observed when people interacted with the fluid using the mouse forces or sliders themselves. This was anticipated by me when the project deadline was extended and I decided on adding interactive elements that it would be put to most use during the presentation. Refer to Appendix **??** for the presentation slides.

Word count: 10172