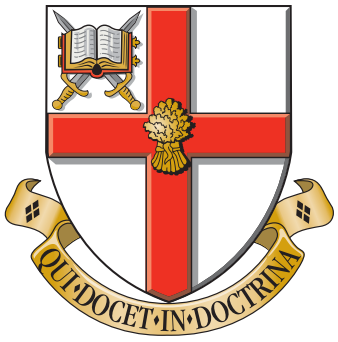


University of Chester

Department of Computer Science
Faculty of Science and Engineering

Interactive Three-Dimensional Simulation and Visualisation of Real Time Blood Flow in Vascular Networks

Thesis submitted in accordance with the requirements of the
University of Chester for the degree of
Doctor of Philosophy
by

Mark Ian Holland

Supervisors: Dr S. R. Pop and Prof. N. W. John

May 2020

Declaration

The material being presented for examination is my own work and has not been submitted for an award of this or another HEI except in minor particulars which are explicitly noted in the body of the thesis. Where research pertaining to the thesis was undertaken collaboratively, the nature and extent of my individual contribution has been made explicit.

Mark Ian Holland

May 2020

Acknowledgements

Firstly, I would like to express my gratitude to my supervisors, Dr. Serban Pop and Prof. Nigel John, for their support and guidance throughout my research. Their loyalty and compassion has been greatly appreciated. I would also like to thank Andrew Muncey for suggesting me for the role.

Secondly, a special thanks to Dr Derek Gould for his enthusiastic and meaningful feedback that focused the design of the simulation application for clinical use.

Finally, I would like to thank my family and friends for their consideration and patience. With special thanks to Judith Holland, Joanna Meredith, and my colleagues, Peter Butcher and Thomas Day, for their support throughout.

Abstract

Interactive Three-Dimensional Simulation and Visualisation of Real Time Blood Flow in Vascular Networks

Mark Ian Holland

One of the challenges in cardiovascular disease management is the clinical decision-making process. When a clinician is dealing with complex and uncertain situations, the decision on whether or how to intervene is made based upon distinct information from diverse sources. There are several variables that can affect how the vascular system responds to treatment. These include: the extent of the damage and scarring, the efficiency of blood flow remodelling, and any associated pathology. Moreover, the effect of an intervention may lead to further unforeseen complications (e.g. another stenosis may be "hidden" further along the vessel). Currently, there is no tool for predicting or exploring such scenarios.

This thesis explores the development of a highly adaptive real-time simulation of blood flow that considers patient specific data and clinician interaction. The simulation should model blood realistically, accurately, and through complex vascular networks in real-time. Developing robust flow scenarios that can be incorporated into the decision and planning medical tool set. The focus will be on specific regions of the anatomy, where accuracy is of the utmost importance and the flow can develop into specific patterns, with the aim of better understanding their condition and predicting factors of their future evolution. Results from the validation of the simulation showed promising comparisons with the literature and demonstrated a viability for clinical use.

Contents

List of Figures

List of Tables

List of Algorithms

List of Source Code

Chapter 1

Introduction

1.1 Motivation

Cardiovascular diseases (CVD) continue to be the foremost cause of deaths worldwide [109], claiming an estimated 17.9 million lives each year [26]. In the UK, CVD is the leading factor in a quarter (25%) of all deaths, an average of 420 each day. It is estimated that the UK health care costs of heart and circulatory diseases is £9 billion each year, with an estimated cost to UK economy of £19 billion each year, this includes premature death, disability and informal costs [23]. There needs to be constant research and development into treating and preventing CVD, to reduce the suffering of those affected, and the costs incurred. Consequently, the United Nations have recognised CVD, and other chronic diseases as a major concern for global health and committed themselves to reducing the effects of these diseases [121]. CVD is an umbrella term to describe all diseases of the heart and circulation, from inherited conditions or complications from birth, to those that may develop later in life, such as heart failure and stroke. CVD pathology is often attributed to two main determining risk factors, medical (high systolic blood pressure, high total cholesterol,

high fasting plasma glucose, and high body mass index) and behavioural (diet, low physical activity, smoking and alcohol use) [199].

Coronary heart disease (CHD) is the most common type of CVD and is responsible for over 66,000 deaths in the UK each year [23]. CHD is primarily a result of atherosclerosis, the narrowing of blood vessels due to formation of fatty deposits called plaques, or atheroma, in the intima of arteries supplying vital organs. This focal atherosclerotic plaque is typically localised around bifurcations and other bends within the arterial tree structure [177]. The narrowing of blood vessels can lead to abnormal flow conditions where the patient's blood circulation is interrupted or disrupted, culminating in major acute cardiac events. When the coronary arteries suffer from a disruption in blood flow, the heart will receive insufficient oxygen to function i.e. myocardial infarction. Occasionally the blood flow is re-routed around a blockage through the nearby minor vessels, this process is known as collateral circulation. Collateral circulation around the heart may enable enough oxygenated blood to reach the cardiac tissue and allow it to survive, these cases have similar treatment options but within a more lenient time scale.

Treatment for atherosclerosis is varied, but most cases involve either bypass surgery or the less invasive angioplasty. The treatment is determined and delivered by clinical health care professionals with limited resources informing their decision-making process. Diagnostic decisions made are frequently based on rough estimates of outcomes, often derived from anatomic observations or extrapolation of physical laws [152]. The decision on whether or how to intervene in each case, is based on disparate pieces of information from different sources that need to be integrated to procure a successful outcome for the patient. These sources of information include but are not limited to coronary anatomy, results from physiological and imaging tests, the patient's history, and their current condition. Often these resources amount to low resolution medical scan images, greyscale visualisations [136], or physical models

fabricated to represent a patient's circulatory system [151]. In some cases, a patient may be eligible for coronary angiography, in which a catheter is inserted into a blood vessel, and using X-ray images as a guide, the catheter is navigated to the region containing the pathology. The angiogram is taken, and the contrast medium shows any blood vessels that are narrowed or blocked.

The information a clinician can gather from these methods is highly limited, and we believe it to be insufficient for complex cases with unknown behaviours. Due to their limitations, current diagnostic imaging and procedures may not provide the necessary information about the geometry or the dynamics of the vascular system. For example, on echocardiography images and on standard Cardiac magnetic resonance imaging (CMR), blood appears without enough contrast to study, and in some cases effectively invisible. When there is a clinical mandate to study the blood flow, contrast media is injected to provide a clearer image. Furthermore, blood flow patterns play a major role in the events during the surgical procedure and the body's response. When the blood flow is disrupted, a clinician cannot predict the haemodynamic changes resulting from treatment. Subsequent decisions may be prone to error, and the effect of an intervention may lead to further unforeseen complications, for example, stenosis or other weaknesses may be 'hidden' within occluded portions of the vessel. Careful planning is required to mitigate these possibilities.

The uncertainty driven by lack of resources or experience can feasibly lead to medical error during treatment consideration and surgery. Although it is not always referenced on death certificates, in the United States from 1990 to 2016, there were an estimated 123,603 deaths with adverse effects of medical treatment (AEMT) as the underlying cause [174]. A modest improvement when accounting for age and geographic variability. Clinicians are drawn to numerical simulations to mitigate the potential for error, help them understand the problems they are faced with, and predict their potential outcomes. Computational fluid dynamics (CFD) for cardiovascular

blood flow simulation is increasingly being applied and researched to create an optimal solution for clinicians when making surgical decisions [177, 146, 75, 114]. Modelling the flow is often beneficial in cases where the data is difficult to obtain experimentally e.g. *in vivo* measurements of blood flow can be limited by the vessel accessibility and the cost of the procedure. The crucial components of cardiac flow simulations are well recognised: domain or geometric mesh, boundary conditions, and numerical methods of the solver. Simulating the flow is a multifaceted problem and research has often been limited to purely numerical simulations. The simulations are increasingly reliable, but research is still ongoing to resolve complex patient specific geometry and the validation of results produced.

Moreover, simulations that produce intuitive or interactive results are scarce. Three important aspects of numerical computing are robustness, accuracy, and speed. These are typically not mutually satisfiable, requiring development to prioritise certain aspects of the deliverable product [47]. Furthermore, trying to reconcile complex computations, like fluid dynamics, to operate in an interactive application is a non-trivial task. When computer graphics technologies are applied, it is often to render previously calculated scenes or harness the computational power of the graphics processing unit (GPU) to accelerate computation.

When simulation results are visualised graphically, they fall into the traditional desktop computer conventions, sometimes, referred to as window, icon, menu, pointing device (WIMP). When handling complex 3D geometry and flow visualisations, this paradigm becomes unwieldy and value is lost when mapping (or translating) 2D actions into 3D. Conversely, interactive 3D visualisation, which includes Virtual Reality (VR), provides users with a fully 3D experience, including natural 3D interaction. The experiences created by these technologies allow more natural methods of interaction, more closely mapping to the techniques used in the real world. Recently, extended reality (XR), an umbrella term for the numerous modalities related to VR,

has become more available and affordable, encouraging a plethora of new possibilities for research and development. Interactive 3D visualisation is already used in many military, scientific, and industrial applications, ranging from training astronauts to designing automobiles [9]. Modern devices allow for levels of interactivity, and degrees of freedom that were previously limited to the realm of expensive bespoke hardware solutions.

Health care changes dramatically with modern digital developments. Imison et al. [71] conclude that delivering technology powered health care will improve patient outcomes and make computing so ubiquitous that it becomes invisible. Their enthusiasm is shared by the UK government as in 2018, the Secretary for Health and Social Care promised almost half a billion pounds to transform technology in the National Health Service [162]. Robert Pearl, former CEO of Kaiser Permanente, the largest not-for-profit managed care providers in the U.S [172, 138], has expressed his vision of the future of technologically enabled health care:

> 'I think we're about to come to the next era of medicine... as much as 30% of what we do today we will do differently... how we evaluate patients, how we follow up on patients, how we bring the expertise in between clinicians, how we manage patients in a hospital, how we think about even the role of the hospital [71].'

Computational power has grown year on year, doubling approximately every 1.3 years [41]. Medical software systems will continue to benefit from the improvement in computational speed and the algorithms it drives. Clinical decision support systems (CDSS) are well known to improve the quality of clinical decision-making [55]. In recent years new operating environments have been fabricated in the form of 'hybrid theatres', allowing single stage, hybrid endovascular and open intervention for a range of pathologies [153]. For example, in 2018, the Imperial College Healthcare

NHS Trust was awarded funding to create a new endovascular hybrid theatre at St Mary's Hospital, delivering cutting edge procedures to patients with complex blood vessel problems [72]. This is perhaps the result of a convergence of approaches to the treatment of CVD with combined cardiology, radiology and surgical multidisciplinary team based management [51]. These theatres are suitable for very advanced surgical applications, being equipped with modern technology and equipment such as, fixed C-Arms, CT scanners or MRI scanners, image fusion, 3D/4D imaging, soft tissue visualisation, modelling, and navigation [125]. In the future we envision operating theatres equipped with immersive 3D technology informing clinicians' decisions and enhancing their capabilities providing valuable insights interactively in real-time. Currently, there is a lack of tooling for the visualisation and simulation of physiological processes, such as modelling the processes of blood flow. This is especially true for environments such as the hybrid theatres, were integrated technology provides comparatively quick results for the procedures they utilise, the time cost associated with accurate and meaningful simulation enabled visualisations is inherently restrictive. The proposed software prototype would integrate seamlessly into our envisioned 'cyber hospital' of the future.

1.2 Aims and Objectives

We aim to produce the simulation engine for a modern prototype surgical planning tool to aid in the clinical decision-making processes by implementing a 3D real-time CFD simulation based on smoothed particle hydrodynamics that that operates within complex patient geometry.

- Implement modern particle fluid simulation methods used in computer graphics to create an accurate blood simulation.

- Optimise the framework to perform effectively as a real-time application.
- Apply the framework to real patient geometry which was generated from modern medical imaging technology.

The results of the CFD simulation will be visualised in 3D to create an immersive and interactive experience for the user.

- The user must be able to view the simulation on a standard desktop monitor and/or a monitor that supports stereoscopy for 3D viewing.
- The user must be able to interact with the fluid flowing through the patient geometry, we envision allowing the user to create different scenarios with blockages to investigate the effects of the fluid re-circulation.
- The visualisation must be configurable to provide each user with an effective experience.

We aim to validate the results of the simulation, proving the accuracy of the CFD model for use by clinicians.

- The simulation will be tested to ensure it will reproduce results based on typical scenarios found in diseased and healthy vascular anatomy.
- A series of test cases will be developed to compare the simulation's numerical and visual output against known and verified fluid models.
- Qualitative feedback will be obtained from experienced clinicians to determine their professional opinions for clinical use of the framework.

1.3 Hypothesis

Following the motivation outlined above, this thesis investigates the following hypothesis:

> 'A real-time computer simulation and visualisation of blood flow through vascular structures can be developed, validated and integrated with the training and decision-making process of clinical interventions during an acute cardiac event.'

We propose that it would be beneficial if clinicians could simulate potential pathological and treatment scenarios, perform meaningful interactions with the rendered blood flow and patient geometry, within an immersive no-risk 3D experience. Then use and trust the results of the visualisation to evaluate the haemodynamic consequences and inform their choices for a patient's course of treatment. Our envisioned workflow is demonstrated in Figure 1.1. However, the proposed system would function as a software library, and be fully applicable to multiple implementations. The real-life applications are widespread, for instance, the software could provide tangible benefits to medical training or procedure rehearsal as a component in a cardiac surgery simulator. In addition, the visualisations produced could be delivered to patients in waiting as a component of their pre-operation brief, to help them understand the procedure they are to undertake.

1.4 Contributions

Several contributions to the state of the art of real-time computational blood simulation and visualisation are made in this thesis:

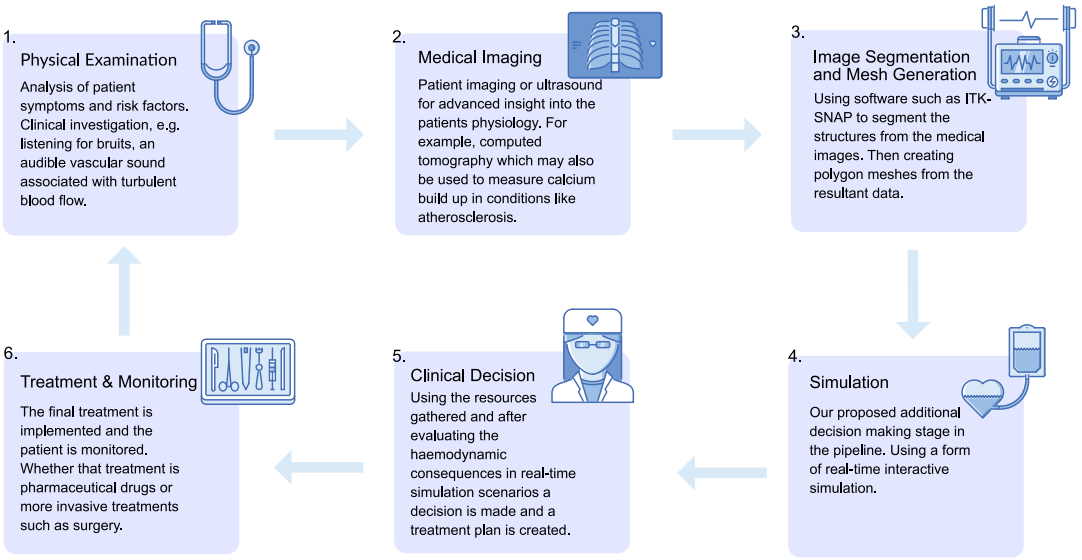


Figure 1.1 Timeline of the proposed addition of simulation into the medical decision-making workflow for cardiovascular conditions, such as atherosclerosis.

- The implementation of a prototype interactive patient-specific blood flow simulation and visualisation tool. The simulation is developed with modern particle-based fluid simulation methods and designed to operate within real patient geometry and produce results in real-time.
- Partially automated workflow including the importing, processing, rendering and visualisation of patient-specific medical image data with interactive CFD results.
- We have demonstrated through a series of validation cases that the simulation is capable of reproducing expected fluid behaviours.

Publication

- Holland, M. I., Pop, S. R., and John, N. W. VR Cardiovascular Blood Simulation as Decision Support for the Future Cyber Hospital. In 2017 International Conference on Cyberworlds (CW), pages 233–236. DOI: 10.1109/CW.2017.49.

1.5 Structure of the Thesis

Chapter 2 of this thesis presents an overview of research into medical imaging and immersive computer visualisation technology, the state of particle based CFD simulations and their application to the simulation of blood flow, the challenges researchers have faced trying to implement these methods to produce real-time results. Current software solutions are explored to highlight the need for a real-time patient specific alternative. Modern rendering and computational hardware required to implement such systems are then explored.

Chapter 3 presents the implementation details and challenges for the prototype real-time blood flow simulation and visualisation application. The development first includes CPU prototypes then advances into GPU accelerated solutions. The chapter explores optimisations for parallel computation of the CFD methods, geometry processing, and real-time rendering techniques for the visualisation.

Chapter 4 finalises the implementation of the simulation engine by cataloguing a series of validation test cases. Each test case is modelled to resemble an aspect of vascular geometry. Results from each test case are examined to demonstrate the simulations efficacy against known fluid behaviours. In addition, the simulation is applied to real-patient geometry signifying the stability in complex environments.

Chapter 5 explores the usage of the simulation engine within a prototype application and the feedback received from an experienced clinician. Interaction with the loaded geometry and simulated blood flow is detailed, including user interaction with the fluid, simulation parameter modification and immersive 3D hardware integration.

Lastly, Chapter 6 concludes the ideas presented and introduces a number of potential routes for the future development of the simulation engine and the prototype application.

Chapter 2

A Review of Real-Time Fluid Simulations with particular emphasis on Blood Flow

This chapter begins with an overview of work related to the requirements and definition of real-time simulations. Secondly, the chapter provides an exploration of the fields involved in the simulation and visualisation of real time blood flow and the methods to validate simulation results, in conjunction with emphasising the hardware and software developments required to produce such a solution. Currently available professional and research CFD software solutions are also evaluated. Finally, this chapter concludes with an introduction to immersive 3D visualisation technologies, including the potential application of such technologies in medical environments.

2.1 Real Time Simulations

Computer simulation is an increasingly important tool for approaching complex practical problems in engineering, science and medicine. By offering further insights

simulations assist in the interpretation of complex physics [97], or in cases where *in vivo* empirical measurements are difficult or invasive [177]. In the context of real time computing, the correctness of the system depends not only on the logical computations, but on the time that the results were produced. A real time application's functional and timing behaviour should be as deterministic as possible, but fast computing alone does not guarantee an appropriate level of predictability [170]. Rather, a real time computing system would be built of time critical tasks methodically scheduled depending on priority to meet their respective deadlines. These deadlines can be defined from the application specific time critical tasks, deadlines that cause a marked detriment or failure of the system [161], or the performance requirements of the computation, being able to generate the results before or when they are needed. Moreover, computer simulations tend to execute logical calculations successively in fixed or variable time-steps [18]. Consequently, for a computer simulation to operate in real-time the application's time steps must emulate or correspond to the physical passage of time whilst maintaining visual continuity when appropriate.

2.1.1 Visual Continuity and Frames Per Second

The goal of computer animation is to create a sequence of pictures that give the illusion of movement, by filling an array of picture elements called pixels on a screen for every image, known as a frame [167]. Maintaining the visual continuity of these animations is an interesting issue with no clear consensus, what is the minimum frequency that frames need to be rendered to provide the user with an appropriate experience. This frame-rate is often expressed as a unit of Frames Per Second (FPS), i.e. in Hertz. Early examples of standardised frame-rate can be found in the variety of equipment used during the early stages of the film industry. In the 1920s, features were filmed at 16 FPS and adequately provided the requirements for visual continuity. After 1930 the film industry standardised to maintain 24 frames per second to mirror the 24 frame

soundtracks [150]. Computer rendered imagery is often different from the imagery found in films, even in animated films the use of many additional effects to enhance the visual continuity of their imagery is still present, such as motion blur and interlacing. Little information is available pertaining to the interaction of rapidly varying complex rendered images and the human visual system [39]. Although 30 frames per second is often viewed as an appropriate frame rate for interactive/real-time computer graphics applications [61, 139], less may be acceptable for some applications [24], and some suggest that a frame rate as low as 10 frames per second is the minimum to achieve a real time animation [101].

Lower frame-rates may be satisfactory for conventional mediums but visual continuity in VR can require much faster refresh rates. According to the best practice guidance for the zSpace stereoscopic 3D hardware, for quality and comfort, applications targeting zSpace are strongly advised to operate at no less than 45 FPS [28]. Similarly, the guidelines for the Oculus Rift, an industry leading VR head-mounted display (HMD), state that applications on their storefront must operate at 45 FPS on machines meeting their minimum specification and 90FPS on machines at their recommended specification. This is especially problematic as every frame must be typically drawn twice, once for each eye, and there may also be some overhead required to apply distortion to the final output frame [130].

Both computational throughput and variable frame rate requirements have driven many researchers to exploit the power of the parallel computing. First researchers tried to maximise the potential within modern general-purpose CPUs, namely by utilising the multi-core architecture with thread-based parallelisation. Secondly, multi-system computer configurations can be interfaced through the message passing interface (MPI) standard, extending the computational potential with each additional machine. However, this delegation to independent devices can incur a computational cost from, managing the distribution of the workload, the time dedicated to data exchange and,

the time consumed during synchronisations [191]. Finally, the programmable GPU has been another active area of development to optimise simulations [44, 62], but this change in the hardware architecture does cause another different set of certain difficulties or limitations, such as, the restriction on the topology of the deformable model, parallelisation of branching algorithms [132]. However, the GPU provides the possibility of achieving parallel computations in real time.

2.1.2 The Graphics Processing Unit

The GPU was developed and used as a special purpose device to render 3D images and scenes, more specifically geometric primitives and the techniques surrounding them, anti-aliasing, texturing, shading etc. As such, the primary application of GPUs has been computer games and other digital art sectors, e.g. film and graphics production. As technology progressed, the architecture of the modern GPU hardware became highly attractive in general-purpose computing for large scale parallelisation. Consequently, graphics rendering pipelines began to include the capability to offload some computation from the CPU to the GPU. This process generally required the computational problem to be formed in a way the GPU could process, using simple graphics elements and geometric primitives. More recently, this unwieldy process has been alleviated with the introduction of GPU programming, APIs that require less graphics programming knowledge but allow developers to write programmes targeting execution on the GPU.

Due to the benefits of GPU programming one-third of the world's top high-performance computers were GPU accelerated in 2018, an increase from previous years [126]. The primary difference between a GPU and a CPU is the number of processor cores. Rather than having a low quantity of highly efficient cores designed to be used in a sequential programming model, GPUs have a massive number of cores, while these cores may be slower for general operations, they execute far more

operations in parallel, see Figure 2.1. The differences in architecture lead to differences in processing patterns. CPUs employ a multiple instruction, multiple data (MIMD) technique to achieve parallel computation, i.e. the CPU cores can employ operations independently to efficiently handle computation. Conversely, the GPU often uses the single instruction, multiple data (SIMD) technique to operate its substantial quantity of cores to perform the same operation on large data sets. SIMD performance suffers considerably when programme logic branches but outperforms MIMD during huge amounts of parallels computations. Consequently, GPUs tremendous computational power has vastly outgrown conventional CPUs. In 2010, Intel measured a 14.9 times speedup for collision detection algorithms on a NVIDIA GTX 280 over the Intel Core i7-960 [93]. As a result of the effectiveness and popularity of GPUs for computation, Intel have recently announced a return to the discrete graphics market [74].

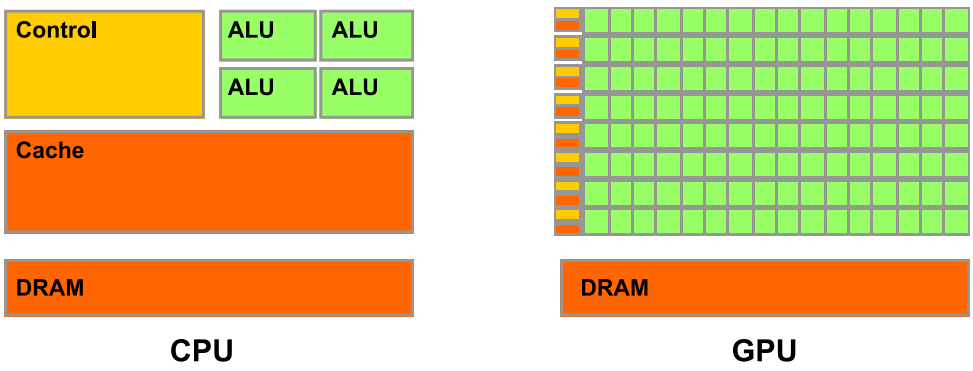


Figure 2.1 An illustration of the differences in CPU and GPU architecture. The GPU boasts a greater quantity of arithmetic logic units (ALU), the computational building blocks of the processor, paired with multiple smaller control units and caches. (Source: NVIDIA Corporation [129])

GPUs are readily available, with one in almost every desktop system, and they are benefiting greatly from Moore's law and the continued increase in transistors in computer hardware. The performance of GPUs has been seen to increase substantially during their release cycle every 9 to 18 months [176]. Moreover, these benefits are beginning to be applied within scientific computation in general but with excellent

applications in fluid dynamics. Finally, where previous research implementations connected additional computer systems, additional GPUs can easily be installed in one computer system to massively increase performance.

Programming computer applications to be accelerated by GPUs can require more specialised programming models. Outside of graphics engines, software-development environments for video games, there are two main GPU parallel computing programming models for general computing on GPUs, general purpose APIs for GPU programming and compute shaders. Popular GPU programming APIs include NVIDIA's CUDA and the Open Computing Language (OpenCL). Compute shaders are a standard component of graphics APIs such as OpenGL [184], Vulkan [187] or Direct3D [112].

NVIDIA's CUDA is a proprietary platform and programming language that debuted in 2007, it includes both a runtime API and driver API. When developing with CUDA, the developer can still program in their preferred high-performance language, generally being C, C++, or Fortran, but other languages are supported, and by incorporating CUDA keywords as extensions can accelerate their programming code. As a proprietary platform CUDA can only compile and execute on GPUs that are manufactured to a NVIDIA designed specification. This could be limiting to the end-user, but NVIDIA are prolific, and their hardware is readily available commercially to businesses and the consumers market, as seen in the workstation (Quadro) and consumer (GeForce) lines of graphics hardware.

Conversely, and as the name would suggest, OpenCL is an open royalty-free standard for cross-platform, parallel programming. OpenCL is maintained by the KHRONOS group, a non-profit technology consortium. First released in 2008, OpenCL was developed to be an easy to integrate parallel processing API, providing task and data-based parallelism. The current kernel language/syntax is a static subset of the C++14 standard. Much like OpenGL, the API is platform agnostic, each hardware vendor implements the functionality of the API in their own graphics driver

software. Consequently, OpenCL has no device limitations and is able operate on most graphics hardware.

Finally, a compute shader is a stage of the graphics pipeline that is used entirely for computing arbitrary information. Modern graphics hardware uses a programmable shader-controlled pipeline. Until recently, computer graphics shaders were only used in the graphics pipeline to control the details of how each pixel is eventually rendered. The shading languages are used in small discrete programs that conform to the various stages in the graphics pipeline. With the addition of compute shaders to both the OpenGL Shading Language (GLSL) [182] and the High-Level Shading Language (HLSL) [156] the graphics pipeline can now be used for general purpose tasks. Compute shaders can operate on most hardware as a component of the modern OpenGL specification, or on DirectX compatible systems with HLSL. Furthermore, all common shader formats can now be compiled down into the SPIR-V format, a cross-API intermediate language that is fully defined by Khronos with native support for shader and kernel features used by APIs such as Vulkan [186].

In comparison to compute shaders, CUDA and OpenCL are more fully featured for general computing and have even seen the development of GPU-accelerated libraries to further facilitate general development. As modern shaders have now been co-opted into running computations as their singular purpose in the graphics pipeline, they suffer from a lack of stricter precision guarantees [183, 185], complex access to the memory details, trial and error debugging. Consequently, compute shaders offer a close to the hardware combined approach with graphics rendering at the cost of developer time. There is also no standard industrial/academic model, but the technologies have matured to be sufficiently capable. As CUDA is developed by NIVIDA and in tandem with NVIDIA hardware it is expected to occasionally deliver results that achieve improved performance over OpenCL due to platform specific optimisations and features [79]. However, there is also a lack of thorough modern

performance comparisons between OpenCL and CUDA, but the consensus seems to be that the frameworks are competitive [45, 173, 40].

2.2 Fluid Flow

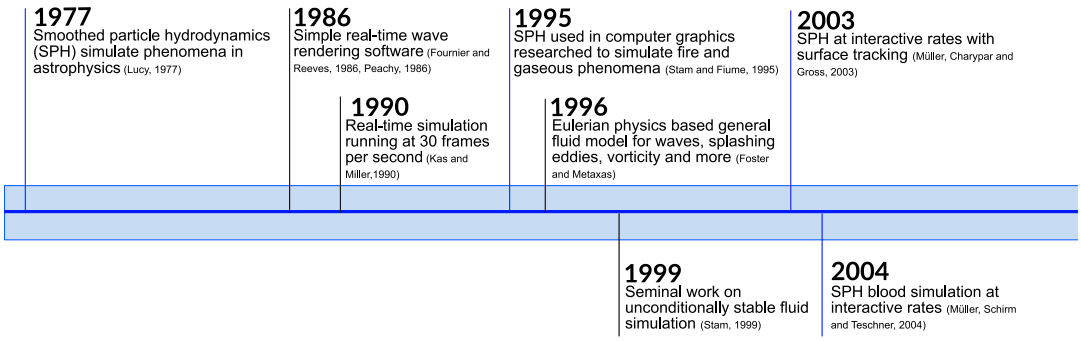


Figure 2.2 Timeline of the history of fluid flow with emphasis on the development of real-time SPH.

An important component of any modern surgical planning tool is an accurate modelling system that realistically simulates the complex mechanical interaction and physiological behaviour of the human body with respect to the surgeon's actions. Historically, CFD has focused on the Eulerian frame of reference, calculating fluid properties within fixed regions in space [14], leading to a number of grid-based methods. Grid or mesh based numerical methods such as the finite difference methods (FDM), finite volume methods (FVM) and the finite element methods (FEM) have been widely applied in CFD [97]. Mesh-free methods tend to substitute the grid formulation for a set of arbitrarily distributed nodes and calculate the fluid properties from the Lagrangian frame of reference [14]. As a result, these methods are expected to be more adaptable and versatile than their grid-based counterparts [57]. In the context of computer graphics, it is worth clarifying the distinction between a mesh for computation and a surface mesh of a virtual object. A computational mesh

represents the geometric Eulerian domain by breaking it down into smaller discrete cells, effectively creating a grid, which in turn are used to compute solutions of partial differential equations, see Figure 2.3. Conversely, a surface mesh is a form of polygon mesh, usually triangle, where each polygon is connected by their common vertices and used to represent and render a virtual object in computer graphics, see Figure 2.4.

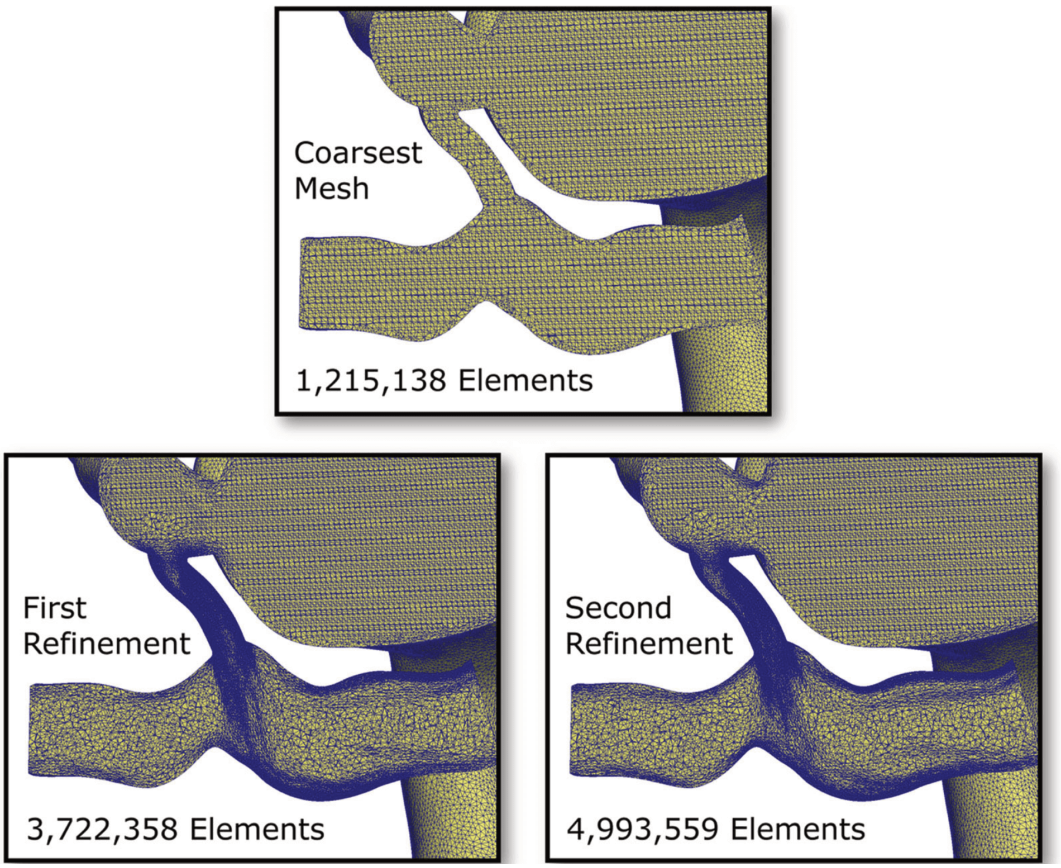


Figure 2.3 Cut-through views of the finite element meshes within the pulmonary arteries. Three levels of refinement are shown, facilitating areas with sharp corners or edges by creating softer blend elements [154]. (Source: Arthurs et al. [8])

Research into real-time fluid simulation has a long history of interesting incremental developments, see Figure 2.2. In the late 1980s, initial attempts were made to model fluids using computer software in real-time. These early simulations were primarily concerned with producing wave-based animations [54, 134, 190] for their use in film and media; results were computationally inaccurate, avoiding solving fluid differential equations in favour of wave equations, and pre-rendered for improved

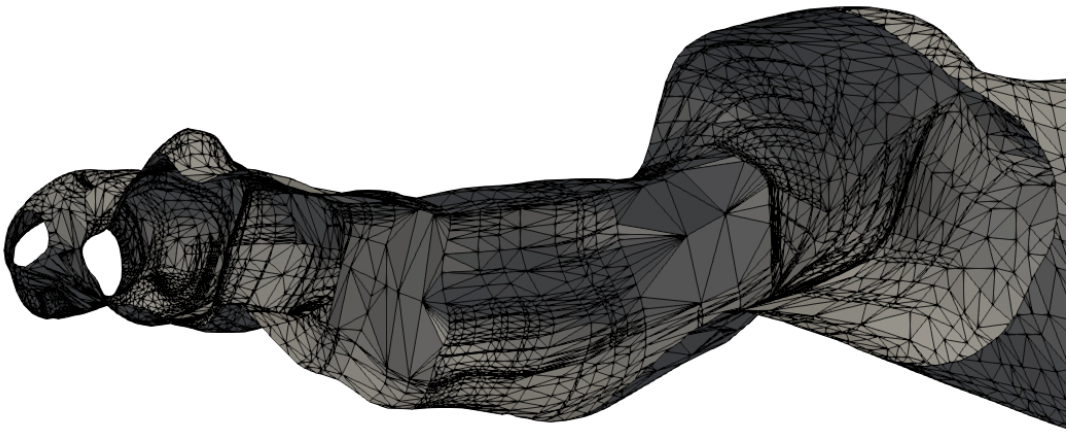


Figure 2.4 Cross section view of a surface mesh depicting a portion of the aorta. The region inside the surface mesh contains no vertices.

graphical fidelity. In 1990, Kass and Miller [80] presented an early implementation of a real time fluid simulation. They developed a simplified simulation model to execute a three-dimensional wave animation at 30 frames per second. The simulation does not take place within any particular geometry but functions under changing boundary conditions. It was demonstrated to flow over various terrains and be suitable for flowing rivers. The computational model was developed from earlier work that involved numerical methods of applying the Navier-Stokes equations [115, 133, 78]. Although to achieve an adequate frame rate for real-time operation, a number of assumptions were made to create the simplified model. Firstly, the water surface is considered a height field, as a result it could not splash, and waves cannot break. Secondly, the vertical component of the velocity of the water particles was ignored, very steep waves would be inaccurate. Finally, the horizontal component of the velocity in a vertical column was said to be constant, providing inaccuracies with turbulent flow. Consequently, these physically unrealistic assumptions produce results that are inaccurate and only fit for animation or demonstration.

Chen and Vitoria Lobo [29] endeavoured to develop a more physically accurate model than their 1995 predecessors with the addition of floating objects, self-propelled objects, streaklines of fluids and blending while still operating in real time. Applying

the Navier-Stokes equations is a requirement to create a physical foundation for animating fluid [29, 80, 169, 4]. The model calculated the Navier-Stokes equations in 2D, because the equations in 3D would be so computationally expensive that it would prevent the simulation from running at interactive rates. Imitating three-dimensional motion in this way is not accurate enough for most engineering applications. Instead the model uses the results gained in 2D and added the third dimension solely using a pressure-based height field treating the fluid as being completely flat. The only previous example of self-propelling objects through a fluid medium being simulated in real time was from Goss in 1990. However, Goss's implementation was not based on physics, making it far less useful in application than Chen and Vitoria Lobo's method of introducing internal boundary conditions to objects and then applying their physical model. A chief concern of the model was the stability of the numerical computation. Working at a detriment to the accuracy of the simulation, the stability can only be achieved through the manipulation of various parameters. For example, high Reynolds numbers in the calculation could result in numerical divergence, the variable was raised to prompt various turbulence behaviours but would then have to reduce to prevent the divergence.

Foster and Metaxas [53] initiated the next phase in fluid simulation, by moving into a more refined eulerian setting they created a physics based general fluid model. This eulerian aspect means that the fluid was discretised into an arbitrary fixed grid, velocity and pressure are defined throughout and updated by a set of finite difference expressions. The model built upon its predecessors by more accurately determining the motion of a liquid in 3D when solving the Navier-Stokes equations. This approach naturally allowed for the possibility of previously simulated effects such as, wave reflection, refraction and diffraction, eddies, vorticity, and splashing. The approach developed the model with a firm focus on explicit time stepping, on each time step the density and velocity of each cell is moved across the velocity field

to neighbouring cells. The maximum stable time-step for numerical integration was limited to maintain a straightforward implementation. Consequently, larger time-steps were inherently unstable, especially when formulating the velocity field. Finally, to maintain results conducive to fluid's incompressibility a relaxation coefficient was implemented, introducing a further potential source of inaccuracy into the results.

Stam [166] innovated upon Foster and Metaxas's development path by presenting a method that provided unconditionally stable fluid dynamics at any time-step. The stability was achieved by the vastly improved advection scheme; Stam departs from Foster and Metaxas's method by implementing an implicit, and partly Lagrangian advection scheme, based in part on a technique known as the 'method of characteristics'. The method is applied to determining the velocity of a point by calculating a linear backtrace across the velocity field and interpolating the results. A second glaring improvement from their predecessor's was the use of Helmholtz-Hodge decomposition. It states that the vector field can be decomposed into a scalar gradient field and a divergence-free vector field. The technique more effectively fulfils the role of the relaxation coefficient Foster and Metaxas used, correcting any divergence. Stam acknowledges that the method produces results too inaccurate to be suitable for use in most engineering applications. Namely the results suffer from what Stam describes as "numerical dissipation", i.e. the flow tends to dampen too rapidly as compared to actual experiments.

2.2.1 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a mesh-free mathematical model which has been applied to CFD, and its use in computer graphics is a modern and active area of research. SPH was first introduced in 1977 with applications within the field of astrophysics [100]. The SPH model is based upon Lagrangian formulation, in which the continuous fluid is discretised into a finite collection of distinct 'particles'. These

particles act as interpolation points within the method, and they are used to calculate the dynamic properties of the fluid (e.g. density, pressure, velocity) at each stage of the simulation.

These particles interact within the bounds of a smoothing kernel function. These smoothing kernels have a fixed length spatial radius, i.e. all neighbouring particles that would fall within this radius have a weighting effect for any given property. Therefore, the physical description of a particle is calculated by summing the properties of its neighbours with the addition of the normalised and symmetrical kernel function. The computation speed and accuracy of a SPH simulation is highly dependent on the choice of smoothing kernel [119].

In 1995, Stam and Fiume introduced the use of SPH to the graphics community when developing their animated depiction of fire and other gaseous phenomena. Müller, Charypar, and Gross implemented the techniques to create an SPH based fluid simulation running at interactive rates in 2003 [119]. Müller, Charypar, and Gross developed bespoke smoothing kernels for the application and used an implementation of Desbrun's 'spikey' kernel [42] to solve a problem with repulsion forces, particles in close proximity would have an erroneously low repulsion force as the kernel gradient reaches zero at the centre. Müller, Charypar, and Gross's implementation of the model was integrated using the Leap frog scheme [66], an integration technique named after its alternating time steps "leaping" over their counterpart, see Figure 2.5. Leap frog integration achieves second order accuracy while being conceptually simple, although the scheme does require some logically branching for the first step of the simulation.

Surface tracking and rendering techniques replaced the height field assumption found in previous fluid simulation attempts and improve the quality and accuracy of the visualisation when surface reconstruction techniques when implemented. Surface tracking and rendering is a method used to display a projection of an isosurface

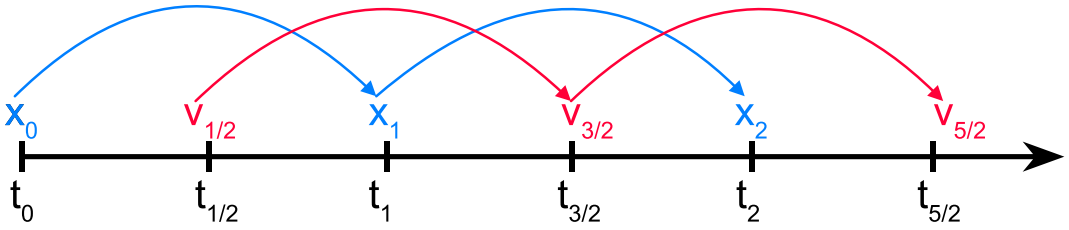


Figure 2.5 Illustration of the leap frog integration scheme. Depicting the integration of velocity (v) and position (x) over time (t) in staggered time steps, "leaping" over one another.

representing a 3D data set [35], see Figure 2.6. The surface of the volume is estimated and reconstructed from a discrete scalar field into an isosurface, typically as a triangle mesh. The triangle mesh is generated using algorithms such as marching cubes [98], as the name would suggest, the domain is divided into a discrete collection of cubes. The cubes are then examined with reference to neighbouring cubes and the polygon faces required to represent the surface within are determined. Although, these techniques are computationally expensive and can raise a problem when maintaining interactive frame rates. When constrained to a maximum of 2200 particles the simulation ran at 20 FPS, an acceptable rate to be considered real-time. However, after applying surface rendering techniques the simulation speed dropped to 5 FPS, a far less appropriate rate. Müller, Charypar, and Gross claimed that during the next generation of graphics hardware, real-time performance will be possible. Although the sentiment may potentially be true for this particular implementation, hardware will not resolve inaccuracies found in the method, solutions to which may require further computational power.

The method is a reformulation of the classic naiver-stokes equation. The calculation takes the form of Equation 2.1. The scalar quantity A is interpolated at the location of particle $\mathbf{r}_i$ by iterating over the particles as j within the radius of the smoothing kernel. Where m_j is the mass of a particle, ρ is the density. Finally, the

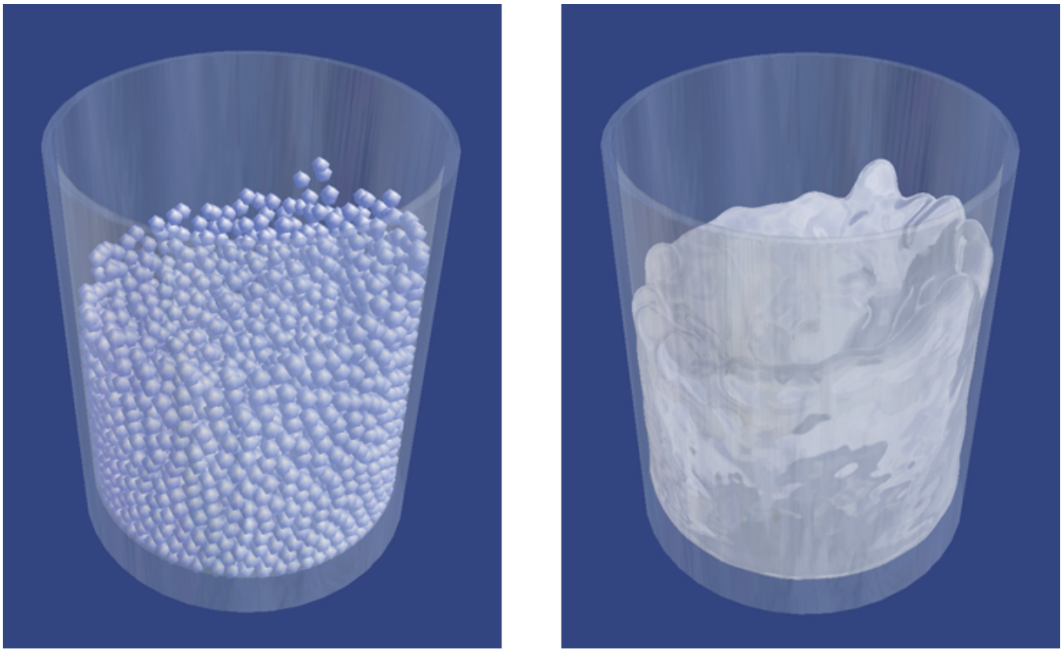


Figure 2.6 Demonstration of SPH fluid surface tracking using the marching cubes algorithm. Image (a) shows simulated fluid particles, (b) the isosurface triangulated via marching cubes (Source: Müller, Charypar, and Gross [119]).

function W is the smoothing kernel function, in which h is the kernel smoothing length and denote the finite support radius in the formulation.

$$A_s(\mathbf{r}_i) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.1}$$

As a mesh-free method, the implementation of SPH is well suited to simulations involving complex boundary dynamics, for example, free surface flows, large boundary displacements and interactivity. However, these benefits are largely due to the multifaceted cost of building the mesh. In fact, computing the effect of a boundary from a mesh can be very straightforward, using the mesh edges can enforce Neumann, where the normal derivative of the function is specified on the boundary, or Dirichlet boundary conditions, value of the function on the boundary is specified. However, the treatment of boundary conditions is certainly one of the most difficult technical

points of the SPH method [159]. The complexity arises from the Lagrangian nature of the method, and the boundary being external to the particle computations. Particles within close proximity of a boundary edge have some portion of the kernel radius deprived of particles.

Research into SPH as a viable model of fluid dynamics is a rapidly advancing area and its applications in engineering are becoming more recognised [194]. Improving the accuracy and convergence of the model are primary goals for complex simulations. Performance is constantly improving with improvements to computer hardware, algorithmic techniques and optimisations. However, to move SPH into a real-time context certain factors become limiting. Primarily, the computational performance. As the number of particles in a simulation increases, the performance decreases, traditionally taking a number of hours, or days in extraneous cases, to perform the computation of a few seconds. This incurs a fragile balancing function between accuracy and performance.

The improved accuracy, stability and adaptivity of the modern SPH method has reached an acceptable level for practical engineering applications [96]. Moreover, SPH has been widely applied to various areas in mechanical engineering, coastal hydrodynamics, high strain hydrodynamics with material strength and many problem domains [97].

2.2.2 Software Solutions for SPH

Due to its many engineering applications, SPH implementations can be found in numerous commercial and open-source software packages, for example, DualSPHysics [46], COMSOL [73], and RealFlow [179] are pioneering excellent research and engineering solutions for hydrodynamic problems. Video game development has also advanced the use of SPH, creating visibly plausible and interesting effects with maximum

performance, branching into their own models, e.g. Position Based Dynamics (PBD) forsaking some accuracy for high fidelity, visually appealing graphics [102]. Examples include NVIDIA's PhysX [34] and Flex [33], real time GPU accelerated software library solutions for video game physics and fluid, respectively.

RealFlow is a commercial software product developed by Next Limit and bundled with several solvers built using SPH and position based dynamics (PBD). RealFlow is marketed for the 3D and visual effects industry and is often found within media such as films and advertising. One major caveat for the prototype we envision is the lack of real-time interactivity. RealFlow, for instance, requires that interactions are scripted ahead of time, creating a trial and error workflow between simulations.

COMSOL is another commercial software product featuring finite element analysis, solver and multiphysics simulation. COMSOL can be configured to obtain results for a variety of applications, including fluid flow, heat transfer, solid mechanics, and electrodynamics [137]. As a multiphysics solver, COMSOL excels in domains where multiple modalities are coupled. Moreover, case studies using multiple fluid simulation models have been developed to open avenues into the use of COMSOL for elements of blood simulation [163].

DualSPHysics on the other hand is a free open-source software package based upon the SPH model. Research using DualSPHysics has been primarily focused on the wave propagation and interaction with coastal structures in 2D and 3D [57]. Initialisation of the simulation is done with configuration files and the visualisation is handled by ParaView [10].

ParaView is frequently used within the scientific community as a method to visualise and analyse extremely large data sets. Aside from CFD, ParaView has found use within nuclear energy, wind energy, forest fire modelling, carbon sequestration and cosmology [19]. The open-source software provides a graphical user interface (GUI) for users to generate and execute a variety of visualisation tasks, including

some SPH specific interpolation visualisations. ParaView was designed to integrate a number of diverse requirements including the ability to handle large data, ease of use and extensibility by developers [2].

2.2.3 Blood Simulation

Medical simulations are often more complicated by the nature of the domain they target, the human body which varies person to person, and changes from moment to moment, however even simple simulators can provide enormous value [155]. At every stage of the circulatory system, whether blood is swirling in the heart or streaming through the arterial tree, a range of mathematical models have been employed to quantify biomechanical conditions [177]. With approximately 100,000 kilometres of blood vessels in the adult human body [25], computationally simulating the global blood circulatory system is a large multifaceted task [148], instead simulation is often used to target specific smaller regions of anatomy or pathology, increasing the accuracy of the simulation and its practical value in planning surgical procedures.

Numerical models have been developed to simulate both Non-Newtonian and Newtonian fluids. Non-Newtonian fluids present as the inverse of Newtonian fluids in respect to the properties of the fluid, primarily the viscosity. Newtonian fluids present with a constant viscosity only affected by temperature or pressure. A non-Newtonian fluid's viscosity is variable and dependant on the various stress forces acting on the fluid. In short viscosity is the property of a fluid to resist the growth of shear deformation. Figure 2.7 demonstrates the relationship between shear stress and shear rate, the gradient of the line being defined as the viscosity of the fluid. While Newtonian fluids show a linear relationship, shear thickening liquids viscosity increases as stress increases and shear thinning liquids exhibit the opposite behaviour.

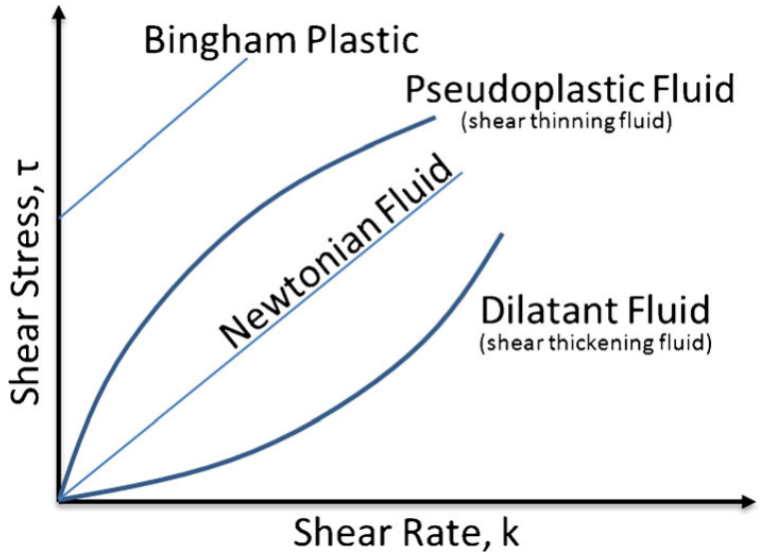


Figure 2.7 Rheology of different kinds of Non-Newtonian fluids, demonstrating the relationships between shear stress and shear rate. (Source: derived from Pinto and Meo [141].

Blood itself is complex entity, it consists of a water-like carrier (plasma) which largely contains fluid-filled vesicles. It is well known that blood behaves as a non-Newtonian fluid, particularly at low shear rates (less than $100s^{-1}$) [77]. The viscosity of blood is dependent on the viscosity of the plasma, its protein content, the ratio of red blood cells to the total blood volume (hematocrit), the temperature, the shear rate, and the narrowness of the vessel (Fahraeus-Lindqvist effect) [11]. However, blood is commonly approximated as a Newtonian fluid in larger arteries as shear rates are expected to be on order of 100 reciprocal seconds (s^{-1}) [171]. Newtonian models are often used to simplify the simulation of non-Newtonian fluids, preventing some computational complexity, as there is sufficient agreement between the computational modelling and experimental data [118].

The complexity of the vascular system guarantees a great deal of variation within the blood flow characteristics. From the medium for the blood, i.e. the vessel, tissue, or organ the blood is flowing through, to the patient-specific conditions like blood pressure, cardiac output or any acute pathology that may have occurred within the system. The vascular system homes a collection of vessels spanning from meters to

microns; flow within differs markedly from microcirculation to the large coronary arteries [196]. Even within small regions there can be deviation in the blood flow between different patients [31]. In general, blood flow in healthy arteries is mostly laminar, non-turbulent flow which contains a great many parallel layers of fluid with no internal disturbance amongst them [14]. The simplest method to model blood flow is to assume laminar, incompressible flow of a Newtonian fluid through straight cylindrical channels of constant diameter. This type of flow is called Poiseuille flow or Hagen-Poiseuille flow, and the flow of fluid can be defined by Poiseuille's law, Equation 2.2.

$$Q = \frac{\pi P r^4}{8\eta l} \tag{2.2}$$

The flow rate Q is based upon the pressure P, the radius r of the tube against the fluid viscosity η and the length of the tube between inlet and outlet l, and proposes that the volume flow is proportional to the fourth power of radius or diameter [195]. Fully developed flow in the aforementioned straight channel tube has a parabolic velocity profile, which can be seen in Figure 2.8.

Blood as an incompressible fluid has been accurately modelled with a Newtonian model and has been traditionally simulated numerically using the incompressible Navier-Stokes equations, these methods were also largely applied from a fixed Eulerian perspective. However, the specific models that have been implemented vary largely, ranging from one-dimensional wave propagation to three-dimensional numerical methods, and can all be used to describe cardiovascular mechanics to great effect. That said, there are still significant challenges in image-based modelling of blood flow, these include algorithmic improvements to geometric modelling, boundary conditions, fluid–structure interactions between the blood stream and vessel wall, hemodynamic parameters, and verification and validation to name a few [178].

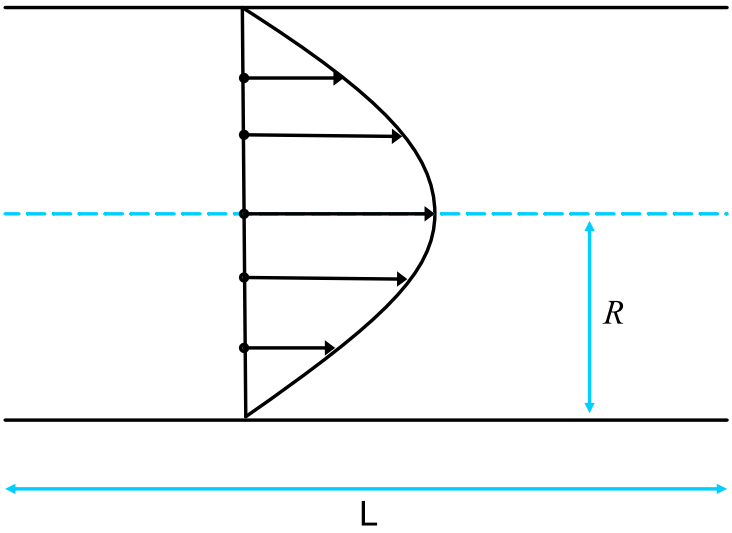


Figure 2.8 Idealised parabolic velocity profile found in laminar flow. The arrows across the velocity profile indicate the magnitude of the velocity in the laminar sheets, shorter arrows having a lower velocity.

Blood flow was quickly considered as potential candidate for SPH simulation in computer graphics. In 2004, Müller, Schirm, and Teschner repeated their earlier success with interactive SPH applications by applying the model to blood flow. The geometry demonstrated was artificial, but the simulation results did produce a boundary reactive to the blood itself. The vessel was perforated to demonstrate the interactivity of the simulation. The bleeding effect produced results similar to cutting surface skin or into the vessel, but no conditions were specified for medium in the regions outside the vessel, air or otherwise. The simulation developed by Nobrega, Carvalho, and Wangenheim was an early example of the integrate realistic triangle-mesh geometry as the medium for a SPH simulation model, and the application did generate and make effective use of a centreline. The boundary condition implemented was an inaccurate reflection condition, simply reflecting particles along the intersected triangle's normal vector. Additionally, the flow was largely guided by a computationally costly force dictated by the tube's centreline, rather than the pressure generated from the particles themselves and the boundary.

Pang et al.'s SPH bleeding simulation was an interesting attempt to accelerate the simulation with a Physical Processing Unit (PPU), a specialised co-processor for physical computations, one implementation was the PhysX chip [103] which was a precursor to the implementation of the same functionality in NVIDIA PhysX and GPUs. The acceleration effects of the PPU were impressive at lower particle counts (below 6000), but the implementation suffered from diminishing returns as the particle count increased, effectively giving no improvement over the CPU implementation when the simulation reached around 6000 particles, see Figure 2.9. This drop in performance of the PPU acceleration may have been due to a data transfer bottleneck, which can still be a limitation found in modern GPUs. The simulation was shown to use some realistic geometry; however, the geometry was not the medium of the flow itself. Kulp et al. produced a GPU optimised variant of patient-specific ventricular blood flow simulation using SPH. Although the simulation did not operate interactively, the performance was greatly improved from previous blood simulation attempts.

Importantly, despite the performance gains from GPU acceleration, validation of blood flow simulations continues to be very difficult. Kulp et al. suggested that the CT and MRI images of a patient's heart could be used with the MRI flow data to compare and validate the simulation. Critically, real-time SPH blood simulation attempts have had insufficient accuracy and been impractical to validate [120, 147], largely due to low particle numbers. More accurate simulation implementations can also take extended periods of time to execute, and in time critical interventions this can massively diminish the usefulness of the results.

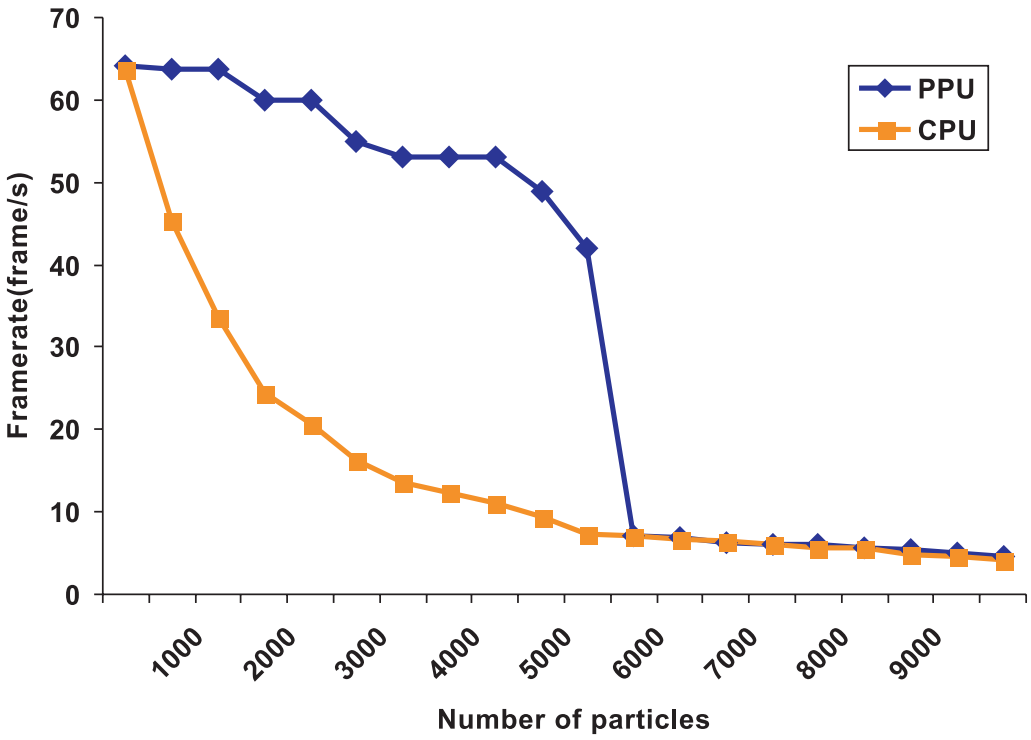


Figure 2.9 Time performance for a SPH bleeding simulation using a PPU and CPU (Source: Pang et al. [132]).

2.3 Patient Specific Care

The development of efficient numerical methods and new computer generations allows the calculation of local flow behaviour under physiologic and anatomically realistic conditions. Although the first applications of the computational simulations were applied to compute velocity fields in generic or idealised vascular anatomy, see Figure 2.10, modern medical imaging has enabled the use of patient-specific anatomic and physiological models within the simulation environment. This patient specific approach has enabled the application of cardiovascular mechanics to the prediction of changes in blood flow during and after clinical treatment procedures [177]. In the late 1990s patient-specific computational haemodynamics began to gather some momentum with researchers bridging image processing and CFD [122]. The further development of efficient methods with improving computer hardware allows the calcu-

lation of flow behaviour under physiologic and anatomically realistic conditions [20]. This has enabled the quantification of cardiovascular mechanics in patient-specific anatomic and physiologic models as a tool for understanding and predicting CVD [20, 144]. Despite the increased research activity in the field, there is a limited number of studies on patient-specific haemodynamic simulations effect on large patient populations. This lack of studies on patient populations is likely based upon the process from image generation to simulation results, having disparate operator-dependant steps including image-segmentation, model preparation, mesh generation and simulation, with no standard integrated workflow [6].

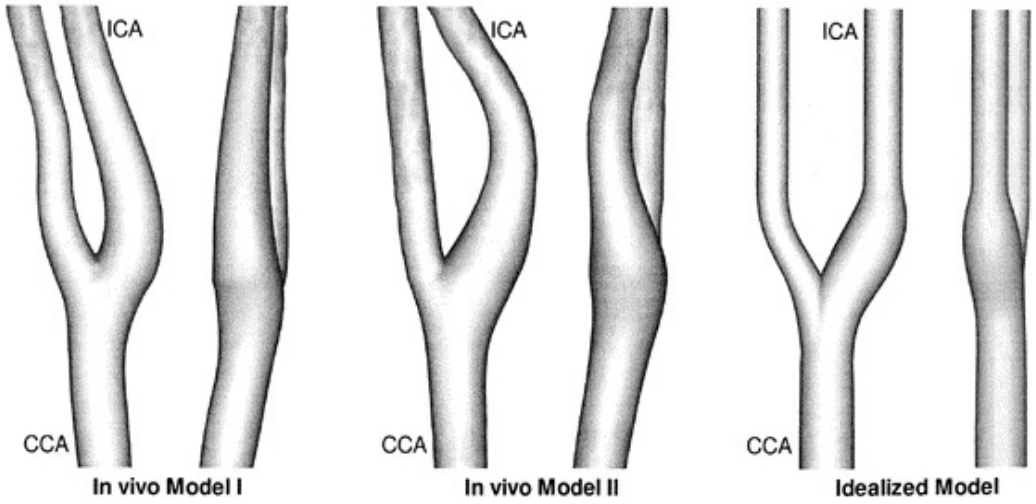


Figure 2.10 Views of the two in vivo carotid bifurcation models reconstructed from black blood MRI and an idealised model. Differences can be seen in the natural resting position, the large natural deviations of radius and angle throughout. (Source: Milner et al. [113]).

As the field uses medical imaging techniques, e.g. MRI and CT, and image processing (acquisition, segmentation and registration) to generate the meshes to either perform the computation or define the patient specific geometry [122], the accuracy of geometric models constructed using these methods is important. Since a 3D model is constructed using only local 2D information from the inherently 3D volumetric data, there is an inherent risk that the mesh will subject to some geometric inaccuracy [177]. There is the potential for variation in the end product after the

conversion from Digital Imaging and Communications in Medicine (DICOM) data sets to Computer-aided Design (CAD) surface mesh formats, depending on the software and the technical parameters used [69].

Software solutions have been developed to create easy access to segmentation and volumetric data visualisation. Most notably, the Insight Toolkit (ITK) [87] and Visualization Toolkit (VTK) [88] from Kitware, Inc [86] are open-source cross platform tool sets for medical image data. ITK provides medical imaging researchers with an extensive suite of leading edge algorithms for registering, segmenting, analysing, and quantifying medical data [157]. VTK, on the other hand, provides a software system for computer graphics, visualisation, and image processing [85]. Subsequently, developers have built upon VTK and ITK to create fully-fledged applications and further software libraries. For instance, the Vascular Modelling Toolkit (VMTK) is a library for 3D reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modelling of blood vessels [105, 140, 38].

SPH as a meshless method immediately reduces the need for the meshing and modelling steps. However, SPH has largely been used to model free surface flows [116, 160, 17], there is a limited amount of cases where SPH was used in combination with complex tube geometry in three-dimensions, especially vascular data. This becomes increasingly true when exploring real-time simulations. Past adoption of physically based models has been limited by the challenging nature of modelling blood flow and the performance generated by computational hardware [147], and the ability to model realistic fluid while attaining suitably accurate results with acceptable efficiency for computer graphics [53].

2.3.1 Validation

Clinicians will need to trust the validity and accuracy of the results, especially regarding the haemodynamic consequences, for any simulation to be useful. Determining the accuracy of CFD simulations requires some form of validation, this is especially true for methods to be considered for clinical practice. Verification for a numerical model can be defined as an assessment of the numerical accuracy of the results it produces, but validating computational models is more complicated as it requires a comparison between the results with those observed empirically in real environments [177]. Therefore, before these simulations can be used practically, the code should be validated by verifying that it can be used to reproduce sets of experimental data, theoretical solutions, or results from other established methods for benchmark problems or actual engineering problems [96]. Numerical fluid simulations are habitually validated by comparison to other data sets and other forms of benchmarking, for example, [68] draw upon multiple data sets to validate different aspects of the simulation in a series of benchmark test problems. The results of a simulation can be collated and compared against expected behaviours we would find in empirical observations, e.g. simple straight channel flows such as Couette and Poiseuille flow or more complex separated flow [1]. In contrast, research into graphical simulations, especially studies with real-time criteria, prioritise computational performance in favour of accuracy. In some cases, it is believed the constraints of acceptable performance prevent suitable validation; Müller, Schirm, and Teschner's 2004 real-time simulation found the number of particles was significantly too low to compare to measured data.

2.4 Clinical Decision Support

When a patient is admitted to the care of a clinician several sources are considered before deciding on a treatment plan, especially in cases involving invasive surgery.

Amongst other resources, healthcare professionals will bolster their own experience and training with patient records and medical recommendations. These recommendations are often curated by a society of experts consulting with the literature and statistics for a given discipline or pathology. Medical recommendations can be delivered with additional information such as a level of recommendation (Strong, Weak) and quality of evidence (Low, High) [27]. These qualities often align to produce a strong recommendation with a high quality of the supporting evidence, but the opposite can also be true, recommendations based on limited evidence [108]. Moreover, recommendations may be based on simple measurements unrelated to a patient's individual characteristics, e.g. the diameter of an aortic aneurysm [52].

Furthermore, while these recommendations may benefit some patients, they rely on the diligence of medical professionals with highly limited time and resources. Many errors may be alleviated by human knowledge and inspection, but it has been proposed that up to half of all medication errors are intimately linked with insufficient information about the patient and drug [92]. This type of error has prompted the development of real-time information systems to detect problems and alert responsible parties when issues are detected [15].

2.4.1 Clinical Decision Support Systems

Clinical decision support systems (CDSSs) are a type of software suite promoted as a method to reduce medical errors and increase health care quality and efficiency [180]. CDSSs support clinical decision-making by delivering evidence from scientific research and electronic health records to health care professionals, sometimes tailored to generate patient-specific guidance [95]. Historically, there have been few examples of CDSSs used in practice, potentially due to organisational and daunting technical challenges yet their use to facilitate evidence-based medicine promises to substantially improve health care quality [165]. Although findings are mixed, there

is sufficient evidence that CDSSs are effective in improving screening for CVD risk factors and clinician practices for CVD-related preventive care services, clinical tests, and treatments [123]

CDSS algorithms were developed to leverage more modern techniques like machine learning to aid in pre-operative decisions [94]. Researchers have developed these principles further by curating elements of computer simulation into CDSSs. For example, the finite element method has been used to provide thoracic surgeons a fast, accurate and simple tool for predicting the stress state of the trachea and the reduction in the ability to swallow after implantation thus helping in taking decisions during pre-operative planning of tracheal interventions [165]. However, there are limited examples of CDSSs integrating computer simulation and visualisation to benefit pre-operative decision making for cardiovascular surgery.

2.5 Hardware For Interactive 3D Visualisation

2.5.1 Virtual Reality and the zSpace Stereoscopic Monitor

3D visualisation is a major component of medical information, for example, when performing therapeutic procedures clinicians are viewing electronic representations of the patient's pathology, and this allows the disease processes and anatomy to be better understood. Volumetric medical image rendering is a method of extracting meaningful information from the data created by computerised imaging modalities such as CT and MRI [202]. The scanners required for advanced medical imaging are now so ubiquitous that they now appear in some operating theatres, these 'hybrid' theatres have equipment available during endovascular operations such as aneurysm repairs, peripheral arterial and venous re-vascularisation and gastrointestinal bleeding management [188], see Figure 2.11.

Figure 2.11 A hybrid operating theatre, combined operating theatre and interventional radiology suite (Source: BBC News and Coleman [16].)

In the 1990s new 3D immersive experiences were proposed to bring new visualisation and interaction solutions into the medical field, for example, optical see-through augmentation of ultrasound slices through a HMD [164]. Since the advent of medical imaging technology, the images produced have evolved from 2D projection to fully isotropic 3D images, and this evolution has made the traditional means of viewing such images inappropriate, especially when surgeons need an accurate 3D representation of the pathology to plan a treatment [202]. The increased availability of high-resolution volumetric imaging impacted healthcare and generated new opportunities to visualise and measure pathological structures, especially within the clinical areas where these imaging techniques are widely used for diagnosis and treatment planning [192]. It is proposed that we can improve visualisation and simulation by using VR to interact with objects as we would in real world, or embed a user into a computer-generated world, using devices such as head-mounted display goggles and data gloves [197]. Today these techniques are actively being researched and successfully applied to medical education, diagnosis, and therapy [76, 155, 158, 193].

(a) The Oculus Rift virtual reality head mounted display with Oculus Touch controllers (Source: Facebook Technologies, LLC. [49]).

(b) The HTC Vive virtual reality head mounted display with accompanying controllers and base stations for tracking (Source: HTC Corporation [67]).

Figure 2.12 Desktop VR HMD configurations.

The application of VR technology in medicine was once viewed as a gimmick and even potentially dangerous [107], but despite the criticisms, VR technology and research has continually flourished and produced a multitude of deliverable solutions within medicine, and related fields. We believe that giving the clinician the options to use immersive devices will only improve their experience, and potentially allow them to view their decisions through new perspectives. HMD VR configurations are currently popular in other domains. Modern examples of HMD's include the Oculus Rift and HTC Vive, see Figure 2.12. The use of HMDs by clinicians is not ideal, however, as they do not currently provide a collaborative experience. This would be certainly be unsuitable for many clinical working spaces and especially unsuited to operating theatres. Instead, transitional virtual reality devices will provide a more suitable environment for clinical use. One alternative with potential in a hospital setting, is the

zSpace stereoscopic monitor [203], comprising of a fully featured passive stereoscopic monitor, glasses based head tracking, and equipped with an integrated 6 degrees of freedom stylus interface, see Figure 2.13. The passive stereoscopy featured in the zSpace monitor is created when two images are superimposed onto the same screen and viewed through the polarised glasses. The platform has proven to be practical in medical applications, it has previously been used to develop medical software, both by zSpace, Inc themselves and medical researchers [76].

Figure 2.13 The immersive zSpace 200 display, featuring 6 degrees of freedom stylus, passive stereoscopic glasses, tracking cameras and 3D monitor (Source: zSpace, Inc [203]).

Using an interactive platform will be an easier transition for clinicians, as little is known about the computer literacy of the medical profession, but computer anxiety and alienation are real problems [107]. It is also more practical for surgical applications to target monitors with similar dimensions to those already used within operating theatres. Moreover, the immersive aspect can be shared to other viewers with the use of the zSpace zView system [205], in which the rendered image can be projected or mirrored

to a secondary monitor. The zView configuration also supports an 'augmented reality view' of the users screen, where the virtual objects are rendered as they appear to the zSpace user, see Figure 2.14. There are other technologies that offer a similar 'augmented' view of virtual objects such as the Microsoft Hololens [111] and Magic Leap [48], HMDs that project holograms into the user's vision, see Figure 2.15 .

Figure 2.14 The zSpace zView collaborative view mode, achievable with a video camera and secondary screen. The secondary monitor renders the virtual objects as they would appear to the primary zSpace user while wearing the stereoscopic glasses (Source: zSpace, Inc [206]).

Previous software systems operate using a conventional user interface (UI) as the medium of control given to a user when operating the system. The UI processes user actions and delivers it to the required computation and facilitates the delivery of the output. Most desktop user interfaces have used the same basic principles for the past decade or more, but with the advent of VR and ubiquitous mobile computing, 3D interfaces are becoming a more important area of research [90]. Developing upon this convention, a 3DUI delivers the control to users performing tasks directly in a 3D spatial context. Working within a 3D environment does not necessarily mean that a user is interacting in 3D, and conversely 3D interaction does not require 3D input

(a) The Microsoft Hololens mixed reality device (Source: Microsoft [111]).

(b) The Magic Leap mixed reality device (Source: [48]).

Figure 2.15 Mixed reality head mounted displays.

devices, e.g. by using a 2D non-stereoscopic display with multi-touch capability, the user performs a 2D gesture to move along the Z axis [13]. 3DUI's are increasingly important with the advances in 3D immersive hardware, but the fundamental issues remain unaltered; real-world tasks are performed in 3D, 3D interaction is difficult to solve and still requires better usability [21].

2.6 Summary

There is no clear consensus on real-time software but there is a trend to recommend frame rates near to 30 FPS. Although it has been difficult to create real-time CFD simulations in the past, modern graphical hardware programming has created new opportunities for high performance programming. However, there is still a precarious balance when optimising performance against the accuracy of the simulation, as it is difficult to validate on small numbers of particles.

Blood has been an active area of research for simulation but there are few examples of interactive real-time patient-specific applications. Blood behaviour is often simplified by assuming laminar flow, slip conditions, and with the application of Newtonian models. These assumptions have been generally found to have good parity with

experimental data. The SPH method should be an effective approach to simulate under these conditions, as it has seen constant improvement and effective boundary methods are being developed. Validation has also been a problem within blood simulation. It is difficult, and sometimes invasive, to gain the required *in vivo* data from patients. There is a clear need for a real-time 3D blood simulation to be developed with interactive behaviour and for the output to be subject to novel validation until *in vivo* data can be applied. The validation needs to prove that the results are representative of reality.

There is great potential for real-time blood simulation to be used as a pre-operative decision making tool to improve efficiency and patient outcomes. As a CDSS, the software will be required to integrate into existing practice, which provides clinicians with new perspectives on historically difficult problems. With patient specific results in a time frame suitable to affect the decision-making process, the potential for costs saved and patient care could be massive. The exploration also would provide a necessary investigation into the application of real-time SPH into the medical domain, if its accuracy and visual continuity is suitable for professional and commercial use, and if it can be developed to become superior to competing methods.

Chapter 3

Implementation of a GPU accelerated Blood Flow Simulation

3.1 Simulation and Objectives

A prototype blood flow simulation application was developed to simulate blood flow through real patient geometry in real-time, with integrated user interaction to form certain diagnostic scenarios i.e. blocking the flow and evaluating the effects of re-circulation, including collateral circulation. The requirements and specifications for the application were developed in part from past and current collaborations [142] with subject specialists, interventional cardiologists (Prof Michael Rees, Ysbyty Glan Clwyd), interventional radiologists (Prof Derek Gould, Royal Liverpool Hospital) and clinical engineers (Dr Thien Howe, Royal Liverpool Hospital). We aim to reduce the effect of having disparate operator-dependant steps including image-segmentation, model preparation, mesh generation and simulation, by creating a more standard integrated workflow. The application is also developed to integrate well with interactive 3D technologies, primarily the zSpace stereoscopic monitor, as an added benefit to those who have access to such technologies. If successful, the simulation could be

integrated alongside pre-existing clinical decision-making strategies, granting a higher depth of understanding and patient specific care, by running through any number of potential outcomes to treatment. Secondly, there is an opportunity to increase the utilisation of interactive 3D displays in the medical domain and the operating theatre. By providing the option for a clinician to access the simulation within an immersive environment, we only envision an increase in uptake and positive experiences when interacting with the simulation.

The key problems identified to be addressed are: performance and achieving simulation results at interactive rates, the compatibility of real-time interaction with the simulation results, the application of real three-dimensional patient geometry, with respects to the boundary and the inlet/outlet conditions, and the validation of simulation results. To simplify the incredibly complex circumstances of blood flow, a steady laminar flow with a solid boundary is assumed. Although some natural behaviours are lost, e.g. an elastic muscular boundary in vessels and the pulsatile flow created by the rhythmic beating of the heart, the approximation is capable of providing suitable credibility despite these limitations on the accuracy. These assumptions are common across the literature surveyed in Section 2.2. Furthermore, elastic and reactive behaviours are an active area of simulation research [3, 81, 135] and including these features into this development is outside the scope of this research.

3.2 Patient-specific Geometry

Patient specific care goals are an increasing priority in the medical profession, especially for conditions such as CHD, which will naturally involve case by case treatment strategies. As the logical progression from viewing 3D medical images with the use of computer graphics, the application of real patient geometry is a great opportunity to

increase the value of any simulation results. By targeting the patient specific pathology, simulation results will grant a greater insight than any generic test cases. The results can be used to design the optimal treatment strategies for the patient, divining valuable information on physical consequences that cannot otherwise be measured. Additionally, results based upon true to life geometry improves the likelihood of generating clinical trust in the results. The use of the patient geometry would decrease the time required for the simulation pipeline. As medical imaging would likely have already been taken, or generated when required in a hybrid theatre, there is no requirement to fabricate life-like geometry for various scenarios. Generation of intricate geometry would incur significant additional costs for each case, and the deliverable product would be created to enable a meshed simulation method, such as the Finite Element Method (FEM). Meshing and modelling steps are instead immediately diminished with the introduction of the meshless SPH method, rather than discretise the whole domain, the application can use the results of automated image segmentation to define a boundary for the particles.

3.2.1 Geometry Processing

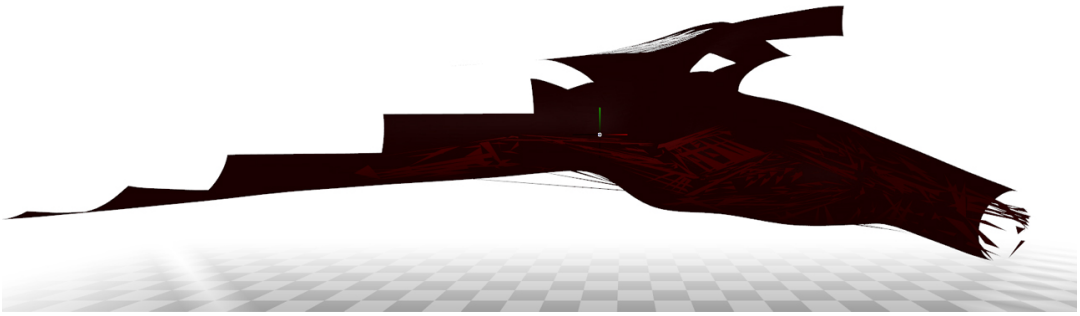
Real patient geometry is the largely the result of magnetic resonance imaging (MRI) and computerised tomography (CT) imagery. There is deviation in the outputs from medical image processing, but previously established routes of geometry transformation have reached maturity, software solutions, such as ITK-SNAP [201] or VMTK [105], were developed to automate processes in the extraction of volumetric data. These applications effectively model the anatomical structures into readily usable data formats, 3D meshes can be generated from 2D images or a collection of 2D slices. The conversion is very successful but there is a potential for inaccuracy in the output and can require some manual editing to correct any misinterpreted regions or to cater the output to an information system's requirements [104]. The

processing of these computer aided design (CAD) deliverables and medical scans is not standardised throughout every implementation and as a result there are many file types to accommodate, we chose to largely focus on the STL (stereolithography) and OBJ file type.

Utilising such complicated geometry from disparate sources begets new challenges. The process of importing mesh geometry is complicated by the number of file formats and the potential for deviation within the format itself, i.e. application or vendor specific file formatting. This is exemplified in the STL format, as files can be encoded in ASCII (American Standard Code for Information Interchange), a standard format for character encoding, or in a binary format. C++ has a mature ecosystem of open-source libraries, including those to facilitate loading CAD file data. Using an external library can potentially accelerate development, but as is the case with the files themselves, libraries can introduce their own potential for deviation.

Deviations when loading geometry often present themselves as holes in the mesh, see Figure 3.1a. This can be caused by several corruptions such as, invalid indices, missing vertices and an unexpected winding order (the order the vertices of a triangle are defined). Most of these problems can be solved by re-meshing, the process of finding a new, hopefully improved, discrete triangle mesh representation of the geometry. To automate this process for clinicians without the relevant experience to manually remesh the geometry, we attempt to clean the mesh during the preprocessing phase. In this time, we triangulate the mesh, i.e. remove any non-triangle primitives, ensure the winding order is anticlockwise, the default for OpenGL, and merge any duplication. Ideally these techniques will remove the need for most manually mesh manipulation, and meshes will render correctly despite errors, see Figure 3.1b.

During simulation, we make certain assumptions about the input geometry. We assume that the geometry supplied to the application by the user is a 3D triangle mesh with open inlets and outlets, we define this as geometry that contains no triangle faces



(a) Rendering realistic patient geometry with severe mesh errors.

(b) Rendering realistic patient geometry after the mesh errors have been corrected.

Figure 3.1 Medical image data triangle mesh processing.

covering these areas, see Figure 3.2. In the scenarios we developed, VMTK's external command line application was used to crop closed outlet triangle faces from some meshes. However, these external preprocessing steps are time-consuming and can increase administrative effort substantially, this additional responsibility would fall to clinicians and their support staff. It is favourable to instead deal with as many edge cases and precomputations within the applications internal preprocessing phase, i.e. before the simulation has begun, but after the geometry has been selected. We decided to leave the clipping of the mesh inlet and outlet to the user, as this sort of 3D manipulation can be quite difficult and may be best left to an experienced medical imaging professional with their preferred 3D mesh editing software. Although we were able to automate the clipping of the mesh extremities, given a defined length, we felt it could potentially remove elements of the patient geometry that were important

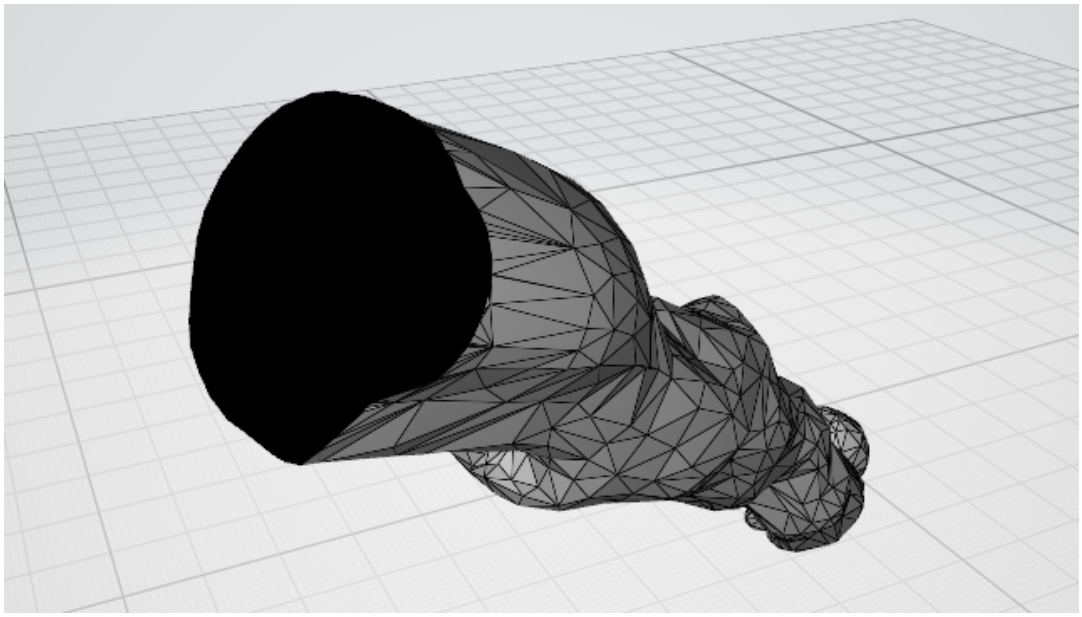


Figure 3.2 One example of patient geometry that was used in our simulation. The geometry is rendered with each triangle face coloured based upon its corresponding mesh normal facing outward. Consequently, black triangles are inside the geometry.

to any given scenario, and could take some level of control away from the heath care professionals.

3.2.2 The Centre line

The centreline, or curve-skeleton when applied to irregular geometry, is a line-like representation of a 3D object, consisting only of curves [32]. Centrelines are natural synthetic descriptors of interconnecting geometries such as bifurcating vessels [5]. The characteristic tubular shape of blood vessels structures has inspired methods of extracting centrelines from medical images. Centrelines are useful in image processing operations such as edge detection and segmentation. Moreover, conventional medical uses of the centreline include detection of stenosis, aneurysms or vessel wall calcifications [32].

The centreline calculated from the patient geometry is a pivotal resource for the simulation, as powerful descriptors of the vessel shapes, the centreline is drawn upon

extensively throughout the execution of the simulation and during preprocessing. The centreline is used to define and control the inlet when seeding particles into the geometry. The centreline also defines where blockages can lie during user interaction, and we can also use it to apply an external force to simulate the flow effects of systole, the beating of the heart, or simply to direct flow through complex geometry.

Generating the vertices of a centre line for simple straight channel tubes is not a difficult problem, however, with real patient geometry there is no clear centre of mass apparent, especially in cases where the vessel also has the potential to include irregularly shaped regions due to pathology, and so it can be difficult to isolate. We generate the centre lines during the preprocessing phase.

Using the VMTK [105] software library, a tool designed with patient geometry in mind, we define the inlet and outlet of the mesh and generate centre lines in a reasonable time for any patient geometry. The implementation in VMTK deals with the computation of centre lines starting from surface models and has the advantage that it is well characterised mathematically and quite stable to perturbations on the surface [106].

The centreline we produce is composed of a collection of vectors that crucially form the component lines, which can be seen rendered in Figure 3.3. By inspecting the first of these lines, we can see where the vessel roughly begins, the vessel inlet, and using the direction of the vector created by the first line segment we determine the direction the flow will be following. This directional hint could also be used for complex geometrical scenarios where the flow cannot be influenced along one axis by a gravity like force. This hint emulates the pressure differential that would be found in a full system, rather than the segment that the simulation operates on. This information was useful during the initial seeding of the particles, i.e. generating the particles, subsequently positioning them within the inlet region, applying some base default conditions and properties. In addition, the particles which then leave the vessel

Figure 3.3 Imported patient geometry rendered with front face culling, see Section 3.6 and the calculated centre line, seen in white running along the full length of the vessel segment.

through the outlet have reached the end of their lifetime and can be reintroduced into the vessel according to a similar sequence, which we refer to as re-seeding, the primary difference being the quantity of particles generated. The default conditions for particles generated into the inlet include:

1. An identifier for later comparisons.

2. A positive quantity for the particle density.

3. Acceleration vector in the direction of the general flow, calculated from a user submitted magnitude and the direction from the centreline unit vector.

3.3 Smoothed Particle Hydrodynamics

Calculating and integrating the dynamic properties of the particles follows the general algorithm shown in Algorithm 1. The algorithm is conceptually simple, with the eventual goal of integrating the particles into their new positions. Each computation generally requires the results of the previous stage, preventing greater concurrency. The implementation of the stages in the SPH algorithm are described in the following sections.

Algorithm 1: SPH algorithm to simulate fluid behaviour in discrete time steps.

```
1  Define initial particle positioning;
2  while the simulation is running do
3      Map each particle to a grid cell;
4      Sort the particle array by grid location;
5      foreach particle do
6          Calculate particle density;
7          Calculate particle pressure;
8      end
9      foreach particle do
10         Compute the forces acting on each particle;
11         Integrate the particle position over time;
12         Push the particle data to the frame buffer;
13     end
14 end
```

3.3.1 Initial Simulation Conditions

When establishing the position of all the individual particles within the inlet, the particle configuration should be as close as possible to the configuration that will appear after the simulation starts [43]. The generation of initial positions, or seeding, is underrepresented in fluid based SPH as the method is often used for free surface flows, where the initial position is not the focus of investigation, or will be manually specified for each domain. The simplest method of arranging particles is within a lattice, the interpolation properties are well-known, and it is programmatically simple to generate. We opted to implement a method of hexagonal close packing (HCP). HCP is an effective packing scheme for uniform spheres, with an optimal density of 74% [43]. The structure is generated with alternating layers of hexagonal lattice planes, every second layer is identical, resulting in an ABAB pattern, seen in Figure 3.4. This seeding pattern greatly reduced the impact of preferred axis issues encountered by previous cubic lattice seeding, without causing any clumping effects seen when randomising a cubic lattice. The initial cubic lattice seeding displayed a directional bias along each axis, and occasionally break into disparate 'sheets' which would begin to clump. Before implementing HCP, we added an element of randomness into the positioning of the cubic lattice, by offsetting each particle randomly within a fixed range. The previous issues were unresolved and the equilibrium became unstable on iterations where the randomness created areas of low density, especially towards the edges of the seeding pool.

3.3.2 Density

Particle density is a prerequisite in the computation of particle pressure. Each particle's density must be found before the pressure calculation can be made, a small bottleneck in the general SPH algorithm. As the fluid volume is represented by a collection of

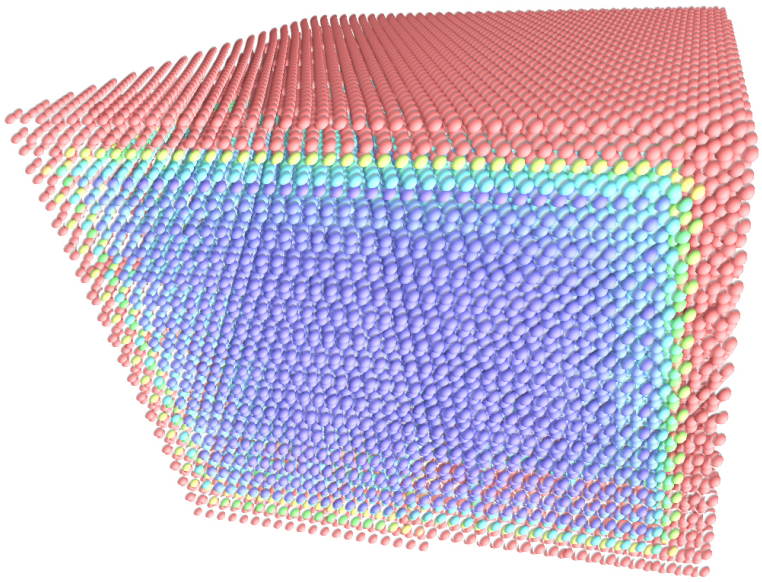


Figure 3.4 Particle lattice generated with hexagonal close packing, an optimal uniform sphere packing scheme.

discrete particles, each particle (i) must represent a volume. This volume is based upon their mass (m) and density (ρ), see Equation 3.1.

$$V_i = m_i/\rho_i \tag{3.1}$$

In our simulation the mass is the same for every particle and set by the user or a preprocessing calculation. Finding the density of an individual particle, effectively one point in the fluid, is done by summing the contributions of all the particles in their neighbourhood (smoothing kernel radius). This is seen in Equation 2.1 with the prominence of density and mass. Furthermore, through substitution into Equation 2.1, we calculate the density for particle i with Equation 3.2.

$$\rho_i = \sum_j m_j W(r_i - r_j, h) \tag{3.2}$$

3.3.3 Pressure

Once the density of every particle is known, the pressure of each particle is calculated with a state equation. In our case we used a modified form of the ideal gas state equation [189], a common and stable approximation exhibiting repulsive forces in fluids at high time-steps [42, 119]. The pressure for a particle is the product of the particle density ρ, and a given user configurable stiffness constant k, see Equation 3.3.

$$p = k\rho \tag{3.3}$$

Instead of the purely repulsive pressure p between particles generated by the original equation, the modification creates attraction-repulsion forces with the divergence of rest density ρ^0, an expected value for the fluid studied, against calculation particle density, see Equation 3.4. When applied, a greater stiffness constant reduces the compressibility of the fluid, at the cost of smaller time steps [70].

$$p = k(\rho - \rho^0) \tag{3.4}$$

3.3.4 Forces

Once the density and pressure of the particle collection is available, forces acting on the fluid are computed. These forces are then used as the components to interpolate the particle acceleration. For example, the internal pressure force can be calculated by applying the general SPH formulation as in Equation 2.1, see Equation 3.5. Then the particle acceleration can be found as $f_i^{pressure} + f_i^{gravity} + f_i^{viscosity} + f_i^{external}$. The $f_i^{external}$ term indicating any other domain specific forces acting upon the fluid.

$$f_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2p_j} \Delta W(r_i - r_j, h) \tag{3.5}$$

3.3.5 External Forces

Blood flows to the lower regions of the body with the assistance of gravity [64], and Gravitational and hydrostatic effects are very important for most orientations [11]. SPH simulations generally contain an external force to mimic the effects of gravity. The same effect is often used to initiate a general direction to flow through an external force. As we cannot assume the orientation of the 3D vessel, or the magnitude of the force itself, we allow the user to specify the 3D vector which will be applied as gravity during each time step. Following that, the centreline directional force can be applied to the initial seeding area or more widely through the domain. A constant gravity like force does not emulate the pressure differential found in full closed systems, resulting in some branches encountering irregular flow in very complex geometry. This is certainly true in the case of a vessel segment that has multiple inlets that all merge into the same channel. With one constant gravitational force it would be possible for the flow to be pushed backwards into the regions along the same axis as the gravitational force.

3.3.6 Integration

We have implemented the Leap-frog integration scheme, see Equation 3.6 and Equation 3.7. The leap-frog scheme is a popular explicit method and achieves second order compared to first order Euler integration. The method earns its name as the position and velocity of particles are updated at interleaved steps, t, i.e. velocity, v, is updated at half steps and positions, x, at integer steps.

$$v_i^{n+\frac{1}{2}} = v_i^{+n\frac{1}{2}} + a_i \Delta t \tag{3.6}$$

$$x_i^{n+1} = x_i + v_i^{n+\frac{1}{2}} \Delta t \tag{3.7}$$

3.4 Neighbourhood Search

3.4.1 Uniform Grid

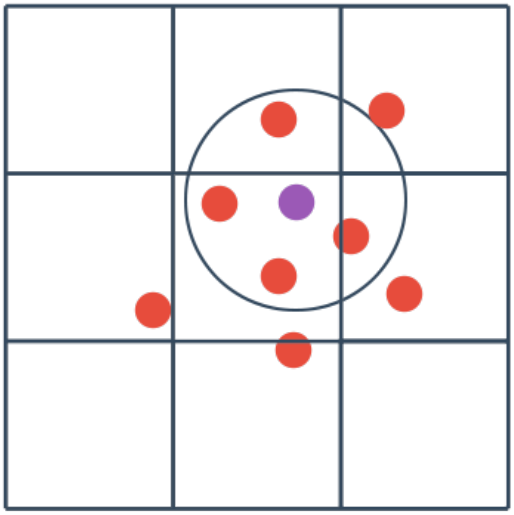


Figure 3.5 Illustration of how particles in the SPH method may fall within a uniform grid and its relation to the smoothing kernel radius, drawn as a circle around the particle in purple.

To fulfil the kernel weighting component of the SPH method, during each step of the simulation each particle has to collect and interact with a list of applicable neighbours, those particles that are within the smoothing radius. Gathering this list of particles by brute force would be far too slow for any type of SPH simulation, the problem is akin to the K nearest neighbours problem's complexity, naive solutions can reach $O(n^2)$ complexity. Instead, the simulation spatial environment is sub-divided into a uniform grid, the size of each cell is equal to the kernel length so that we can safely cover the kernel radius by considering a fixed number of cells, see Figure 3.5.

During each step of the simulation, every particle is mapped into the grid, using a simple hashing function, which effectively flattens the grid structure from a 3D array into a 1D array, allowing for optimised access patterns on the GPU. This flattening

process is visualised in Figure 3.6. This grid reference is then used to identify which particles are within the same, or adjacent cells. The grid is quickly constructed and resolves the complexity of accessing particles to O(1), while hierarchical data structures are typically accessed in O(log n).

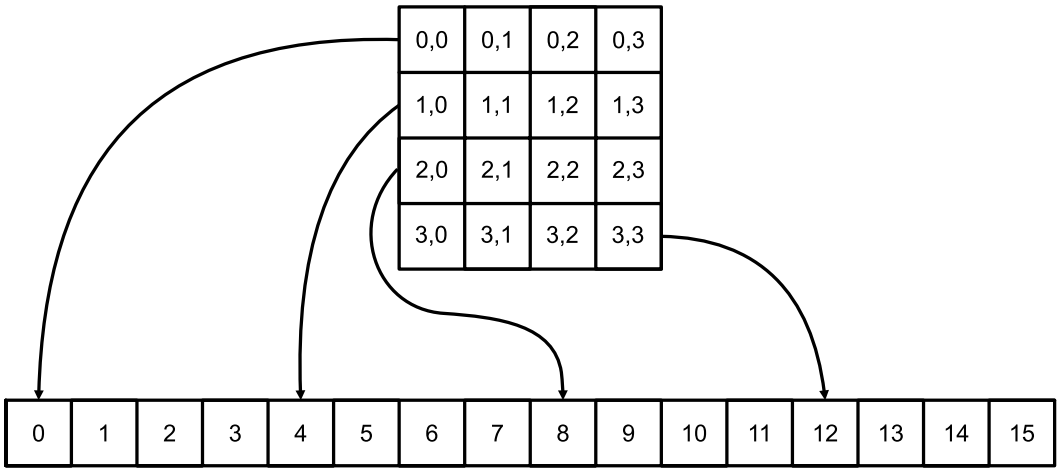


Figure 3.6 Discretising a 2D dimensional spatial grid into a 1D indexed array in memory. Cartesian coordinates are visible on the 2D grid, which are then translated into array indices shown on the 1D array (Source: derived from [167])

Figure 3.7 demonstrates the grid in action, by colouring each particle according to their grid cell reference. The simulation area is predefined by the scale of the patient geometry, with some padding added around the geometry to account for the boundary particles. As the size will always be defined we did not need to enforce a more compact or memory conserving hashing function. Uniform grids often suffer from inefficient memory utilisation, as depending on the domain characteristics the fluid will generally trend into one area, which may only be a small region of the full domain. This inefficient memory pattern can also impact performance to due higher memory transfers [62]. In an attempt to mitigate this, we avoided allocating memory for every grid cell, the grid is instead implemented conceptually. Particles are mapped to their grid cell corresponding with their position in the domain, subsequently the particles are sorted to imitate the grid layout.

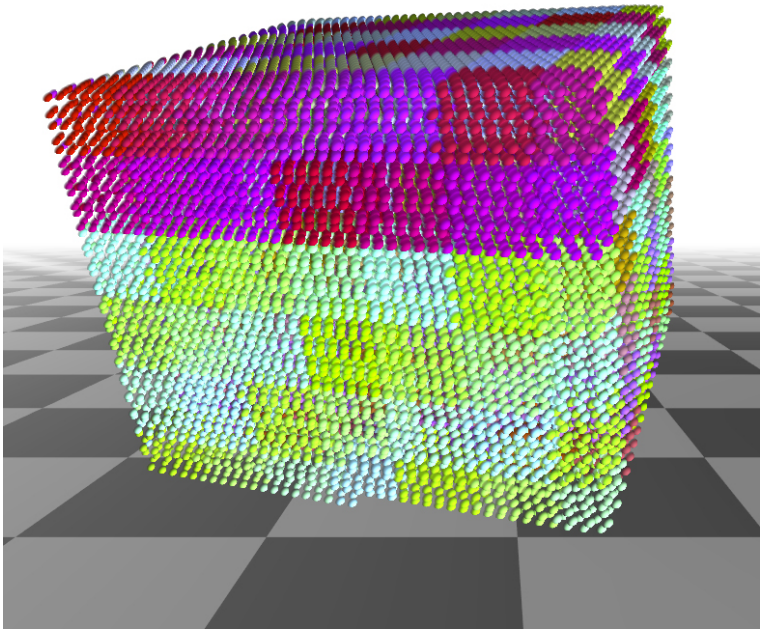


Figure 3.7 SPH particles coloured by their grid cell reference.

3.4.2 Discretisation

The particles are sorted by a conventional radix sort, using their flattened 1D grid reference as the key. Multidimensional access methods are costly compared to one-dimensional cases because there is no ordering that preserves spatial locality. The idea is to develop a single numeric index on a one-dimensional space for each point in multidimensional space [117]. By computing the prefix sum of the particle array, we map the grid into memory as a series of indices to particles, each cell as a reference to the first and last particle within the cell. This develops a single index in a 1D space for each cell in the 3D simulation domain.

Because the implementation uses a 1D index for a 3D domain, if a line was drawn between each cell in index order it would draw a uninterrupted line or curve through the 3D domain. Due to this indexing system, iteration over these cells is

very straightforward and in some cases we are able to iterate over multiple cells in one iteration by taking advantage of the compact 1D array indexing. For example, in Source Code 3.1, the look up array contains three vectors in each index, in the simulation our 1D indexing method allows us to iterate over the particles three grid cells that are adjacent in memory.

As a small optimisation, we help to unroll the loop over the grid cells by creating a look up table. We know that we will be observing 27 cells for each particle, so we were able to predefine the previous three loops into one, while also defining which cells can be accessed together. The processes also makes the simulation easier to maintain, the difference can be seen when comparing Source Code 3.1 to Source Code 3.2.

Source Code 3.1 3D iteration of the uniform grid. Three nested loops, one for each dimension of the grid.

```
1 for(int x = -1; x < 2; x++)
2    for(int y = -1; y < 2; y++)
3        for(int z = -1; z < 2; z++)
4            //Access neighbouring cell
```

The simulation can also run the neighbourhood search in a limited capacity. By limiting the maximum neighbours in each cell to use in the computation, we can increase the performance relative to the number of total particles. There is a potential loss of accuracy as the particles in each cell will be considered in an arbitrary order, and by disregarding a significant number of particles results will be affected somewhat. The increase in performance allows for a larger total number of particles. However, it is difficult to gauge the difference when the total particles are always relatively low in a real-time simulation, and SPH simulations are inherently non-deterministic, there will always be some small fluctuations within the results.

Source Code 3.2 1D iteration of the uniform grid. Iteration is facilitated by a look up table (LUT). The items in the LUT transform the current particle 3D grid reference, which after then flattening to a 1D index can access a grid cell.

```
__device__ const int3_triplet loop_lut[] = {
        {{-1,-1,-1},{ 0,-1,-1 },{1,-1,-1}},
        {{-1,0,-1},{ 0,0,-1 },{1,0,-1}},
        {{-1,1,-1},{ 0,1,-1 },{1,1,-1}},
        {{-1,-1,0},{ 0,-1,0 },{1,-1,0}},
        {{-1,0,0},{ 0,0,0 },{1,0,0}},
        {{-1,1,0},{ 0,1,0 },{1,1,0}},
        {{-1,-1,1},{ 0,-1,1 },{1,-1,1}},
        {{-1,0,1},{ 0,0,1 },{1,0,1}},
        {{-1,1,1},{ 0,1,1 },{1,1,1}},
};

const uint3 cc = calc_grid_cell(particle_position);
for (const auto i : loop_lut) {
    const uint adj_cell = calc_grid_hash({ cc + i.x });
    const uint adj_end = calc_grid_hash({ cc + i.z });
    const int cell_end = cell_begin_end[adj_end].y;
    int cell_iterator = cell_begin_end[adj_cell].x;
    while (cell_iterator != cell_end)
    //Access neighbouring cells
}
```

After the particles were sorted in-place by their grid cell reference, they needed to be indexed so that their access was straight-forward during particle iteration. First, a histogram is calculated, which creates an array of values that contains the quantity of particles in each grid cell. Secondly, we take the prefix sum from the histogram, which creates the memory index for the beginning of each cell. Finally, we take the prefix sum and generate an array of pairs containing the position of first and last particle in each cell, see Figure 3.8. This allows us to quickly query the particle array on each step without having to do any location calculations within the loop.

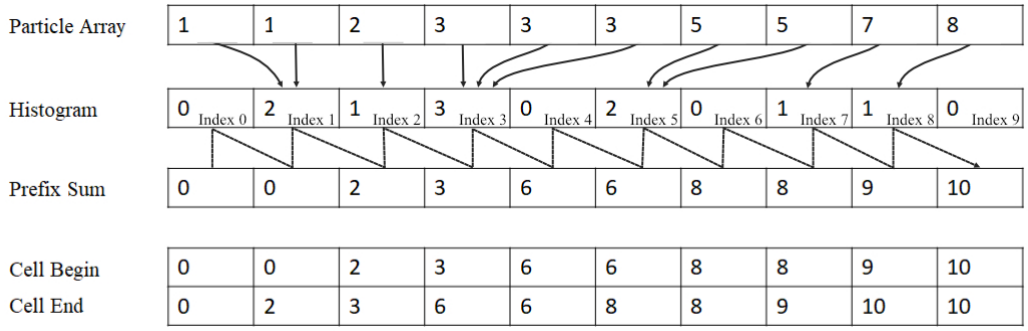


Figure 3.8 Example of the process to index the particle array. In this example the particle array consists only of the grid reference property, a discrete form of a particles position in the domain.

3.4.3 Smoothing Kernels

The 6th Polynomial Kernel

We use the 6th Polynomial (Poly6) kernel [119], which can be seen in Equation 3.8, for the calculation of the dynamic density of the fluid at each time step. A large portion of the kernel function W can be precalculated during the simulation preprocessing phase. The decision structure within the kernel describes the relationship defined by the kernel radius and its weight. No contribution is given by the particle itself, when r the distance between two particles is equal to zero, or by particles outside of the smoothing kernel radius, h. With the exception of ignoring any contribution to

particles outside of the kernel radius with the comparison of r and h, the Poly6 kernel does not have any branching logic, avoiding the potential of divergent CUDA threads, which limit the efficient use of computational resources. Secondly, the kernel was designed to be calculated using r^2, this eliminates a costly square root calculation during one of the most frequently occurring CUDA kernels.

$$Wpoly6(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - ||r||^2)^3 & 0 \leqslant ||r|| \leqslant h \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Desbrun's 'Spikey' Kernel

However, the Poly6 kernel is not appropriate for all of the fluid's dynamic properties. The biggest limiting factor with its use is the occurrence of clumping when calculating the pressure when the distance between particles positions r is closing, the repulsive factor diminishes because the gradient of the kernel approaches zero at the centre, see Figure 3.9a. Instead we implemented Desbrun's spikey kernel [42] function $Wspikey$ to provide the required repulsive forces and a resolution to the particles clumping behaviour within the kernel radius h. The kernel provides a more accurate representation of pressure with a non vanishing gradient near the centre, see Figure 3.9b.

$$Wspikey(r,h) = -\frac{45}{\pi h^6} \begin{cases} \frac{r}{||r||}(h - ||r||)^2 & 0 \leqslant ||r|| \leqslant h \\ 0 & \end{cases} \tag{3.9}$$

Müllers Viscosity Kernel

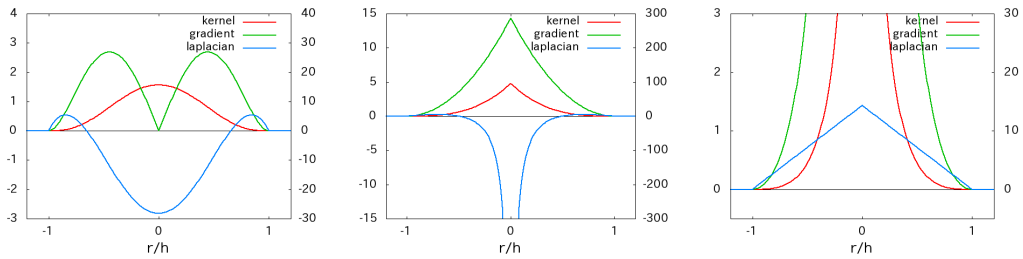
Similarly, when computing the inner tension of the fluid, viscosity, the poly6 kernel can produce negative velocities, which would cause an increase in their relative velocities. As the viscosity would normally decrease the fluids kinetic energy by transferring it into heat, the viscosity should only produce a smoothing effect on

the fluid velocity, irrespective of the distance between the particles r. The effect is amplified on simulations with a smaller quantity of particles, hence the problem in real-time simulations. Instead we employ Muller's viscosity kernel [119], $Wviscosity$, to avoid these negative velocities, as it contributes only positive results in any given circumstance within the kernel radius h, see Figure 3.9c.

$$Wviscosity(r,h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \leqslant ||r|| \leqslant h \\ 0 \end{cases} \tag{3.10}$$

3.5 The Boundary Condition

The boundary condition is an important and difficult aspect of modelling fluid flow, especially within the SPH model, in fact, it may be the most difficult aspect of the simulation. The first objective of the boundary is how to avoid fluid particles from penetrating solid boundaries, the potential implementation of a fluid-to-solid interaction solver. Secondly, the boundary needs to aid in the construction of certain fluid behaviours and characteristics, notably the slip/no-slip conditions of the fluid interaction. Slip and no-slip conditions are the two main conditions for the flow velocity of a viscous fluid. The no-slip condition stipulates that fluid in direct contact with solid will have identical velocity of the boundary. Particles in this condition



(a) 6th Polynomial (Poly6) Kernel [119]. (b) Debrun's Spikey Kernel [42]. (c) Müller's Viscosity kernel [119].

Figure 3.9 SPH Smoothing Kernels (Source: PukiWiki [145]).

would be seen to stick along the edge of the boundary. The slip condition assumes the opposite, a relative movement between the fluid and the boundary [149]. The formal definition of SPH contains no definition of a boundary. Naive or toy implementations of SPH designate a solid boundary, based upon a distance from the origin on each axis. When the particles collide or stray outside this box, the particles are moved back into the simulation area and the particle velocity component perpendicular to the surface is reversed. However, there has recently been a number of more accurate boundary paradigms in research and engineering applications, namely force-based, fixed particle and mirror boundaries.

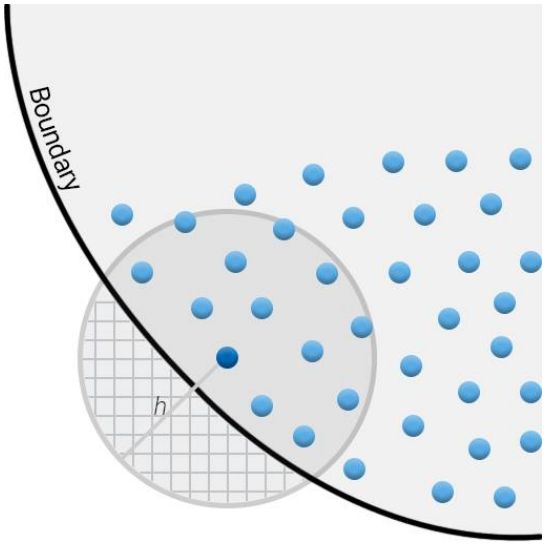


Figure 3.10 Kernel occupancy near a rigid body without kernel correction.

Force based boundary

Initiated by Monaghan [116], the boundary is defined in the simulation as a collection of particles, or sometimes as a triangle mesh [119], they exert central repulsive forces onto the particles they contact, the implementation of which is guided by the known interaction of molecules. As such, the force based method is one of the easiest methods to implement. The force generated is similar to the Lennard-Jones potential force between atoms, i.e. both a repulsive and attractive force is created within a

fixed length, and the magnitude of that force is determined by the distance from the boundary. Those particles that approach the boundary are forced to have an empty region within the kernel radius, leading to problems with instability in those regions and non-physical results, e.g. 'clumping', particles unnaturally receding into one another, was experienced in particles as they encounter negative pressures near the boundary. The problem can be seen in Figure 3.10, the cross hatched region has no contribution to the dynamic property calculation within the SPH model.

Dynamic Particles

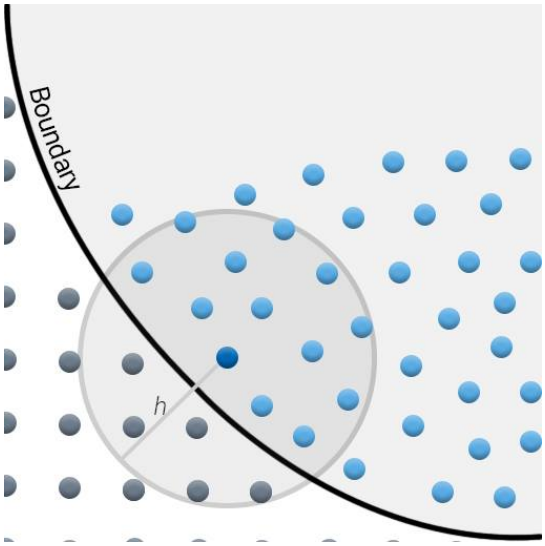


Figure 3.11 Kernel occupancy near a rigid body with dynamic particles.

Sometimes referred to as fixed boundary particles, this boundary technique would have the simulation include particles with a fixed position upon and outside of the boundary, which is generally predefined. Crucially, these fixed particles have the same dynamic properties as the fluid. Consequently, these particles engage in the same equations of continuity and state as the fluid particles but without altering their positions within the SPH method [36]. The positions of the particles can be moved outside of the SPH formulation to implement moving boundaries and similar behaviour. This is very useful within SPH as this boundary technique does

not create any additional algorithmic steps, and instead can be nested into the pre-existing structures of particle iteration. Increasing the particle count in this way does hinder the performance of the system, but accurately portraying the boundary would be best served by generating the maximum quantity of particles. This requires careful management to alter the simulation parameters for larger or more complicated geometric structures.

Mirror Particles

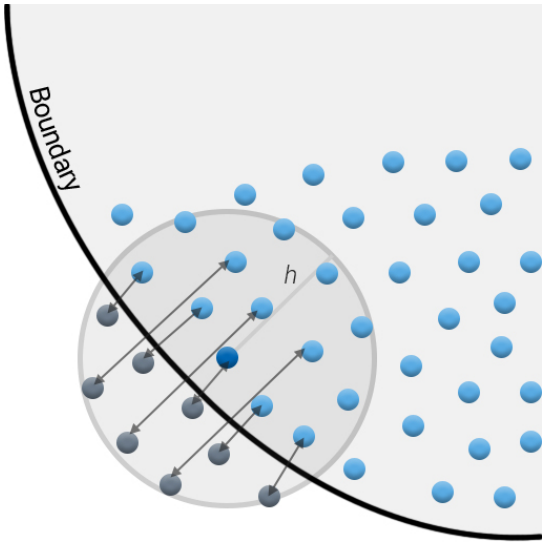


Figure 3.12 Kernel occupancy near a rigid body with mirror particles.

Sometimes referred to as ghost particles, these particles are similar to dynamic particles that have a fixed position. However, rather than having a predefined position they are generated based upon the neighbourhood of the particle in contact [37]. When the real particle is close to the boundary, a particle is mirrored outside, with the same density and pressure but an opposite velocity [36]. Unlike dynamic particles, the implementation can be more involved as the number and position of boundary particles will change on each iteration. This type of boundary could be more suitable for a moving boundary, or one which reacts to the flow itself, as the boundary will consistently regenerate on each frame.

Biological structures are noted to have physical features that are not typically found standard mechanical devices and fabrications e.g. elastic behaviour on outer membranes and internal reservoirs of liquid. Contractile matter can also occur within vessels, either as the muscle producing the pulsatile flow or as a result of the pressures generated by the blood flow. In other words, the vessel walls can be dynamic and alter the flow in some circumstances. Due to the constraints imposed by creating an interactive simulation we decided to avoid the elastic and deformation behaviour for the first implementation of our simulation.

3.5.1 Enforcing The Boundary

Initially we tested a force based boundary, and the system was fairly successful, resulting in good overall stability. This was expected as Müller [119] used this technique to demonstrate an excellent solid boundary at interactive rates, using the triangle faces of the mesh rather than arbitrary boundary points. This force based boundary provided a computationally cheap boundary, but at the cost of accuracy within the particles near the boundary. In fact, we began to lose the performance benefits of the force based boundary as we needed to include the boundary particles within the viscosity calculation to create a more accurate representation of any given slip condition. In our initial prototype we also encountered an unsatisfactory amount of particles escaping through the boundary. We attributed this loss mostly to problems with the source mesh having uneven and inconsistent triangle faces. We added additional SPH sub-steps to try to avoid the particles 'missing' the boundary i.e. by performing all of the SPH calculations multiple times in the same frame with a fraction of the total time step each. However, the performance of the simulation quickly diminished as frame time grew linearly. We also began to experiment with better boundary construction, see Section 3.5.2. Rather than invest more development

resources into the implementation of the force boundary, we considered the inaccuracy reason enough to transition the implementation into another method.

In an effort to increase the accuracy around the boundary, we implemented dynamic, or fixed, particles on the boundary. These particles along the edge of the boundary contributed to the particles dynamic property computations, mitigating the kernel occupancy issues with the force based boundary. The disadvantage of this technique is that it does effectively increase the particle count, although the particles are not included in the advection stage, they have to be added to the collection which will be iterated over each frame. Mirror particles were deemed to be less useful in our implementation as we favoured the performance benefits of generating our boundary's fixed particles in a preprocessing stage. Secondly, we aimed to generate blockages in the simulation area as a result of user input, the dynamic particle paradigm was the logical choice as any new blockages could be discretised into a collection of particles and added to the boundary particle collection, similarly preventing any new algorithmic steps to the SPH simulation.

3.5.2 Generating The Boundary

While generating the boundary particles, we initially attempted to use the raw vertices of the mesh, using these points seemed to be a convenient and inexpensive way of positioning the boundary particles. As these meshes are generated from medical imagery, the resulting mesh does not always have small, simple faces or uniform faces. Therefore, we naively subdivided the mesh, recursively measuring the edges of each triangle, and if they were greater than a predefined length we split the triangle into three smaller triangles. Unfortunately, this only served to increase the extent that the idiosyncrasies in the mesh would effect the boundary, particles could still be great distances apart when measured through one particular dimension, rather than the aggregate distance. We resolved to re-sample the surface as a preprocessing step,

using the Poisson Disk sampling method [198, 22]. Poisson Disk sampling is a dart throwing algorithm that is able to place points randomly, but to produce a uniform distribution upon a plane. The result was a more evenly distributed boundary on all examples tested.

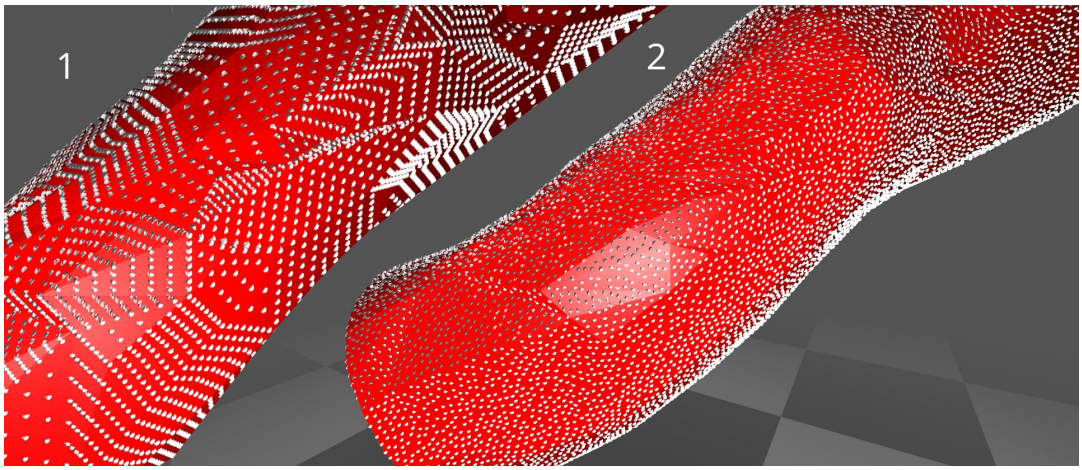


Figure 3.13 Patient geometry showing the differences in sampling between 1, a naive triangle sub sampling method and 2, a Poisson Disk picking algorithm. The output of each sampling method is represented as white particles on the surface of the mesh.

To then create additional layers of particles to fulfil the full kernel radius, we proceeded along the normal of each sampling point and creating additional points. The user can specify the number of boundary layers to be created. These layers will be positioned within one kernel length from the outside of the mesh. The downside of this method was the sheer amount of particles created. After sampling over the mesh, a significant number of particles are generated, by multiplying this quantity by an order of three or more, the quantities can become unmanageable in terms of performance. For example, when sampling a simple straight channel mesh with a configured value radius of 0.02 (the allowable distance between samples in the Poisson Disk algorithm) and generating three layers of boundary particles, 216,441 particles are generated by the algorithm and can be seen in Figure 3.14. Some user configuration of the Poisson Disk radius and quantity of boundary layers may be required on complex or large geometry which generates excessive amounts of boundary particles. As can

be seen in Table 3.2, there is some degradation in performance as the particle count increases, this includes the increase in boundary particles and may limit the simulation performance on large geometry. The limitation does not encumber the prototype greatly at this stage as we expect the simulation to occupy selected smaller regions of geometry, e.g. the abdominal aortic bifurcation, or the carotid bifurcation.

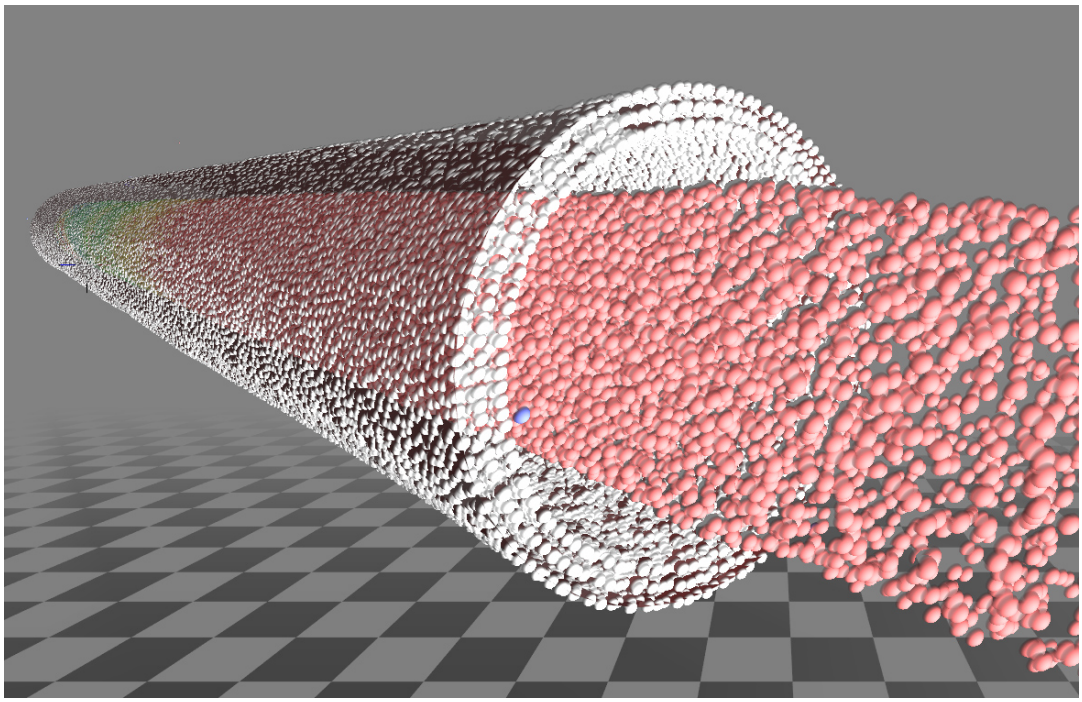


Figure 3.14 Artificial geometry highlighting the quantity of boundary particles (white) that can be generated on a straight channel tube. This example generated 216,441 boundary particles with a Poisson Disk radius of 0.02 and three layers of boundary particles. The particles are coloured by velocity, from the fastest moving in red, transitioning to slower speeds as yellow, green, and the slowest in blue.

3.6 Real-Time Rendering

Compared to modern high fidelity graphics applications, like video games and sophisticated CAD software, we needed to render a comparatively small amount of content. We needed to render some interface elements, menus, stylus pointer etc, but the crucial elements for visualisation were the particles and a triangle mesh (patient

Source Code 3.3 Loading the vessel boundary into the SPH simulation.

```
void load_boundary(float* boundary, const size_t boundary_size, float*
  ↪ boundary_normals, const size_t normal_size, const float radius)
{
  const float layer_count = h_params_.boundary_layers;
  const float spacing = radius / layer_count;
  size_t size = boundary_size * static_cast<size_t>(layer_count);
  std::vector<float3> temp_pos, temp_norm;
  std::vector<int> grid_ref;
  temp_pos.reserve(size);
  grid_ref.reserve(size);
  assert(boundary_size == normal_size);
  for (size_t i = 0; i < boundary_size; i++) {
    const size_t index = i * 3;
    const auto bound = make_float3(boundary[index], boundary[index+1],
     ↪ boundary[index+2]);
    const auto norm = normalize(make_float3(boundary_normals[index],
     ↪ boundary_normals[index + 1], boundary_normals[index + 2]));
    for (size_t j = 0; j < layer_count; j++)
    {  //advance along normal and create particle
      const auto val = bound + norm * static_cast<float>(spacing * j);
      temp_pos.push_back(val);
      temp_norm.push_back(norm);
      grid_ref.push_back(calc_grid_hash(calc_grid_cell(val,
       ↪ h_params_.grid_min, h_params_.cell_size), h_params_.grid_size));
    }
  };// It is faster to build a host array then copy to GPU
  boundary_.pos = temp_pos;
  boundary_.grid_reference = grid_ref;
  size = boundary_.pos.size();
    // All boundary related arrays are resized here.
  sort_key_value_boundary();
  compute_histogram_boundary();

  const auto num_cells{ h_params_.num_cells + 1 };
  boundary_.cell_begin_end.resize(num_cells);
  raw_boundary_t raw_b = to_raw_boundary(boundary_);
  gpuErrchk(cudaMemcpy(d_ptr_boundary_, &raw_b, sizeof(raw_boundary_t),
   ↪ cudaMemcpyHostToDevice));
  fill_cell_begin_end_boundary();
}
```

geometry). Graphics engines and frameworks could have introduced too much overhead, although some may offer excellent accessibility to optimisation, we chose to implement the rendering system using bare bones graphics APIs. OpenGL is a mature and ubiquitous graphics API standard maintained by Khronos and implemented by all major graphics hardware providers. OpenGL has been subject to an extensive period of modernisation, moving from the antiquated fixed pipeline into a modern approach with a dynamic pipeline and direct access to shaders and therefore, the GPU itself. However, Khronos has begun the long process of super-seeding OpenGL with their successor, VULKAN. When we began the prototype, VULKAN had been announced with promises of improved performance and an API that performed with less hardware abstraction than its predecessor. However, during the design phase for the blood simulation application, VULKAN development had not yet reached a stable version 1.0 candidate [181], therefore we decided that its implementation could prove to be too 'bleeding edge' for no proven performance gain .i.e development may have been hindered by the lack of documentation, breaking changes from updates, bugs in the implementation, or the lack of community support. Secondly, the simulation could be refactored to use multiple rendering back-ends after a working prototype was produced. Microsoft's DirectX on the other hand, is only compatible with PC's running windows operating system so its application would limit the end product to only one commercially available operating system.

The visualisation is crucial to the simulations effectiveness, the results must be presented to the user with enough clarity for them to understand the complexities of any haemodynamic consequences. As the particles will reside within the geometry, they will be obscured by the faces of the triangle mesh. Initially we chose to render the geometry as a wire-frame, and although the particles became clearly visible, the detail of the geometry was lost and it became difficult to deduce where particles were approaching some areas of the boundary. A user-defined level of transparency proved

to be more effective, but the user lost all the important details of the mesh when requiring maximum clarity, eventually becoming completely invisible. It was clear that we needed to render the regions that were 'behind' the particles and clear all triangles faces occluding the particle representation. Therefore, we implemented the capability for our user to enable front face culling of the vessel. Face culling is a technique used in graphics rendering to improve drawing performance by reducing the quantity of triangle faces the GPU renders. Typically, the back face of a scene is discarded in the face culling process. This back-face culling determines which elements of the next frame will be occluded and discards them completely. We reversed this process so that that even when the mesh is moved or rotated the 'closer' faces are discarded, see Figure 3.15, and particles are clearly visible to the user.

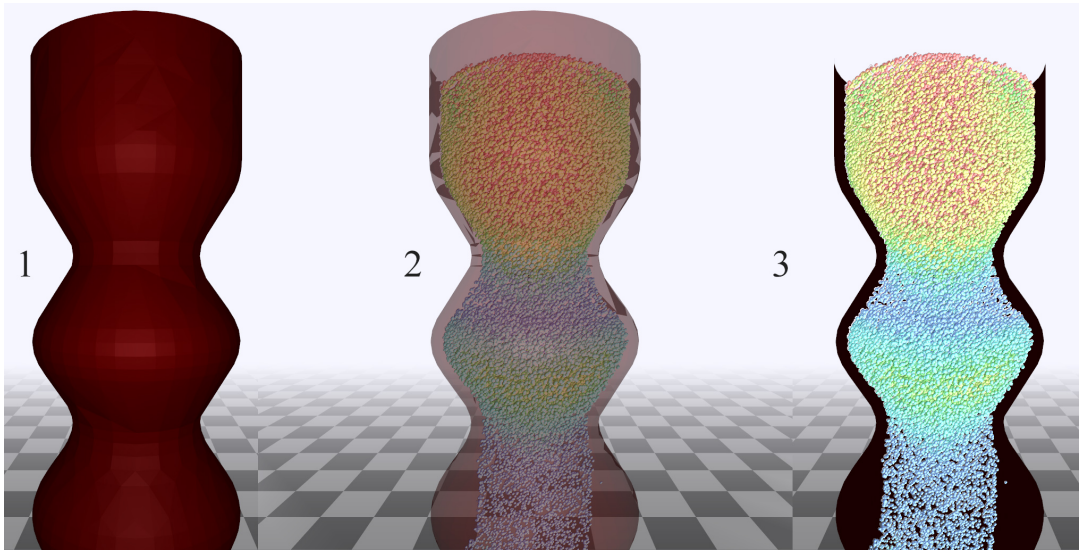


Figure 3.15 Comparison between rendering techniques used to reduce particle occlusion by the vessel geometry. 1. No special rendering technique, particles are fully occluded. 2. Transparent vessel effect created when the user configures the alpha channel to 0.5 (50%). 3. Only the 'back' faces of the vessel are drawn after the user enables front face culling.

Particle rendering was especially important within our simulation as we chose to avoid surface generation techniques for SPH. We required clear and distinct particles under any given rotation or translation. Particle rendering initially functioned by representing particles as custom instanced billboards that discarded any pixel outside

of the conventional circular shape. In computer graphics, billboards function as a flat texture that is always facing the virtual camera. We used this billboarding method until we implemented the functionality to rotate the vessel after user input, then the transformations created strange behaviours. Particles would scale hugely or become almost flat when rotated against a certain axis, behaviours that were missing when rotating the scene itself. Rather than trying to re-engineer the billboards to rotate against the rotation of the vessel, we opted to leverage the OpenGL point sprite API, which functions as a built-in billboard that will face the virtual camera regardless of any transformation. The sprite size of a point is usually defined by pixels so that the point is always the same relative size, this did mean that the particles would 'grow' as the user zoomed out. We facilitated this feature by passing the screen size and resolution to the vertex shader so it could make the point a constant size, regardless of the user position in the scene.

Within our visualisation we use the colour of the particle to reference physical quantities and properties of the simulation e.g. particle density, velocity and grid cell reference. We tend to refer to physical properties with a colour scale between red and blue, red being the highest and blue as the lowest, see Figure 3.16. During the simulation this colour based rendering allows the user to quickly determine the physical conditions and some behaviours of the flow without having to resort to printing values to console or external files, which can slow execution. Initially all colours were determined by querying the particle collection to learn the range of values for each property. Using the minimum and maximum values for a property we could scale the colour weighting relatively; we would query for these values after a set number of frames passed (100 frames in our CPU tests). The iterations with this additional computation did not suffer largely in performance as these properties could be read within the normal advection methods. However, the process was deemed somewhat ineffective as the colours could flicker quickly if extraneous values were

read. Instead we define domain specific values that coincide with our colours, this can require some user input, but it does allow us a better understanding of the numbers present while the simulation is running.

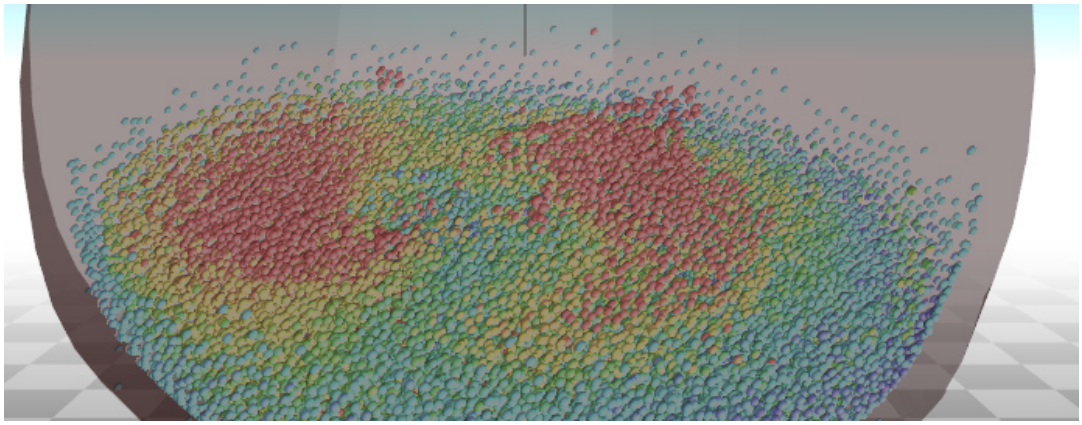


Figure 3.16 Bowl shaped geometry holding particles coloured by physical quantities and properties of the simulation. Red particles have the highest velocity, blue particles with the lowest velocity, and colours mixed between the two for the intermediate velocities.

The grid reference colour scheme seen in Figure 3.7, had to operate on a variable number of grid cells and still produce a valid output and visually discernible colours. The effect was created by using the 1D discretised grid reference from Section 3.4.1 and computing a blue, green and red colour component by unpacking the grid reference with the modulus operator and variable divisions, see Source Code 3.4.

Source Code 3.4 RGB Colour value from a particle grid reference.

```
1 const int blue = grid_reference % 256;
2 const int green = grid_reference / 256 % 256;
3 const int red = grid_reference / 256 / 256 % 256;
```

3.7 Performance

The first prototype of the simulation had the capability to render the geometry and perform the SPH calculations with the force based boundary, see Figure 3.17. The simulation could render the pre-selected vessel geometry and its centreline, with user selected variable transparency, labelled as 'Vessel Alpha'. Some dynamic properties could be changed by the user during execution with a simple GUI interface, these including domain specific boundary force properties, density, the viscosity constant and stiffness. Allowing the user to change these particles helped us to see how the geometry could affect the results of the simulation and how we may need to tweak the parameters of the simulation to produce accurate results. The working prototype was developed to run entirely on the CPU. A basic form of parallelisation was implemented using OpenMP [131], rather than any GPU utilisation. As expected, the initial implementation outperformed earlier interactive SPH examples despite the lack of regimented optimisation. This was largely due to the improvements in the hardware over the past decade.

3.7.1 Utilising the Graphics Processing Unit

For our SPH simulation we chose to implement our GPU acceleration using NVIDIA's CUDA, as we had existing NVIDIA based hardware resources and no requirements for any particular platform. We chose to accept the limitations that come with proprietary systems, moreover if the simulation was integrated into a market-ready product, having NVIDIA support could even be beneficial. In the early stages of our prototyping we did experiment with GLSL shaders, and the feature set of transform-feedback. We found that although the performance was competitive, adding the additional developmental complexity of low-level coding could incur excessive time costs during implementation. CUDA has a rich feature set including primitives and C++ like syntax,

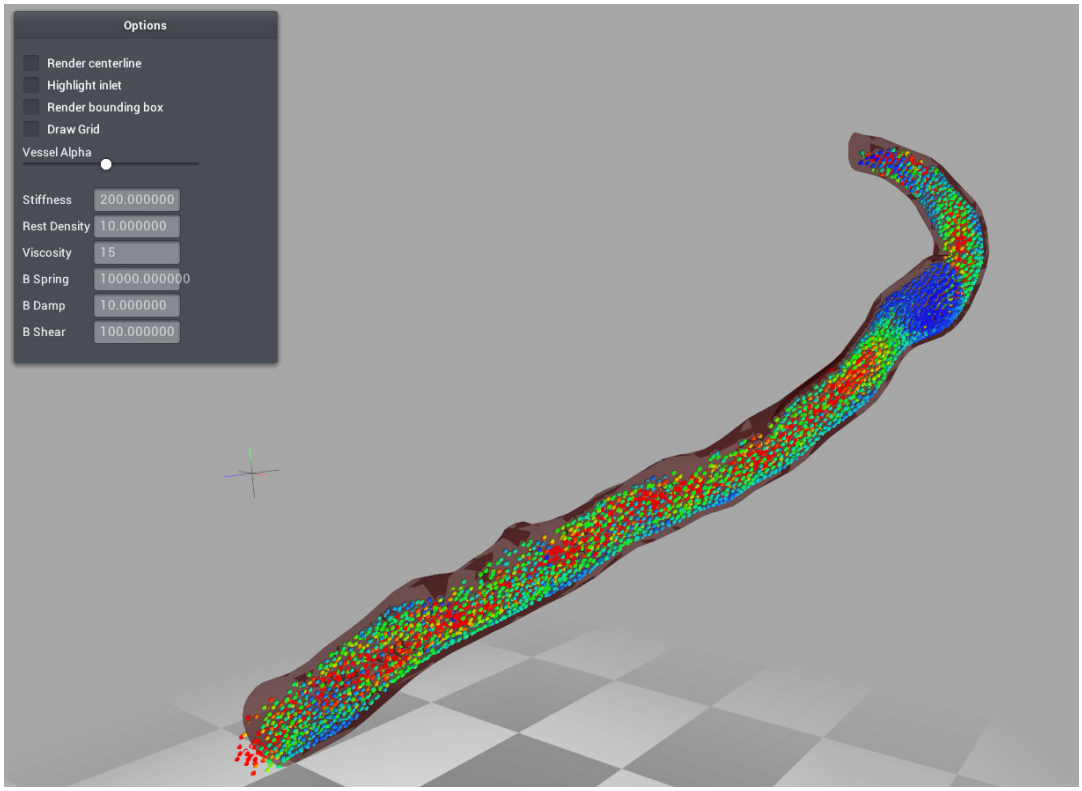


Figure 3.17 Initial CPU based SPH blood flow prototype operating within real patient geometry. Particles coloured by velocity, particles range from blue, having lower velocity, and red for the particles with the highest velocity.

a great deal of community and professional support, and a large suite of community driven and production tested libraries, e.g. Thrust [65] and CUB [110]. Additionally, debugging graphics applications is already fraught with difficultly without neglecting the benefits that come when using NVIDIA's Nsight [127] debugging suite.

A CUDA application's source code consists of two disparate sections, traditional host code (CPU) and CUDA device code. When compiled, the NVIDIA CUDA Compiler (NVCC) embeds the GPU functions in binary images and forwards the host code to the C++ compiler. These GPU device functions are conventionally referred to as kernels, not to be confused with the previously mentioned SPH smoothing kernels, see Section 3.4.3. As indicated by the compilation process, the host and device memory models exist in completely different spaces and hardware. Therefore, any data that is used within both models has to be explicitly transferred, before a

kernel launch as parameters, or after execution as the resultant data. Transferring GPU derived data into CPU accessible data has been a common problem for GPU implementations [62]. Data transfer time can be as significant as kernel runtime [60]. This poses a difficult challenge for SPH implementations. After the particle data is updated in GPU memory, if the application requires the data to be within reach of CPU code, the data must be transferred. To avoid moving large particle data sets between GPU and CPU during rendering, we take full advantage of OpenGL Interoperability. In the past some implementations would be required to transfer the particle data to the CPU in order to format it for OpenGL before transferring it back to the GPU. Instead we initialise a shared object in GPU memory, i.e. an OpenGL frame-buffer, map it into CUDA's memory space and format the data within a CUDA kernel, avoiding any costly host-device transfers, see Source Code 3.5.

Program optimisation is an iterative process, each change that is made to improve performance must be compared in some measure to previous builds. We largely based our optimisation decisions on frame timing, as the frame rate increased, we could take more efforts to make the SPH model accurate. One crucial element in common optimisation strategies is cache optimisation, great performance gains can be made when adhering to locality of memory. For optimum performance the memory access needs to be coherent, especially when handling multiple threads on a discrete device/GPU. Memory access can be implemented in various patterns to preserve coalescence and perform multiple of reads or writes of data collections with one operation. Moreover, the greatest benefit is achieved when memory is controlled to access the L1 cache, or texture caches in CUDA [128]. Kernel efficiency and overall performance can be impacted by the programs logical flow control i.e. decision based conditional code execution, e.g. if statements. Flow control decisions can result in uniform or divergent branching. Branch efficiency is defined by the ratio of executed uniform flow control decisions over all executed conditionals [127]. When

the flow control decisions produce a lot of divergent branches, performance becomes sub-optimal. If the control flow forces the threads of a warp, groups of CUDA threads operating in lockstep, to execute different execution paths, the instruction is considered to be divergent. Consequently, the serialised execution paths execute an increased number of instructions for the warp. When these instructions increase, especially the expensive global read operations, the cost of the code segments increases greatly and has a proportional effect on performance.

One example of when the CUDA kernels were refactored to reduce performance degradation caused by branching was bounds checking. When accessing primitive arrays and calculating indices to access programmatically there is a chance to attempt to access memory that is out of bounds i.e. outside of the array declaration and potentially uninitialised memory. In an early iteration of the CUDA programme, bounds checking was done numerous times within each neighbourhood search, see Source Code 3.6. This checking mitigated the chance to access invalid data caused by particles exiting the computational domain, creating a grid reference that was beyond the initialised grid memory. If the bounds were found to be invalid, the particle would be removed from computation, queued for reseeding, and handled early in the next frame. Instead, during the CUDA kernel developed to map particles into the grid, any particles that were outside the domain were immediately handled removing any chance of invalid grid references. Removing the branch had an immediate and noticeable improvement of performance, see Table 3.1 As the bounds checking was present in almost all computational kernels, the performance impact was severe, and the changes prompted an FPS increase of almost 25 per cent.

Source Code 3.5 CUDA Graphics Interoperability.

```
void push_to_buffer(){
    float *vbo_ptr;
    size_t num_bytes;
    gpuErrchk(cudaGraphicsMapResources(1, &cuda_vbo_resource_,
     ↪ nullptr));
    gpuErrchk(cudaGraphicsResourceGetMappedPointer((void**)
     ↪ &vbo_ptr, &num_bytes, cuda_vbo_resource_));

    static occupancy_helper_sizes occupancy;
    cuda_occupany_helper(occupancy, advect_particles_k);
    const int grid_size = (particles_.grid_reference.size() +
     ↪ occupancy.block - 1) / occupancy.block;
    advect_particles_k<<<grid_size, occupancy.block, 0,
     ↪ d_stream1 >>>(vbo_ptr, d_ptr_particles_);
    gpuErrchk(cudaGetLastError());
    gpuErrchk(cudaGraphicsUnmapResources(1,
     ↪ &cuda_vbo_resource_, nullptr));
    gpuErrchk(cudaGetLastError());
}

__global__ // CUDA Kernel macro
void advect_particles_k(float* p, raw_particles * particles)
{
    const int index = threadIdx.x + blockIdx.x * blockDim.x;
    if (index < particles->size) {
     const auto particle_index = index * 4;
     const float3 particle_pos = particles->pos[index];
     p[particle_index] = particle_pos.x;
     p[particle_index + 1] = particle_pos.y;
     p[particle_index + 2] = particle_pos.z;
     p[particle_index + 3] = 0.f;
    }
}
```

Source Code 3.6 Grid reference to index bounds checking.

```
auto cell_iterator = boundary->cell_begin_end[adj_cell].x;
const uint cell_end = boundary->cell_begin_end[adj_end].y;
while (cell_iterator != cell_end) {
    if (cell_iterator != UNDEFINED_CELL) {
        // Potential Branch
    }
}
```

Table 3.1 Framerate comparison before (A) and after (B) the removal of bounds checking before array access. Both tests were executed on a NVIDIA 970 GTX paired with an Intel i5 3570K. Both tests operated with the same input parameters and the backward facing step geometry, see Section 4.2.5.

Software Iteration	Average FPS (**SD**)
A	36.35 (**4.50**)
B	45.41 (**6.79**)

3.7.2 Data Structure Parallelisation

When the GPU prototype was developed the neighbourhood search component had to be optimised for the differences in GPU and CPU architecture. The CPU prototype would store the list of neighbours on each time step, whereas the GPU relied on an optimised data structure and high parallelism to allow easy iteration over the neighbours without storing them. Storing the neighbourhood list incurs a large memory footprint and has the potential to reduce the accuracy of the sim. Due to the dynamic memory patterns of the GPU, creating multidimensional dynamic memory is sub-optimal. Furthermore, using a 1D array in place of the multidimensional array

would require a constant maximum number of neighbours which has the potential for a negative effect on the accuracy of the simulation.

The main data structure within the simulation is the collection of particles, read and written to many thousands of times per rendered frame, it was pivotal that this data structure would take utmost advantage of the cache structure of the GPU. The initial CPU implementation defined the particle data as an array of structures (AoS), i.e. the data was ordered into a collection of 'C' style structs, with member properties relating to the particles dynamic data see Source Code 3.7. However, the performance centric GPU implementation defined the particles as a structure of arrays (SoA), essentially the reverse of the previous implementation, the data is laid flat in a series of arrays contained within one struct, see Source Code 3.8. Accessing and maintaining AoS data requires much less administrative effort, the data is accessed using one index operator (`operator[]`). Although AoS are much easier to manage and use, they are only competitively performant when all member fields are read [50], whereas the cache locality of SoA and more sporadic access of fields found in the SPH calculations are the clear choice.

Source Code 3.7 Particle data demonstrating an array of structures (AoS).

```
struct particle{
    float3 position;
    float3 density;
    float3 velocity;
}
particle particles[100] = particle_array;
//access particle at index 5
particle my_particle = particles[5];
```

The GPU prototype yielded favourable conditions for interaction by a large margin. Table 3.2 illustrates the massive improvement in frame-rate when the simulation is

Source Code 3.8 Particle data demonstrating a structure of arrays (SoA).

```
struct{
    float3* position;
    float3* density;
    float3* velocity;
// Dynamic arrays to be allocated later
} particles;
//access the properties of the particle at index 5
float3 position = particles.position[5];
float3 density = particles.density[5];
float3 velocity = particles.velocity[5];
```

accelerated by the GPU. All tests below 50,000 particles show the FPS to deviate between 95 and 100 FPS, which is likely due to operating conditions on the computer during the test rather than the particle count. Moreover, GPU performance at lower particle quantities performs closer to expectations when the computation time is measured independently, see Figure 3.18. The GPU prototype FPS begins to suffer after 50,000 particles are initialised in the simulation. However, in the same test case the CPU prototype would be considered far below the operating rate for interactive applications and visual continuity. Furthermore, the GPU accelerated prototype was also beholden to the additional 38,409 invisible boundary particles of the patient-specific geometry. Moreover, the visualisation itself may be having a detrimental effect on performance. Figure 3.18 demonstrates the time required to compute the SPH method independently from the visualisation, the performance deterioration remains linear after 50,000 particles. When simulation operates with 100,000 particles the SPH method is performed after an average of 12.9ms, a potential maximum of 83 FPS. A stark difference from the actualised 48FPS seen in Table 3.2. This stark change indicates that there may be a limitation or performance bottleneck when rendering the

particles and interface that may be subject to improvement with further performance optimisation.

Table 3.2 Framerate comparison between prototype implementations. Both Tests were executed on a NVIDIA 970 GTX GPU paired with a Intel i5 3570K CPU and a 144Hz monitor. Both tests operated with the same input parameters and the Aorta Section A geometry, see Section 4.3.1.

Particle Count	Avg FPS CPU Prototype (**SD**)	Avg FPS GPU Prototype (**SD**)
10,000	34 (**4.1**)	96 (**8.7**)
20,000	17 (**2.4**)	101 (**7.9**)
30,000	10 (**1.2**)	100 (**5.8**)
50,000	5 (**1.1**)	81 (**3.7**)
100,000	1 (**0.3**)	48 (**1.6**)

3.8 Summary

Two blood flow prototypes were developed, one native CPU application and one GPU acceleration solution. CUDA was chosen as the GPU acceleration framework of choice, due to the rich documentation, availability of libraries and support from NVIDIA. After the benefits of the GPU acceleration were measured the GPU prototype became the primary area of development. The deliverable prototype is able to import, render and process medical image data before executing the simulation. The SPH based simulation operates at interactive rates, providing 3D immersive results in real-time. Aside from algorithmic optimisations, real-time rendering techniques were implemented to drive performance and user experience.

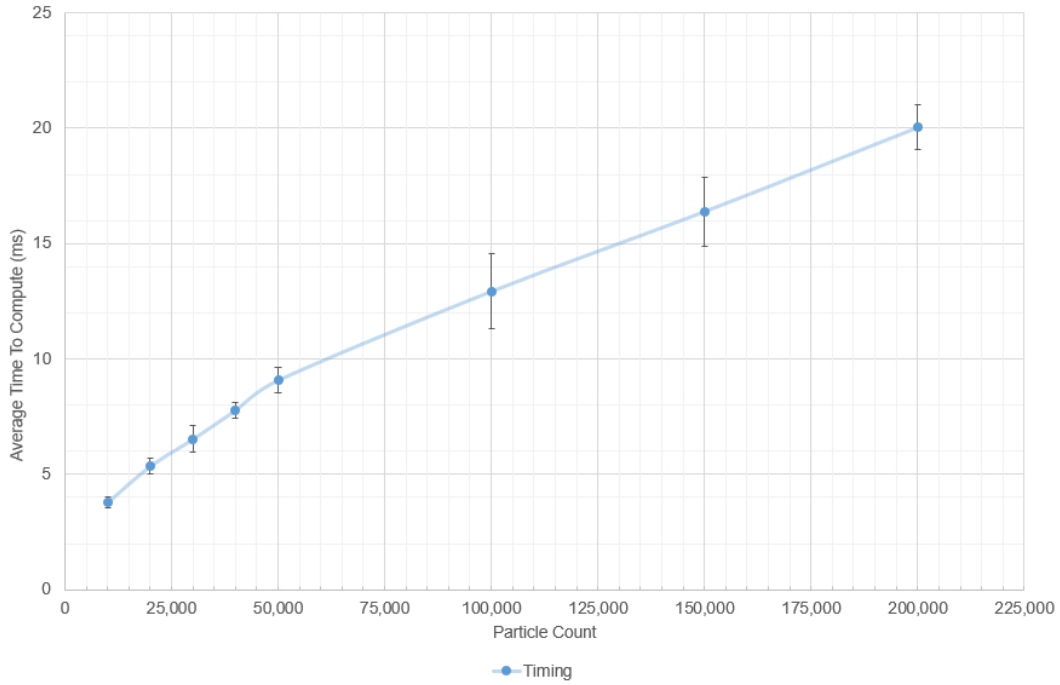


Figure 3.18 GPU SPH computation timings, measured separately from the visualisation. All CUDA kernels required to map the particles into the grid, compute the dynamic properties, and update position were measured. Tests were executed on a NVIDIA 970 GTX GPU paired with a Intel i5 3570K CPU. Measurements were taken on the Aorta Section A geometry, see Section 4.3.1.

Chapter 4

Validating Common Blood Flow Behaviours

The SPH simulation results were validated by collecting the particle data, then comparing the dynamic properties with known and expected flow behaviours. This has not been achieved in the past due to the performance complications of achieving a real-time circulation with any degree of accuracy, especially with the SPH model, there has been a dearth of workable solutions that provide an appropriate quantity of particles and an adequate environment for validation. In addition, validating computational models against real world behaviour can be a complicated endeavour, this problem is confounded by our targeted domain of blood flow, an area where empirical measurements of the *in vivo* target behaviour is difficult and sometimes invasive. Nonetheless, investigating a series of test cases signifies that the simulation can match expected behaviour in artificial benchmark scenarios. Consequently, we developed a series of artificial test cases that correspond to behaviours that replicate the circumstances found in patient geometry.

As our research is focused on the transient behaviour of the blood flow and recirculation, Poiseuille velocity distribution is an appropriate benchmark.With Poiseuille

flow, see Equation 2.2, as an analytic benchmark we defined what we expect from our artificial geometry. Consequently, we examined anatomic readings from the literature to replicate the same behaviours when creating our test cases. For example, when considering the transition of diameter from larger vessels to narrow vessels, we could look to the umbilical vein. Blood flow in the umbilical vein has a steady assumed parabolic velocity profile, while the umbilical driving pressure and the narrow entrance force a sudden and considerable increase in the velocity of the blood entering the ductus venosus [83, 84]. The test cases have been developed with particular emphasis on the cardiovascular system, but the simulation engine could be deployed onto geometry derived from any segment of the circulatory system.

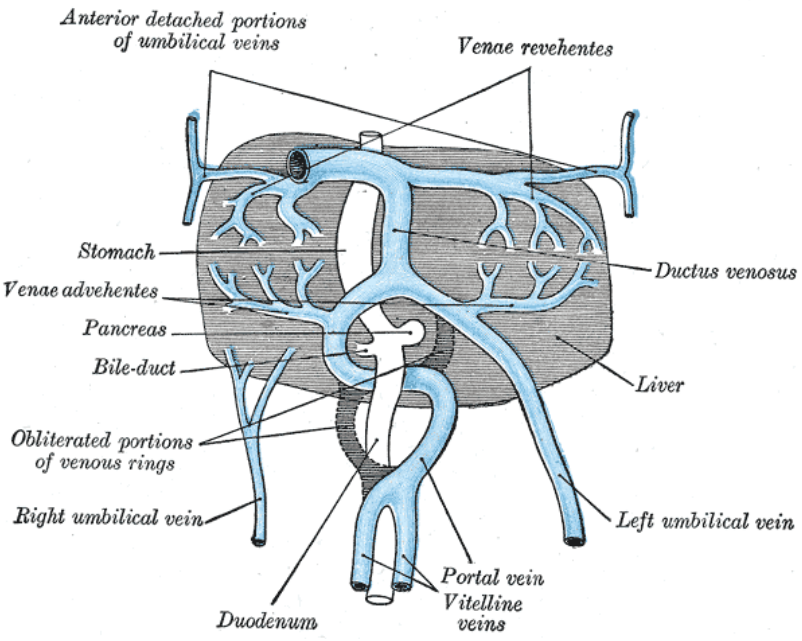


Figure 4.1 The liver and the veins in connection with it, of a human embryo, twenty-four or twenty-five days old, as seen from the ventral surface (Source: Gray [59]).

All the following test cases will have a graphical depiction, a screen capture directly from the real-time simulation, alongside a quantitative representation, and a graph of the fluid particles velocity magnitude as they pass through a designated clipping plane.

4.1 Data Collection

Particle data is not automatically collected and saved to file by the prototype application. Transferring thousands of particle properties to file takes too much time and creates a large performance bottleneck between frames. As seen in Figure 4.2, we sliced through the domain and collected only the particle information which lay on the selected plane and the origin of a user selected 3D axis. The output forms a straight line through the domain which we used to create a velocity curve, free from the influence of adjacent layers of fluid. In this form it is unlikely a significant quantity of particles will lie exactly on any given plane, consequently, a variable value $\delta$was introduced to determine the depth of the plane. This process allowed us to interactively sample the simulation in each test case to quickly examine the contentious regions. The data could also be collected over multiple frames to generate a representation of the dynamic properties over time, represented by the simulation time-step.

Secondly, to generate a more complete view of the simulation results we exported the entire simulation domain to a file that could be processed by ParaView [10], a multi-platform scientific data analysis and visualisation application well suited to large data sets. ParaView provided point analysis tools that allowed us to graph the various capabilities of the simulation without having to rebuild this same functionality into the blood flow prototype. The data could be subject to point analysis, such as a point-line interpolation, where the data is interpolated on a line section across the point data. As ParaView is largely utilising the features of VTK internally, the blood flow prototype could potentially emulate these features in the future if they proved to be useful for clinicians.

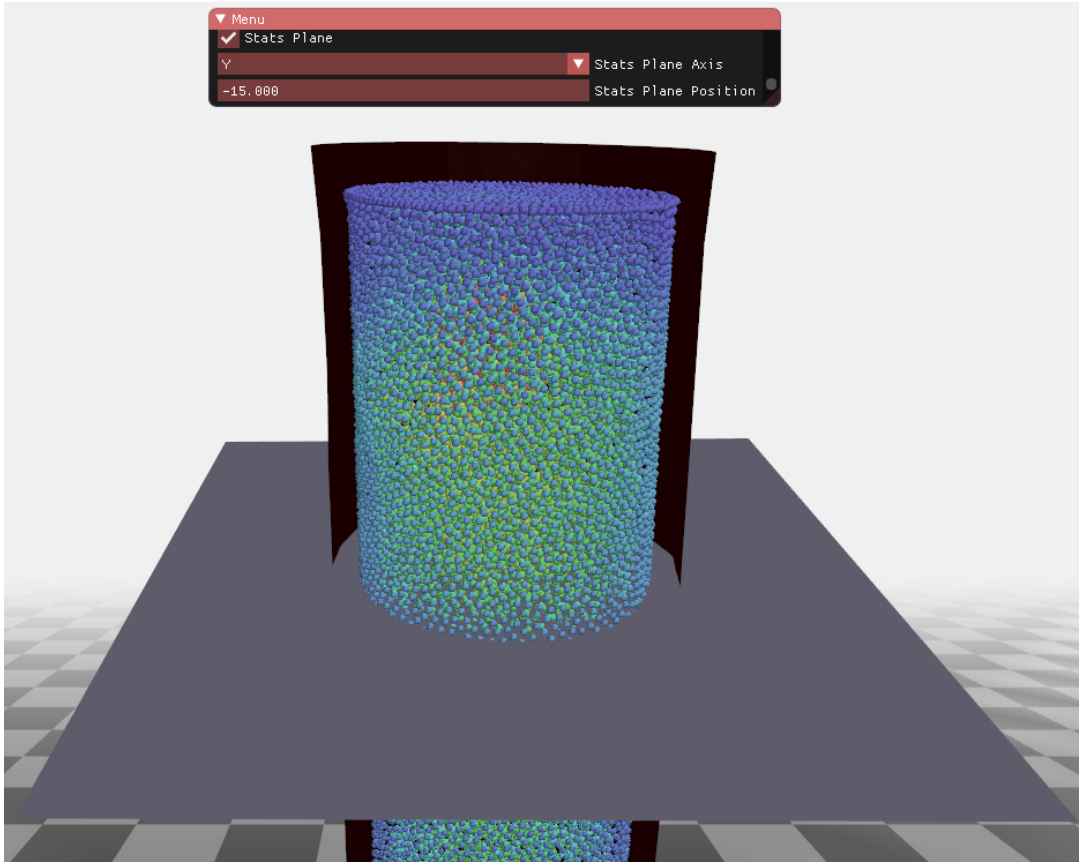


Figure 4.2 Visualisation of the first data collection technique. A plane is visualised that is the criteria for collecting particle data, any particles that lie on the plane or nearby will be collected and mapped as a velocity profile graph.

4.2 Test Cases

To develop a robust validation for clinical use we developed the following series of test cases. The test cases were designed to replicate common validation experiments and vascular structures without the additional complexities of patient geometry. By simplifying the behaviour in this way, we can firmly predict how the fluid should behave prior to the simulation experiments and compare our results with the outcomes from empirical and numerical experiments in the literature. Moreover, after validating the flow within the simplified test cases we could expand our validation

into real-patient geometry to ensure that the application could facilitate the additional complexity, and that congruent behaviours are demonstrated.

4.2.1 Straight Channel

Expected Behaviours

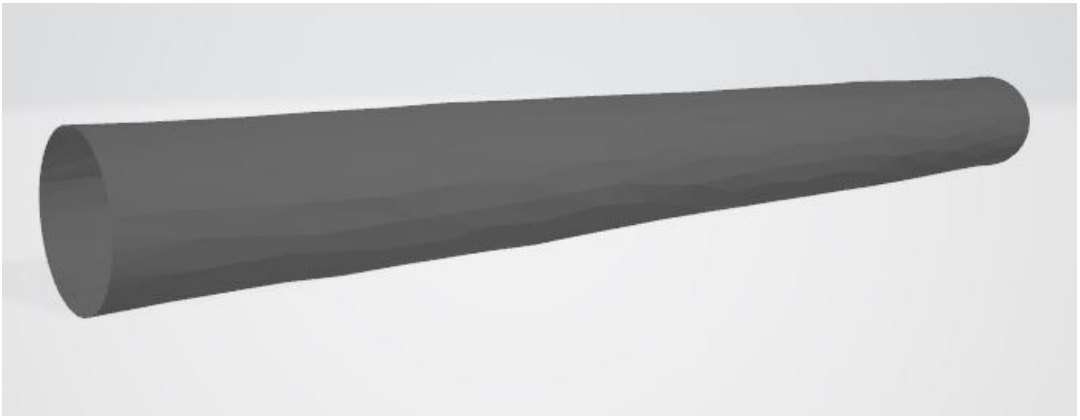


Figure 4.3 Simple triangle mesh geometry developed to test straight channel flows with a uniform diameter.

The first and simplest case presented is a straight cylindrical channel of constant diameter. This case is intended to model laminar fluid flow from inlet to outlet. The velocity profile is expected to match the unchanging parabolic profile of Poiseuille flow in Newtonian models, see Figure 2.2. This flow pattern firstly displays the greater velocity magnitude of fluid trending towards the centre of the geometry. Secondly, the fluid particles demonstrate the slip condition of the boundary and either slow down greatly to a near-stop when in contact with the boundary (slip) or come to a complete stop (no-slip), depending on the viscous forces defined in the simulation parameters.

Simulation Results

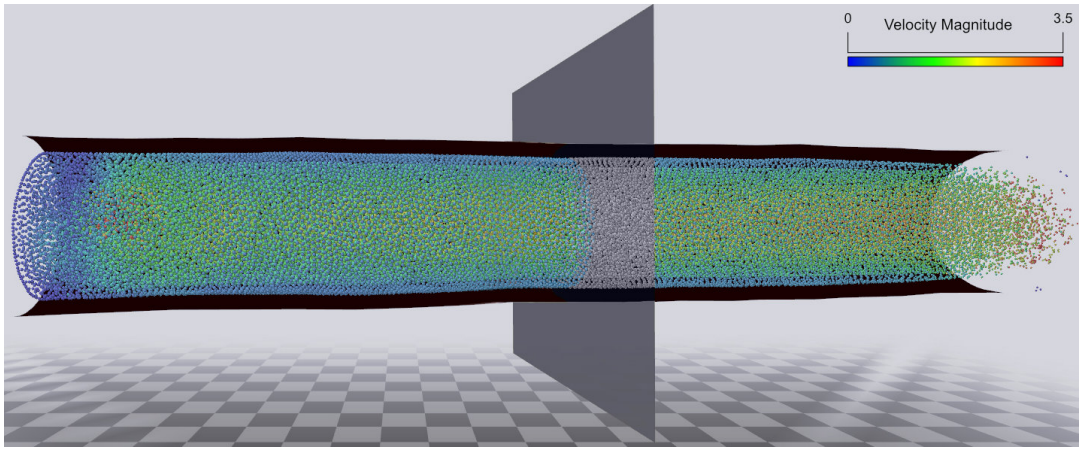


Figure 4.4 Visualisation of particles at the time of data collection, visually in agreement with the velocity profile expected. The grey plane indicates the region of discrete particle data collection. Flow modelled from left to right.

Steady flow is achieved through the vessel in the prototype simulation. Figure 4.4 shows the live real-time interactive rendering of the simulation. The simulation is visually in agreement with the expected flow behaviour, laminar flow demonstrating Poiseuille flow, greater velocity (red coloured particles) within the central channel of the tube which becomes slower towards the boundary. Figure 4.5 demonstrates how the same data is rendered within the ParaView software and provides a clearer image of the slow moving region around the boundary layer.

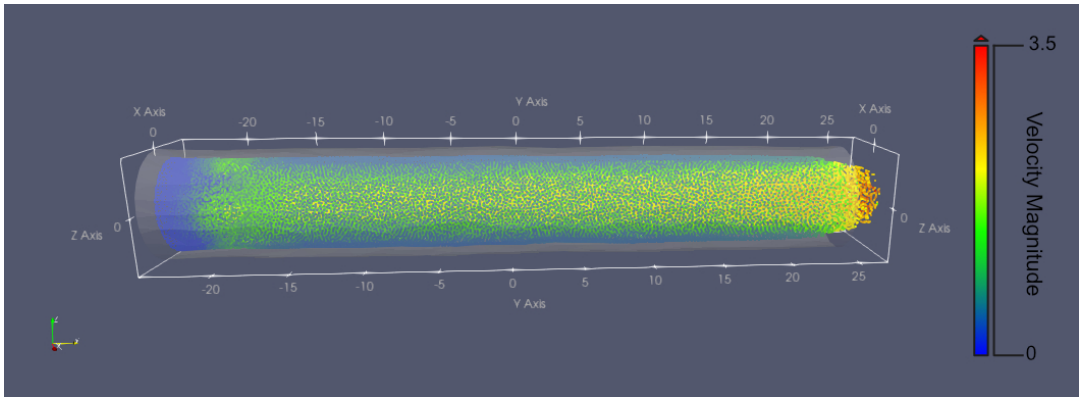


Figure 4.5 ParaView rendering of the particle flow through the geometry (transparent).

When the data is graphed directly from a plane of the simulation, see Figure 4.6, we can see the parabolic form of the velocity profile outlined by the particle positions in the vessel. As the positions of particles are collected during a limited time frame through a narrow plane, the data is discrete but still clearly demonstrates the velocity profile expected. When the point data is interpolated along a line in ParaView, we take influence from all the particles near the line. Figure 4.7 demonstrates the velocity profile found from the interpolated line. The interpolated data yields a greater indication of the general flow velocity than the discrete data. There are no discrepancies found in the interpolated data at a cost of reduced specificity. Alongside the visualisation, this demonstrates that the flow will match this pattern throughout the tube. This profile aligns closely with those taken from *in vivo* measurements in the literature [82].

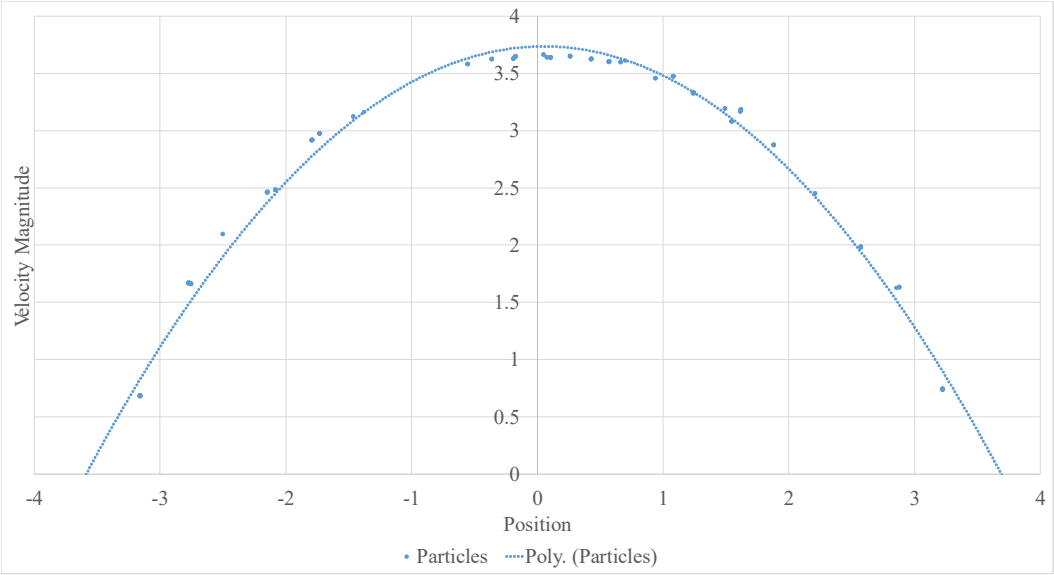


Figure 4.6 Parabolic velocity profile for fluid flow simulated in a straight channel tube.

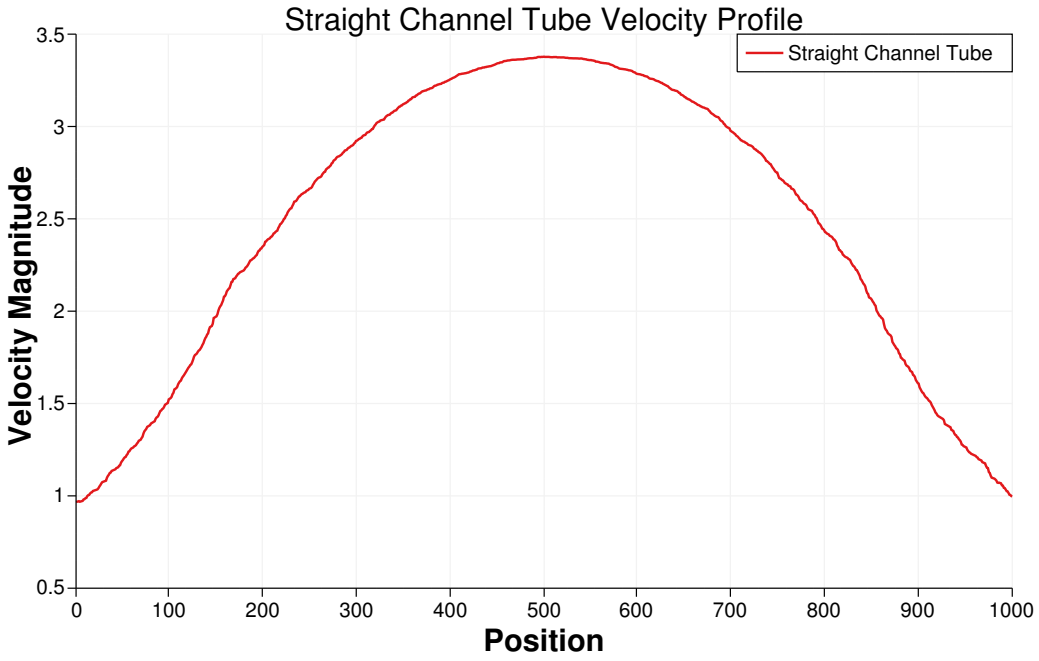


Figure 4.7 Parabolic velocity profile for fluid flow simulated in a straight channel tube. Position is the normalisation distance across the line sampling the particles.

When the discrete data is plotted against the analytical solution, we get a close match, see Figure 4.8. The analytical solution is the idealised result of the Poisuelle flow equations for the geometry used. Judging from the similarity of velocity profiles of a straight channel tube, we were able to investigate how the flow develops in more complicated scenarios.

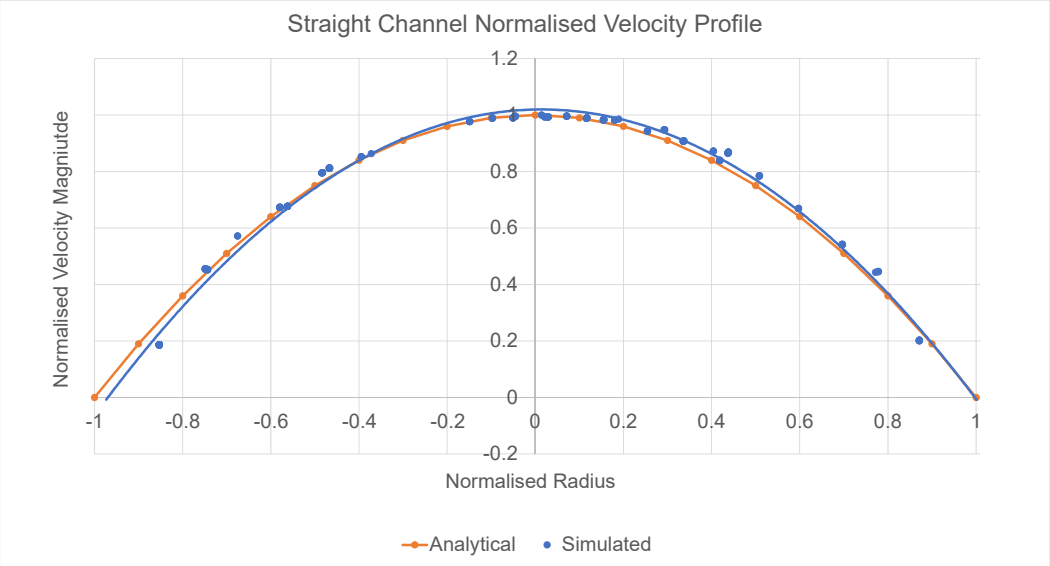


Figure 4.8 Comparison of straight channel velocity profile within a straight channel tube to an analytical solution (Poiseuille Flow).

4.2.2 Cone

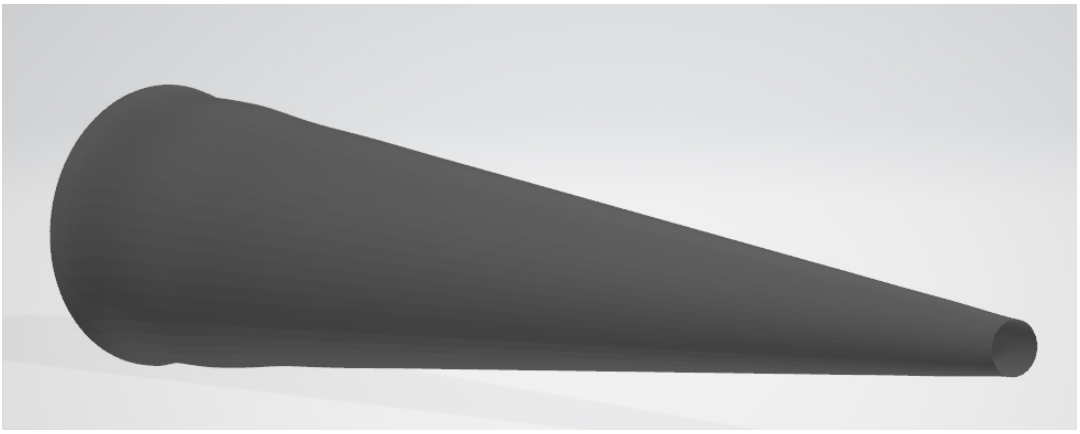


Figure 4.9 Cone geometry developed to test straight channel flows when the vessel has a variable diameter.

Expected Behaviours

The next basic case presented is a straight conical channel of constantly changing diameter, scaling from a large inlet to a narrow outlet. Although the previous case proved the simulation could replicate simple behaviour through geometry with a

uniform radius, it could not prove that the flow is dynamic and reactive to the geometry. This case is intended to model laminar fluid flow from inlet to outlet through a tapered geometry similar to that found in vessel narrowing pathology, such as the vessel narrowing found when plaque builds up during atherosclerosis. The velocity profile is expected to resemble the parabolic profile of Poiseuille flow in Newtonian models, with differing velocity magnitudes throughout the tube. When compared, the areas of the tube with a greater radius should show a less dramatic parabola with a typical laminar velocity profile, more closely resembling the straight channel flow. Whereas we expect areas of the tube with a lower radius to display a velocity profile with a higher mean and maximum velocity magnitude, as the same net flow will be forced through a narrower region by the pressure of the flow. If the driving pressure was constant the flow rate would instead decrease.

Simulation Results

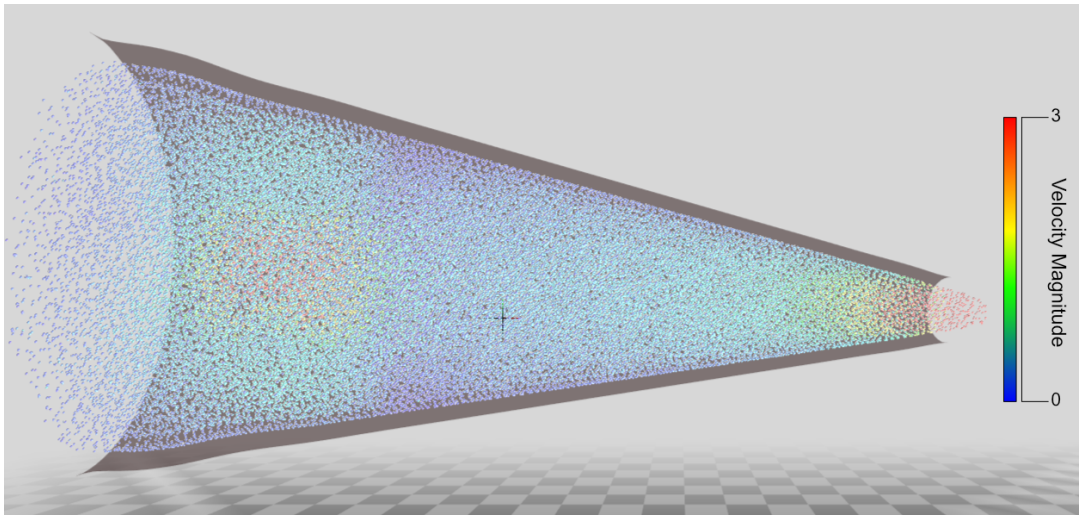


Figure 4.10 Visualisation of particles at the time of data collection, visually in agreement with the velocity profile expected. Particles flowed from the wider region to the narrow region. The seeding point for the particles can be seen as the high velocity (red) area within the wide region. Flow modelled from left to right.

As expected, the visualisation in the blood flow prototype shows the highest velocity (red particles) at the narrowest region of the vessel (the outlet), see Figure 4.10. The ParaView rendering, see Figure 4.11, clearly shows the general increase in speed with the reduction in vessel radius. The slow, deep blue, particles trend towards the inlet, whilst getting marginally faster as they approach the outlet.

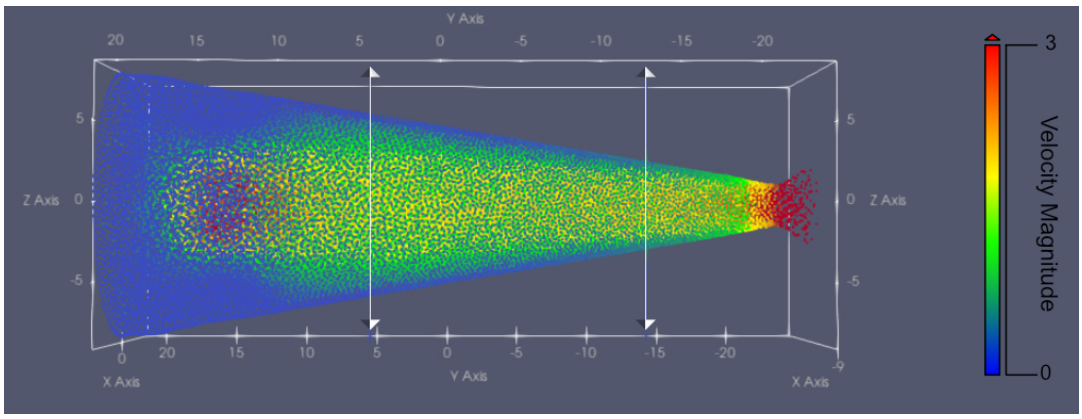


Figure 4.11 Demonstration of the line interpolation taken for the cone geometry. This ParaView render displays two lines representing the regions used to capture and interpolate the data for the narrow and wider regions.

More importantly the particles deeper within the cone were sampled in two different locations, one narrow region close to the outlet, and one within a wider region in a central part of the vessel, see Figure 4.11. Figure 4.12 demonstrates the velocity of the particles within the cone at these two different locations. The velocity profile of the wider region very closely resembles the straight channel profile when compared with the sharp profile of the narrow region. There is a distinct increase in flow velocity and the maximum velocity, even between the short distance sampled. This case is important to prove the flow is dynamic. The flow is affected by the geometry and also the forces that develop in other regions of the fluid, primarily pressure of the fluid developing as the radius closes.

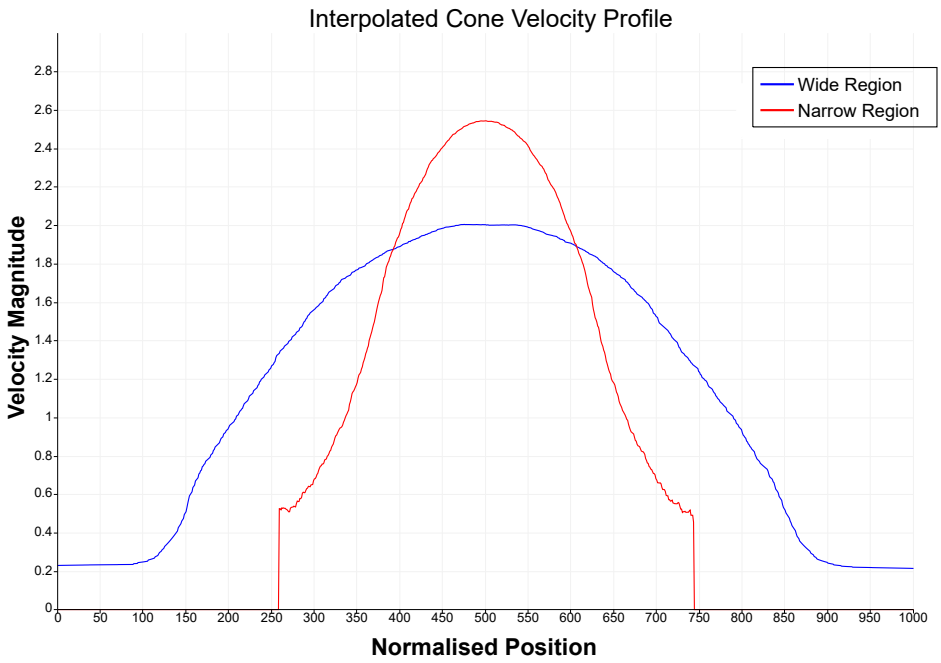


Figure 4.12 Parabolic velocity profile for fluid flow simulated in a cone shaped tube. Position axis is measured as the resolution across the line that passes through the data.

4.2.3 Hose

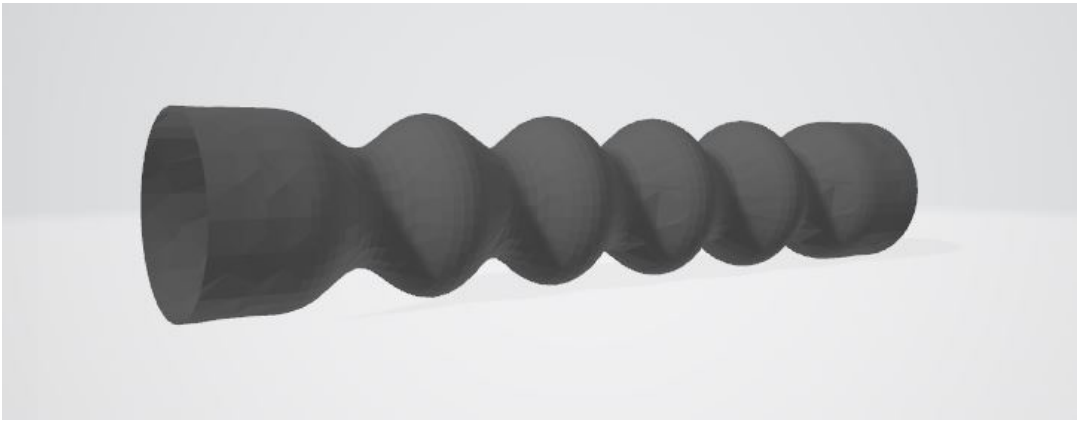


Figure 4.13 Artificial hose geometry developed to test straight channel flows with irregularities along the vessel.

Expected Behaviours

To expand upon the previous two test cases, we developed this artificial 'hose' structure, see Figure 4.13. The hose is a complex tube with an inconsistent radius, the

geometry presents with narrow and wide regions to resemble vessels with bulbous pathology, e.g. the widening observed when aneurysms balloon out the vessel walls. This test case will indicate that the flow behaviour can relax and normalise while recirculating within irregularities. As per Pousille's Law, and similarly to the previous cone test case, the narrow regions should have a flat velocity profile, trending towards higher mean velocity. There is the potential for small slow moving divergent flow patterns within the wider regions where fluid becomes trapped between the viscous forces of the boundary and the high velocity of the central fluid.

Simulation Results

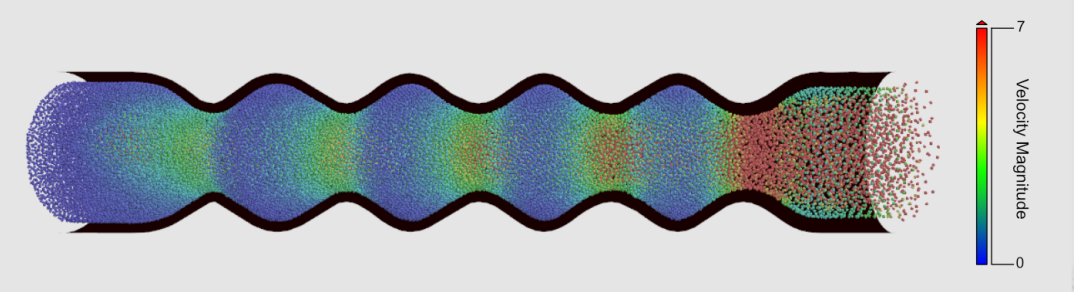


Figure 4.14 Real-time rendering with the hose geometry from the blood flow prototype. Rotated 90°. Flow pictured from left to right.

The real-time prototype provides an excellent visualisation for this case, see Figure 4.14. The colouring of the particles demonstrates the increased velocity of the flow within the narrow regions of the hose. Fluid found within bulbous extremities becomes 'trapped' between the fast, inner channel of fluid and the boundary. Despite these regions of changing geometric properties, the fluid flow remains stable and symmetrical. Furthermore, the yellow regions of fluid between the extremities of the geometry demonstrate the fluid's redistribution. In Figure 4.14 we can observe that these regions are equally sized, uniformly redistributing some of the fluid from these bulbous regions around the high-speed central channel.

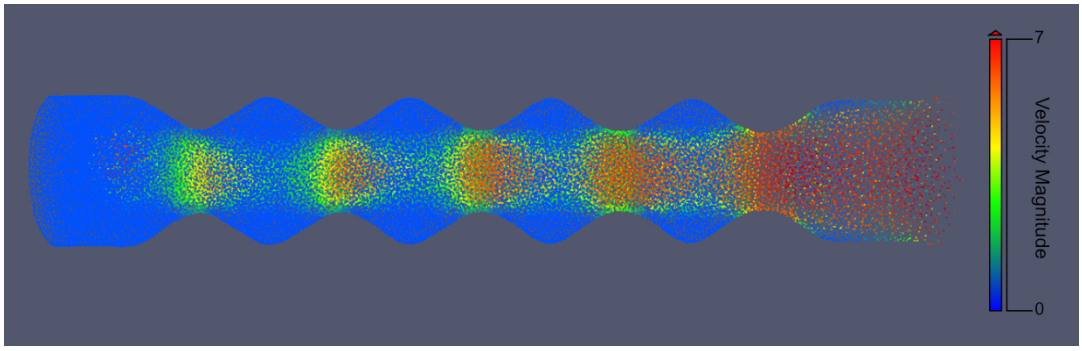


Figure 4.15 ParaView rendering of the particles within the hose geometry, including the two lines used for line interpolation in Figure 4.18.

When the data is sampled and interpolated, we can see that the flow reaches a greater maximum velocity, see Figure 4.18. Secondly, the velocity within the narrow regions is generally higher at all points within the vessel. However, the interpolation cannot detail how flat the velocity becomes as it takes influence from surrounding regions of fluid. Conversely, when comparing the discrete data, in Figure 4.17 and 4.16, the discrete sampling shows the difference when particle velocities are considered through a flat sampling plane. These planes demonstrate how different the velocity profiles are when considered in isolation, the narrow region is almost level compared to the evenly distributed parabolic profile from the wider region. Finally, we observed that the extremities of the wide region shown in Figure 4.16, contain some discrepancies in the velocity profile. These particles, which we observed to have fluctuations in velocity and direction, demonstrate some small additional circulation behaviour becoming apparent within the trapped fluid.

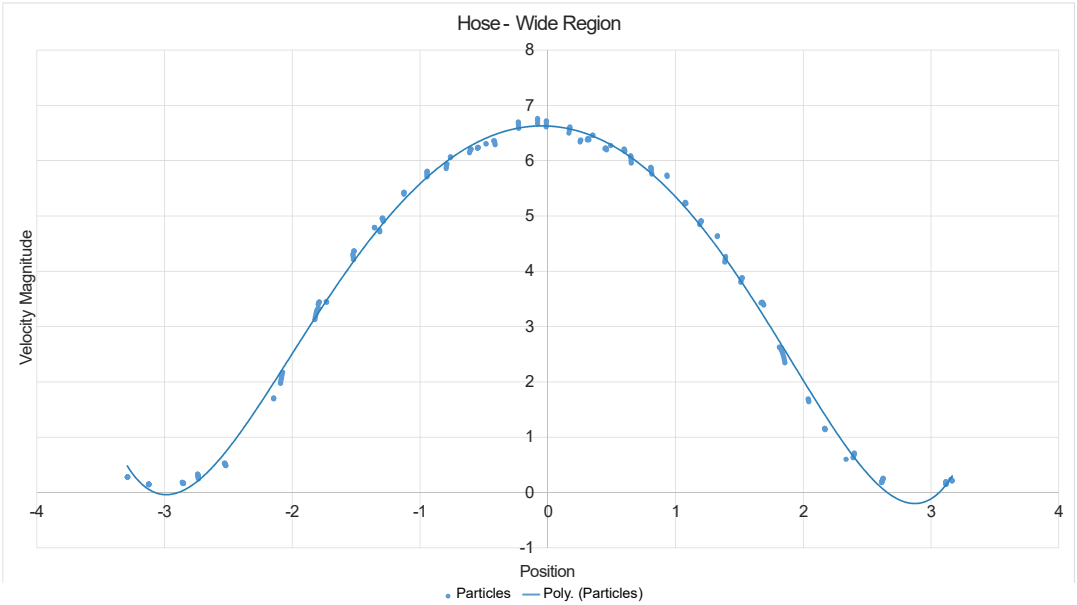


Figure 4.16 Velocity profile generated from the discrete particle data from the wider bulbous region of the hose geometry.

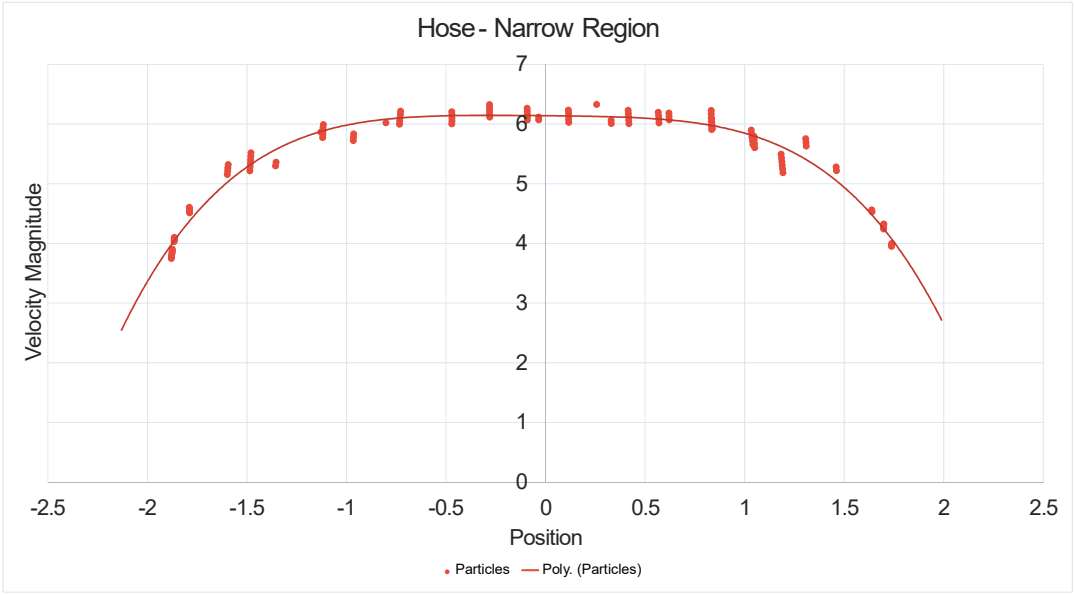


Figure 4.17 Velocity profile generated from the discrete particle data from the narrow region of the hose geometry.

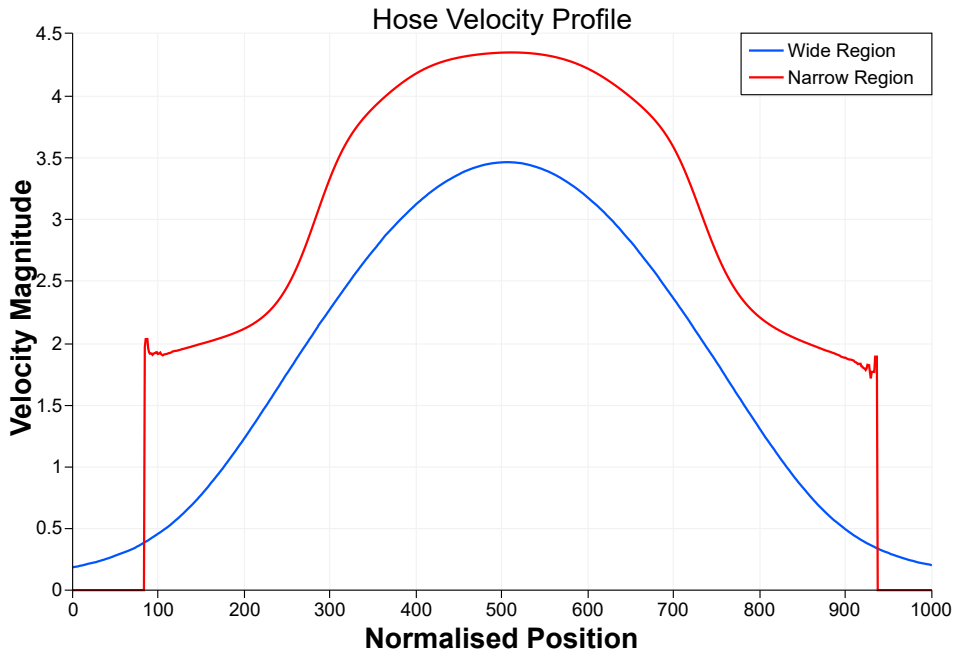


Figure 4.18 Parabolic velocity profile for fluid flow simulated in a hose shaped tube.

4.2.4 Bifurcation

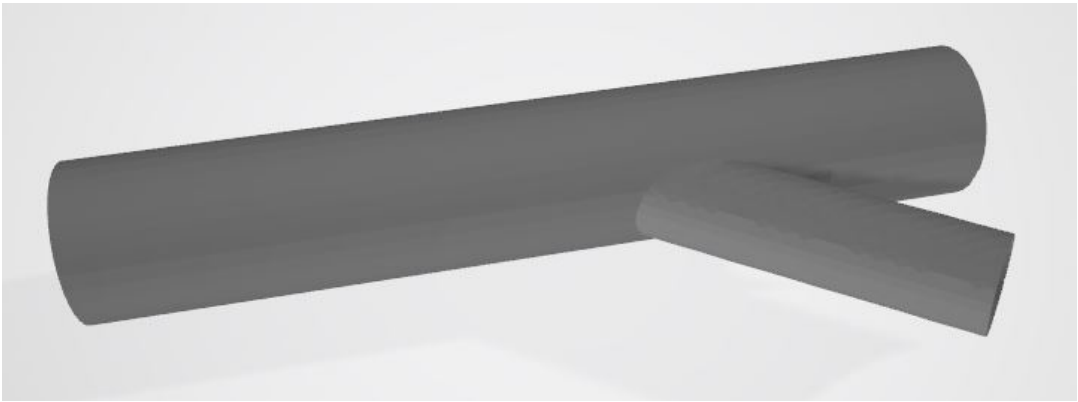


Figure 4.19 Artificial bifurcation geometry developed to test straight channel flows after the division of flow into two branches.

Expected Behaviours

Our penultimate test case was the simple bifurcation, with branching child vessels of different sizes, the child vessel extruding from the origin vessel has a lower diameter than its sibling. This structure is exemplified within human anatomy as the inferior

mesenteric artery (IMA), the third major branch of the abdominal aorta. The difference in vessel radius was developed to display that the divergent flow will have different properties whilst continuing stable flow within the child vessels. Our focus in these vessels is the fluid flow speed after the bifurcation, the flow within the narrower of the two child vessels should show a lower velocity magnitude with velocity profile. Whereas the larger vessel should remain largely unchanged. This is our first case to demonstrate that a narrow radius can lead to a slower velocity in the correct circumstances. This will be due to the topology of the artificial vessel, i.e. the child vessel will not be subject to the vast majority of the pressure forces within the parent vessel. As the resistance of the flow is inversely proportional, see Equation 2.2, to the radius we should see a noticeable decrease in the flow through the child tube.

Simulation Results

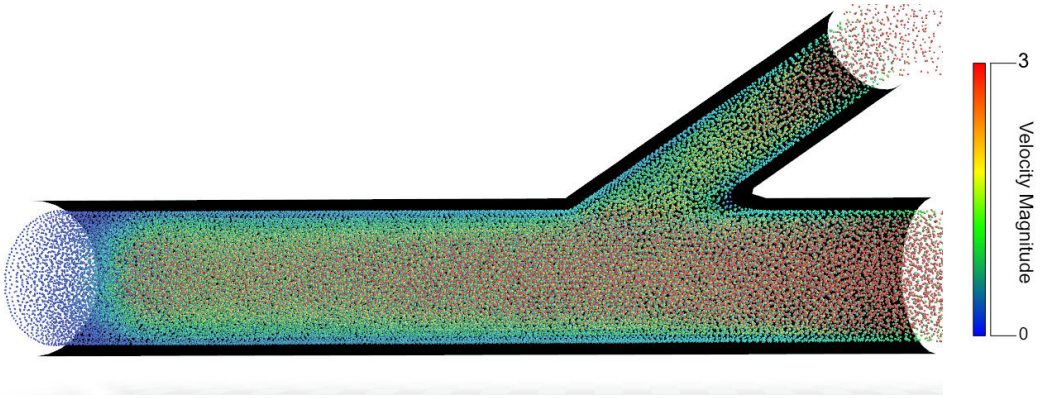


Figure 4.20 Blood flow prototype operating within the artificial bifurcation geometry. Flow Modelled from left to right.

As expected, the general flow speed within the narrow child vessel is lower than the velocity found within the main channel. There does appear to be an increase in velocity toward the outlet of the child vessel which may be due to the reduced boundary influence when nearing the open outlet, as the effect is also present within

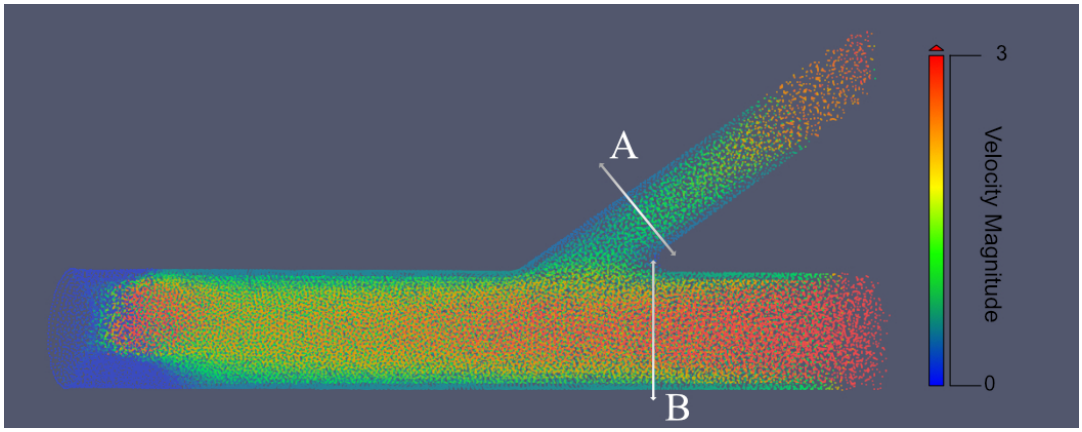


Figure 4.21 ParaView rendering of the artificial bifurcation geometry and the sampling lines used to plot the resulting velocity profiles.

the main channel, in combination with possible density errors when approaching areas with no particles whatsoever.

4.2.5 Backward Facing Step

Geometry

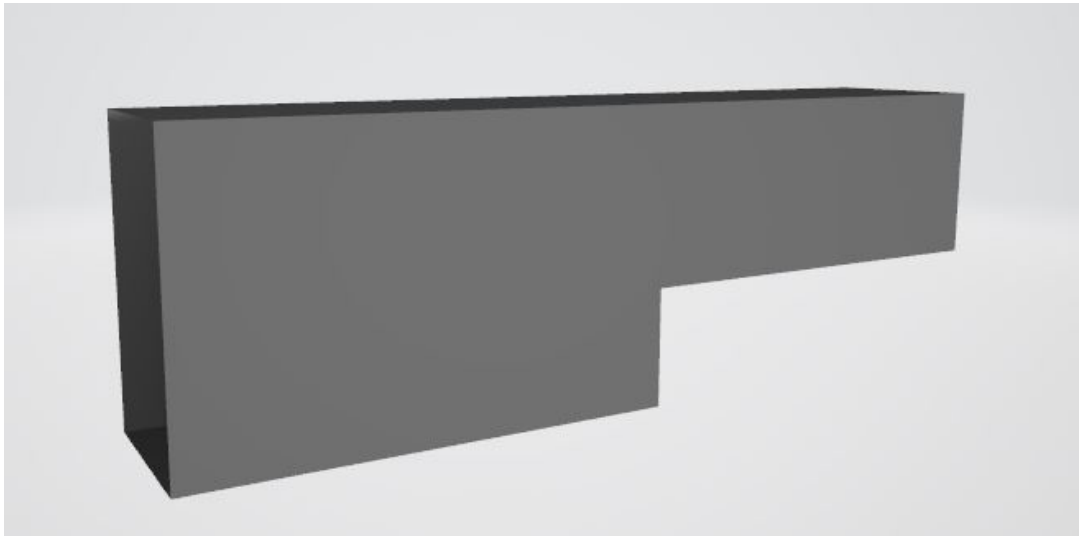


Figure 4.23 Artificial backward facing step test case geometry. The geometry features a sharp expansion to the channel, effectively doubling the diameter of the vessel.

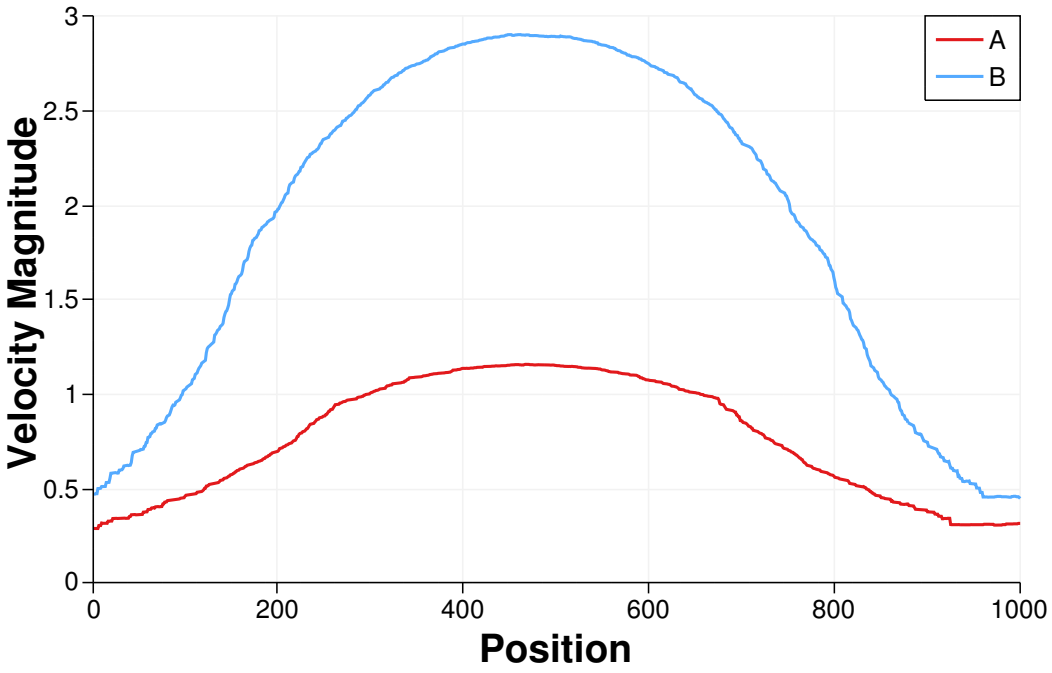


Figure 4.22 Velocity magnitude through two lines interpolated from the blood flow prototype functioning within an artificial bifurcation geometry. Line A bisects the child vessel, Line B is positioned across the main vessel, after the bifurcation.

The backward facing step a very popular benchmarking and validation test problem for CFD simulations owing to its simple geometry and the availability of quality experimental data [7, 91, 200]. The geometry features flow in a straight channel with a sudden expansion which causes a level of distinct flow separation. This flow behaviour is important for recirculation as a candidate to replicate vessels with the potential for back eddies, the swirling reverse currents found in turbulent systems, e.g. geometry found in cases with pathologies like aortic aneurysms, which presents with a dilation (expansion) of the aorta.

Simulation Results

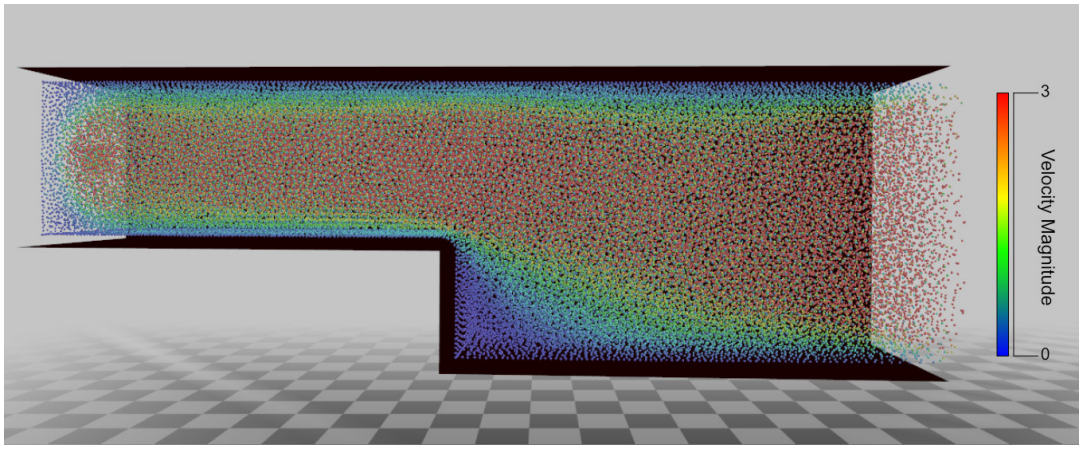


Figure 4.24 Simulated fluid flow in the artificial backward facing step geometry.

As demonstrated in previous numerical simulations, after reattachment of the upper wall eddy, the flow slowly recovers towards a fully developed Poiseuille flow [56]. From Figure 4.25 we can observe how the stream of high velocity fluid sinks initially after the vessel expansion, isolating a region of fluid within the expansion. Furthermore, the velocity profiles taken at multiple cross-sections in Figure 4.26 highlight how closely in agreement the backward facing step result is with results found in the literature [7, 200], see Figure 4.27.

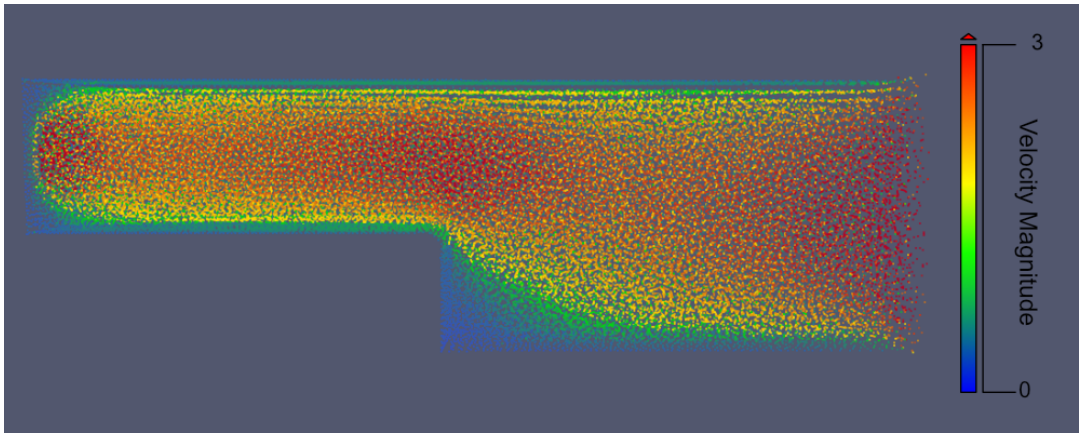


Figure 4.25 ParaView rendering of the backward facing step results. The dark red region on the left of the geometry is the seeding point for the particles.

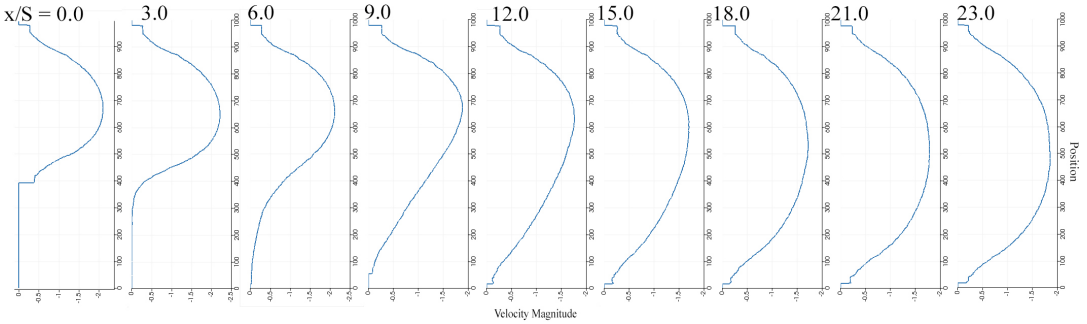


Figure 4.26 Velocity profiles within the backwards facing step geometry.

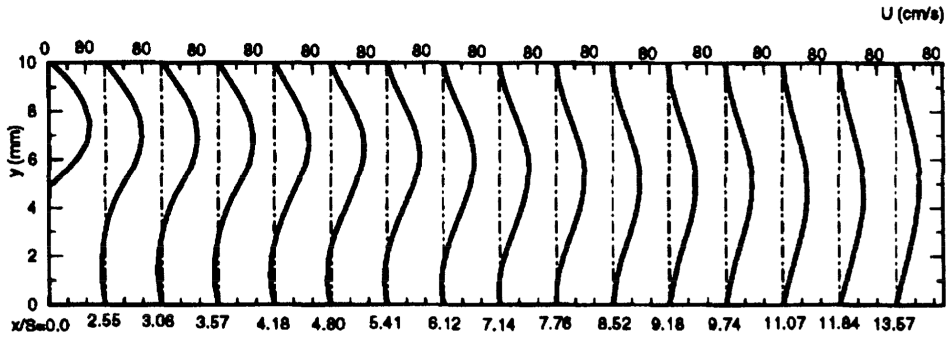


Figure 4.27 Velocity profiles for flow within the backwards facing step geometry at different cross section (x/S) positions from the literature [200].

4.3 Realistic and Real patient Geometry

Following the results of the series of test cases, we also endeavoured to prove that our blood flow prototype could operate interactively within complex, high resolution, realistic and real patient geometrical domains. The following test cases are all sourced from real patient geometry.

4.3.1 Aorta Section

Resulting from previous collaborations with the Glan Clwyd Cardiac centre we have access to two aorta sections from two different patients, which we will refer to as

section A, Figure 4.28 and B, Figure 4.30. The Aorta is the first arterial segment of the systemic blood circulation, directly connected with the heart. The aorta has two distinct regions, the thoracic and abdominal aorta.

Section A

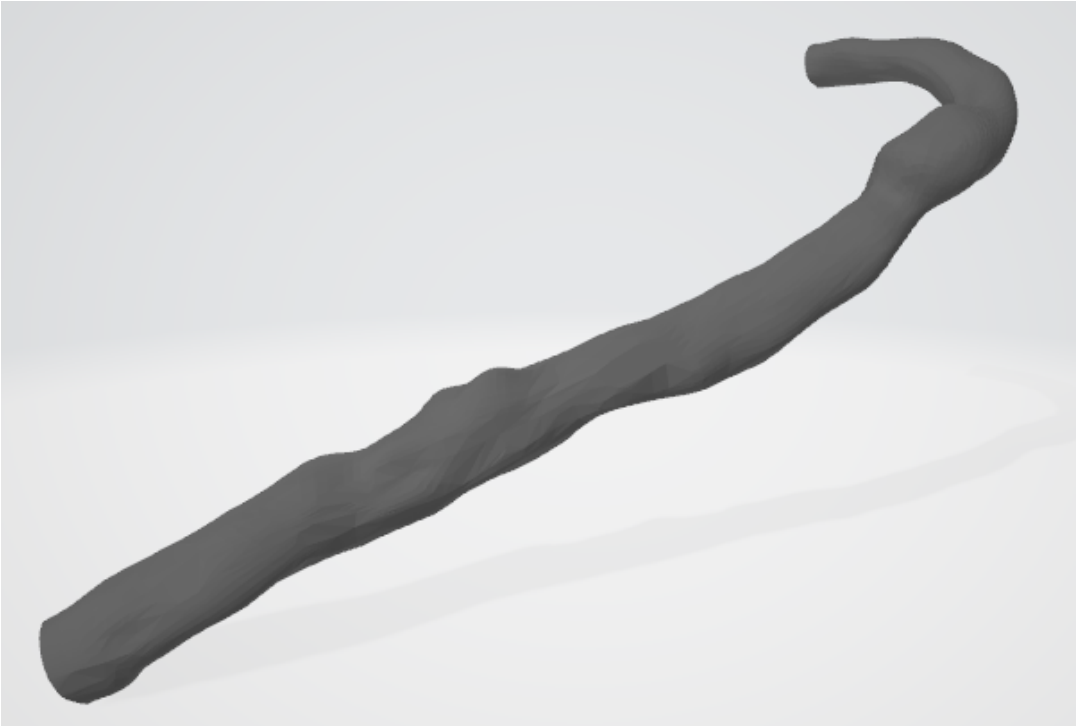


Figure 4.28 First aorta segment geometry, labelled section A. The inlet and outlet have been cropped to open the vessel.

Despite many irregularities within the geometry, the same velocity profile we observed in the straight channel geometry begins to emerge. We can observe slow moving particles around the boundary surrounding the high-speed red particles within, this is particularly apparent within the enlarged region towards the right of Figure 4.29. This case provides good evidence for the robustness of the simulation as the fluid does not only flow along one axis unlike the straight channel test case.

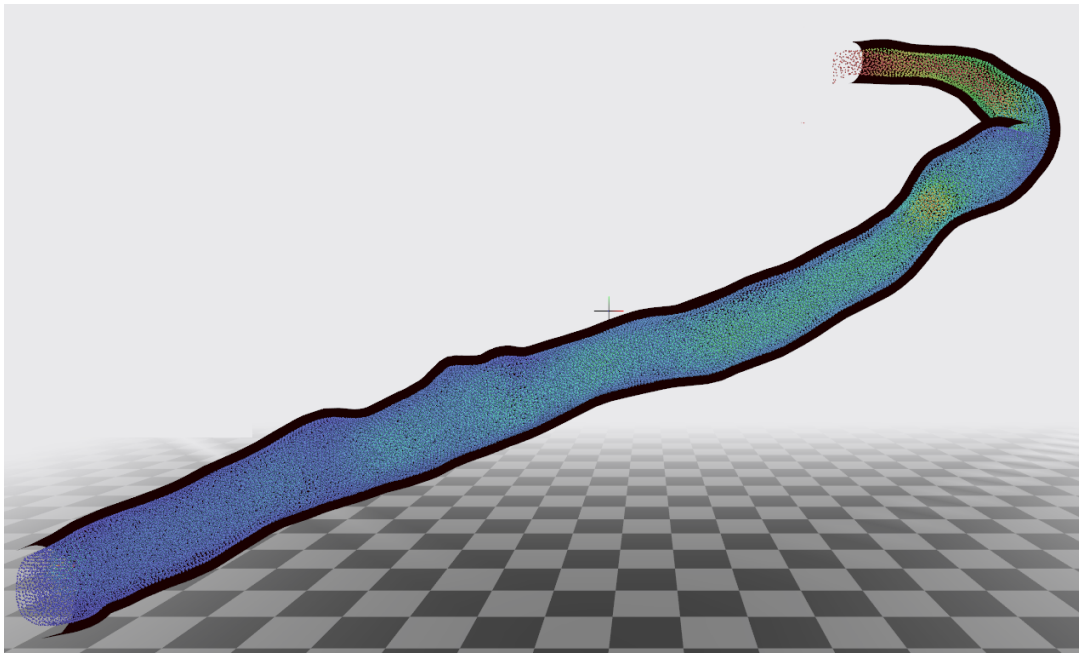


Figure 4.29 Blood flow simulation results for aorta section A. Flow modelled from left to right.

Section B

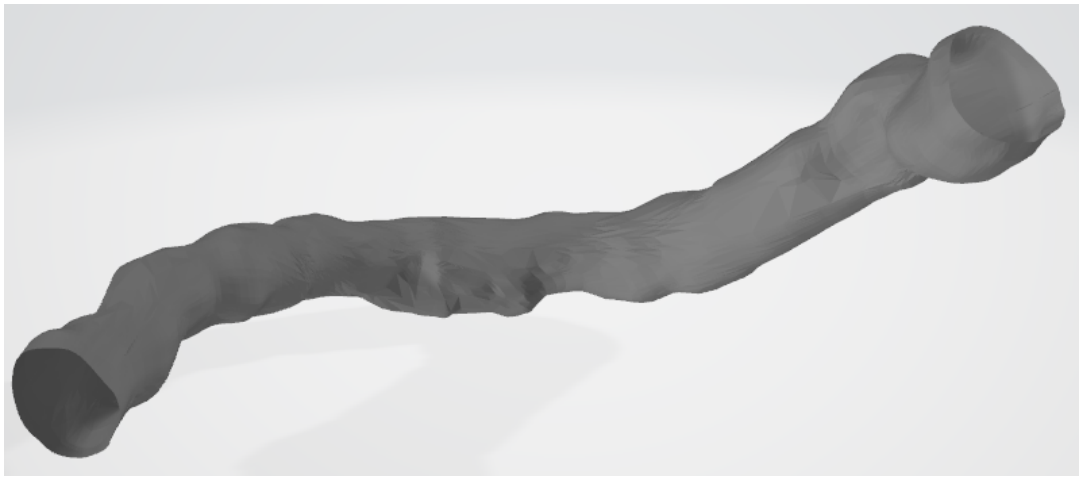


Figure 4.30 Second aorta segment geometry, labelled section B. The inlet and outlet have been cropped to open the vessel.

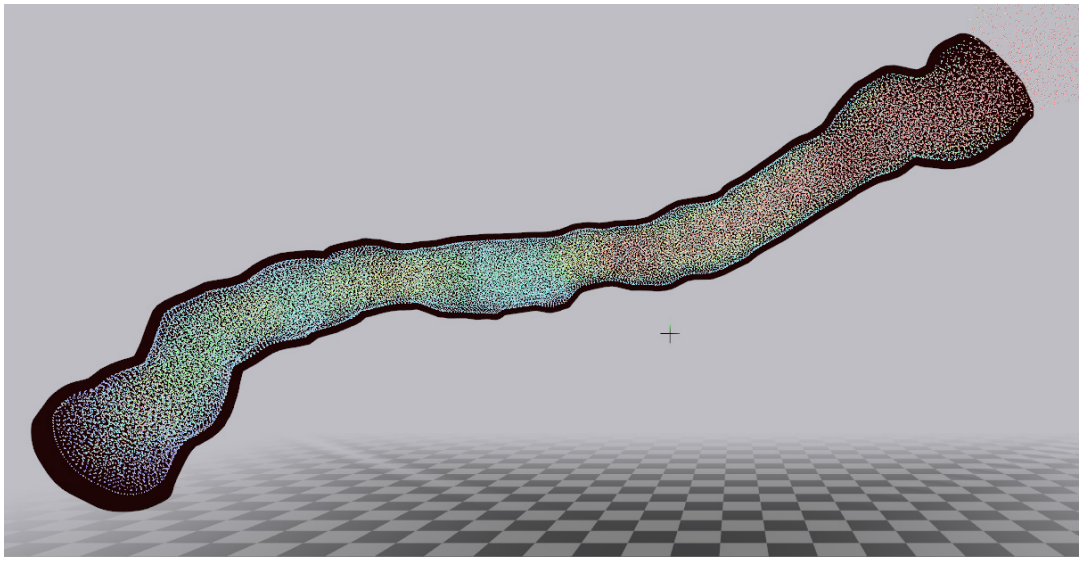


Figure 4.31 Blood flow simulation results for aorta section B. Flow modelled from left to right.

4.3.2 Bifurcation

The bifurcation is a crucial test case as is known to be susceptible to the formation of atherosclerotic lesions, and also as we are primarily concerned with the recirculation of blood flow.

Abdominal Aortic Bifurcation

Our first patient specific bifurcation is also from the Aorta. The abdominal aortic bifurcation, shown in Figure 4.32, bifurcates into the two common iliac arteries. In healthy examples, the flow from the bifurcation will split into two high velocity streams.

Figure 4.32 Aorta segment geometry including the abdominal aortic bifurcation to the left and right common iliac arteries.

From Figure 4.33, after the bifurcation we can observe the flow increase in relative velocity within the narrow child vessels. The same quantity of fluid is forced through channels with a combined diameter lower than the initial segment of the vessel.

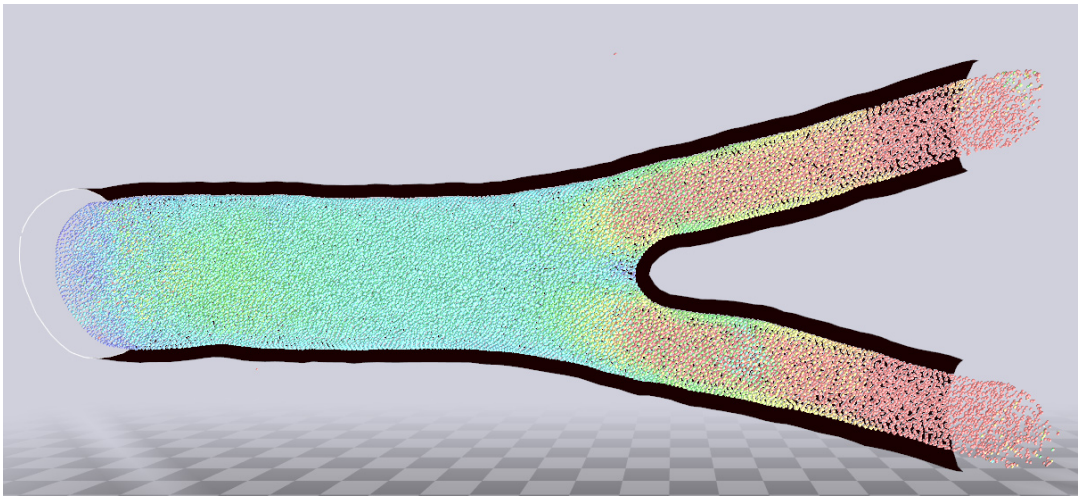


Figure 4.33 Geometry representing the abdominal aortic bifurcation. The inlet of the geometry is highlighted in white. Flow modelled from left to right.

Carotid bifurcation

Our second patient specific bifurcation is that of a carotid bifurcation, see Figure 4.34. The geometry consists of three primary areas, common carotid, internal carotid, and external carotid. The geometry provides a contrasting environment from that found in

the previous aortic bifurcation, which is relatively uniform in child vessel radius. The child vessels radius differs greatly, the external carotid is narrower than the internal carotid, and regions of bulging geometry are present. The carotid bifurcation is an important and well-represented test case for fluid simulations in the literature.

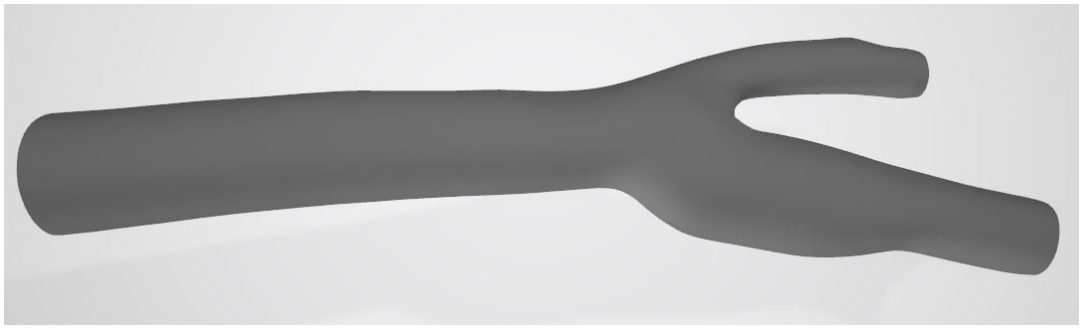


Figure 4.34 Real geometry representing the carotid bifurcation.

From Figure 4.35 we can observe the simulated flow within the carotid bifurcation. The results highlight a few key areas exhibiting interesting behaviour. Firstly, within the bulging region of the internal carotid shows a reduction in high velocity red particles, until the vessel narrows once again. This is due to having the same driving force acting on flow within a greater radius.

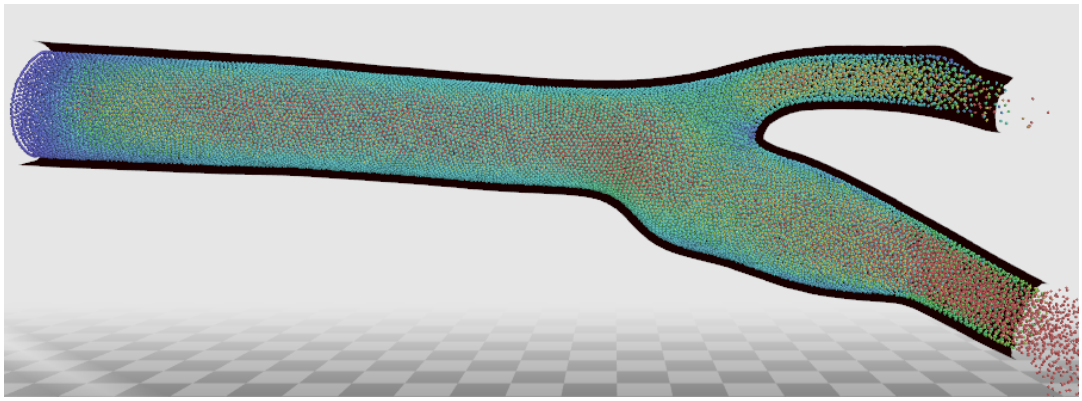


Figure 4.35 Blood flow simulation prototype operating within real geometry representing the carotid bifurcation. Flow modelled from left to right.

4.3.3 Patient Specific Vascular Geometry

The patient specific geometry previously used in [175] was obtained for use in our study. It had been produced from rotational X-ray images of a real patient. As an anatomically accurate model of a large component of the human vascular system, this tortuous vessel structure is an excellent test of the systems capabilities. As the largest and most complex piece of geometry used to validate the software its usage provides important insight into how capable the simulation is when operating within complex geometry.

Figure 4.36 Surface mesh of a larger example of patient specific vascular geometry produced from rotational X-ray geometry.

Despite the increased size and complexity, the simulation performed as intended within this complex geometry. The simulation produced some interesting behaviours, the majority of the vessel exhibits typical straight channel flow behaviour, with some areas of increased velocity due to increasing driving pressure after flowing through very narrow regions. An area of particular interest is within the three-pronged bifurcation structure in Figure 4.37. We can observe the great deviations in particle velocity, the narrowest of the regions producing generally higher velocity. As the natural continuation of the original channel, the uppermost branch receives the bulk of the fluid and exhibits a relatively higher velocity. The flow then branches again. The

middle branch shows a mix of generally slower moving particles as it only receives one third of the original flow and has relatively low pressure driving it. The driving pressure will be lower as the three child channels are of a similar size to the feeding vessel. Finally, the lower of the branches indicates an increased velocity through the narrowest region, before stabilising when the vessel expands.

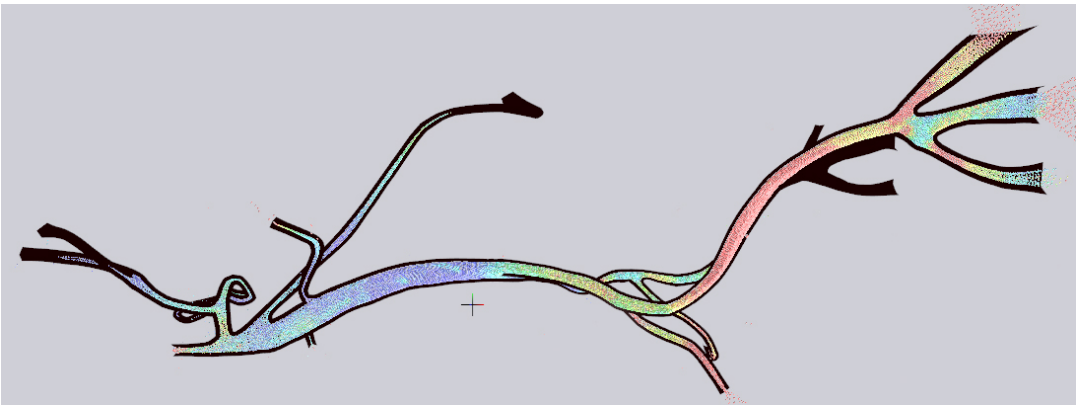


Figure 4.37 Blood simulation running through the rotational X-ray geometry. Flow modelled from left to right.

4.4 Summary

Through a series of test cases we have demonstrated the viability of the simulation to handle a broad spectrum of vascular structures and CVD pathology. The test cases were designed to illustrate the behaviours we need to simulation within human anatomy. Through comparison to measurements found in the literature we can demonstrate that the results from the blood simulation have a good basis in reality and the application has viability for clinical use. In addition, we have demonstrated that the simulation will operate effectively in both restricted vessel segments and larger complex vascular networks.

Chapter 5

Using the Simulation Prototype

This chapter describes a demonstrator application that builds on the blood flow simulation engine described in the previous chapters, to show the beginnings of a clinical decision support tool. The blood flow prototype was designed to be the engine for a decision-making tool for clinicians. Visualising and manipulating blood flow through any given medical geometry. Early consultation with clinicians formed the initial requirements of the prototype application. Alongside the geometry visualisation, clinicians required methods of interaction with the flow, specifically to create and remove blockages. Subsequent meetings yielded positive feedback and future meetings will shape the development of new interactions. A clinical trial of the software will be future work, but comments received from an interventional radiologist augurs well for its future use. Feedback was gathered following a demonstration of the prototype application, see Appendix A, and a subsequent interview, see Appendix B. The application demonstration showcased the prototype implementation of the simulation framework, operating in the geometry explored in Chapter 4 and the interactivity outlined later in this Chapter. In particular, in communications following an interactive demonstration of the prototype in February 11, 2019 with Professor Derek Gould, an interventional radiologist with 35 years of experience, he noted '...a nice simulation of

laminar flow through a vessel and an aneurysm.' In response to the initial meetings at the beginning of the project, he also noted, 'I was clear about the need to simulate the real world non-laminar impacts of the changed configuration of the aneurysm.' During an interview conducted on February 21, 2020, Professor Derek Gould and Dr Thien Howe, provided feedback that predicted the simulation to have a number of potential uses in decision support, in particular it was noted that it could be used to simulate an interventional task, aid in the application of contrast agent and benefit cases with an aortic aneurysm by determining local pressures and consequently predict the areas with a risk of a rupture, see Appendix B.

Interaction with the vessel begins with the customisation of the rendered scene, i.e. the ability to move, rotate and scale the vessel. By enabling this interaction, we immediately begin to benefit from an immersive 3D environment. New perspectives can be achieved that may have been lost when viewing volumetric data. Secondly, the vessel must effectively visualise the blood flow. Particles must not be occluded by the vessel geometry if specified by the user. Finally, the simulation needed novel interactions to generate hypothetical scenarios which could then be used to evaluate the haemodynamic consequences.

The prototype tool is primarily designed to operate on the conventional desktop computer, specifically the keyboard and mouse interface. However, the application has also been updated for use on the 3D immersive zSpace display, paving the way for future generic implementations on other immersive technologies.

5.1 User Interface and Simulation Parameters

Users load surface geometry derived from medical image data and initiate the simulation. The mesh is then processed, we clean the mesh as described in Section 3.2.1, compute the centreline and generate the boundary particles. Once processed, the

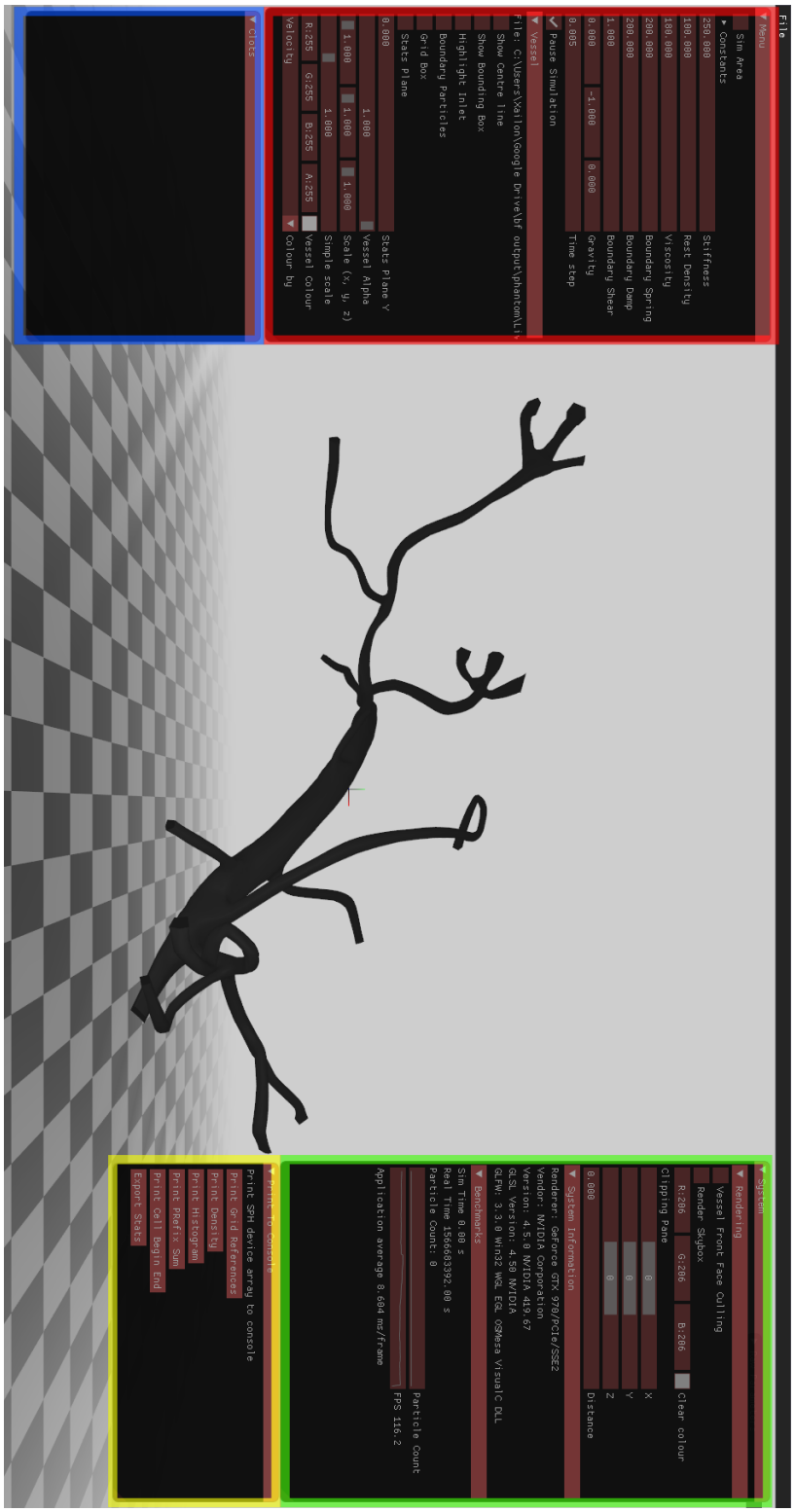


Figure 5.1 Blood Simulation Prototype example scene show casing the GUI. Real patient geometry has been loaded into the application. The interface panels are highlighted with coloured borders: red for the menu panel, blue for the clots panel, green for the system panel, and yellow for the print to console panel.

mesh can be rendered and several options become available to the user, e.g. highlight the inlet, render the centreline, the colour and transparency of the rendered vessel. Secondly, various simulation parameters can be modified, either interactively at run time using the GUI or before the application has started through the modification of a simple human-readable configuration file, see Appendix C for an example. The JavaScript Object Notation (JSON) file type was chosen as a lightweight format that is both easy for humans to read and write, and easy for machines to parse. The only disadvantage of JSON being the lack of standardised comments, text ignored when the file is parsed. This may lead to a different format being considered in future, as some of the parameters may require some elaboration and exemplar values directly in the configuration file text.

5.1.1 Running the Simulation Prototype

The prototype application has been developed primarily as a demonstrator and provides access to low level details of the simulation that might not be needed in a clinical version. Currently there are four panels that can be selected: 'Menu', 'System', 'Clots', and 'Print to Console'. Each panel can be moved, resized or collapsed down. The first and most important panel to the user is the 'Menu' panel. Crucially, this panel exposes the dynamic simulation parameters to the user. Parameters such as rest density, viscosity coefficient, time step, gravity and boundary contribution can be modified in real-time. The read-only constant variables of the simulation can be also be viewed on this panel. However, these constants must be modified before launch in the configuration file. Furthermore, the simulation can be paused from this panel to make multiple changes without causing instability.

Alongside the flow simulation parameter section of the application, the 'Menu' panel hosts some rendering options. These options enable the user to customise their view of the simulation and geometry. Normally unseen elements such as the centreline

and boundary particles can be revealed for greater insight into the simulation. Further insight and perspective can potentially be gained from scaling the vessel to examine areas of interest more closely. The vessel and background colour can be defined by the user to provide better contrast with the particles and the scene background. Additionally, the particles can be coloured by a selection of different properties from the SPH computations; location defined by the particle grid cell, particle velocity, density, or pressure. Through these colours a user can closely examine the behaviour of the fluid in regions of interest.

The 'System' panel also enables some rendering customisation. The background colour (or clear colour) of the scene can be customised to offer maximum visibility for any user. Front face culling can be configured, see Section 3.6. Secondly, the panel reveals technical details and live benchmarking metrics from the simulation. Notably the simulation frame rate and frame timings are available, a crucial detail for real-time applications. If the frame rate is unsatisfactory for interactive use, the user may have loaded excessively large geometry files, specified too many boundary layers, see Section 3.5.2, or find themselves suffering with incapable hardware.

The 'Print to Console' panel contains several logging facilities for exporting the particle data to the application console and to disk in tabular format (CSV). This panel is largely for researcher and developer use, providing data insight into the inner workings of the simulation. Finally, the 'clots' panel allows control over blockages added to the vessel, providing the user with details of how many blockages there are, how many particles they are comprised of, and a button to remove the blockage. The details of the blockages are discussed in Section 5.3.

5.1.2 Scene Interaction

Interaction with the 3D scene should emulate the behaviour of interacting with vascular anatomy. Mapping 2D actions into 3D spatial behaviour is a constant challenge for application designers. This is especially true when trying to uphold the general principle for human-computer interaction which emphasises the importance of a predictable and reproducible behaviour [12]. Managing 3D interactions incorporates 6 DOF, 3 axes for positioning and 3 for rotation. To maintain a level of consistency and simplicity, all traditional desktop interactions were developed around the standard 3-button mouse. The left mouse button is used to move objects when clicked and dragged into a new location. Holding and dragging with the right mouse button rotates the objects until the button is released.

The movement, or translation, of the vessel is based around the ray-casting technique. When a user clicks within the application a virtual line is generated from the position of the cursor into the scene. If this ray collides with the vessel, we consider it selected and collect the plane on which it was selected. If the user then holds and drags the mouse button, the next position is calculated from the new ray's position on the previously selected plane. As this process is captured within the main loop of the application, it appears seamless. Calculating the collision of the ray with the vessel mesh would be inefficient if the application was to naively query every triangle in the scene for a collision within a mesh. To prevent any performance impact, we quantify the vessel into an axis-aligned bounding box (AABB), a closed volume that completely encompasses the triangle mesh. Consequently, the application queries against the AABB of the vessel, a much simpler and faster bounds calculation.

The rotation is modelled as a virtual trackball, a common and convenient method for handling 3D rotations. Virtual trackballs simulate a physical trackball, commonly not displayed on screen, but conceptualised at the centre of the object and having a

size proportional to the object's size [63]. Finally, the middle mouse button indicates the position to add a blockage to the vessel. In addition, five keyboard shortcuts have been mapped to allow quick access to common viewing angles, including: the vessel's initial position, 90-degree rotations for each axis and one 90-degree rotation on every axis. Similarly, the arrow keys can be used to rotate the vessel precisely.

5.2 Using the zSpace Display

Figure 5.2 Real patient geometry rendering on the immersive zSpace 200 display (artistic impression used in figure to depict 3D effect, derived from (Source: zSpace, Inc [204]))

To further the opportunities for clinicians with access to immersive technologies, the prototype was developed with the capability to operate on the zSpace hardware. The zSpace system allows the user of the simulation to immersive themselves within a virtual environment and interact with the vessels as if they were a real physical entity. The stereoscopic display and glasses give the vessel the appearance of protruding

beyond the monitor, and react to the user's head movements, rotating the rendering of the scene to mimic natural human sight, see Figure 5.2. Moreover, through the use of a 6 degrees of freedom stylus the imported geometry can be manipulated and inspected in a natural and intuitive way.

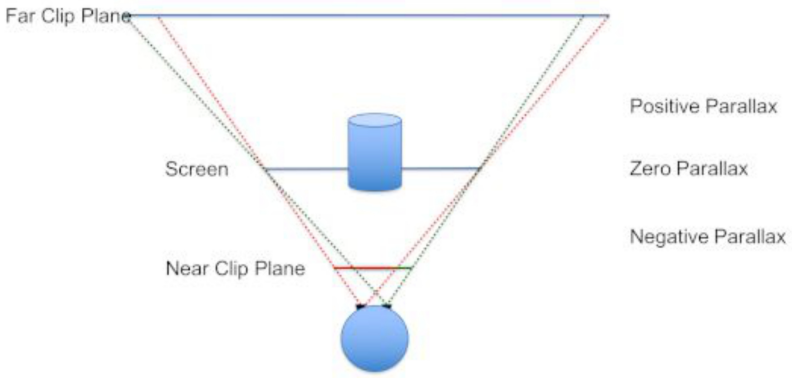


Figure 5.3 2D quad buffering configuration with two frustums simulating what happens in the human visual system (Source: zSpace, Inc [204]).

The zSpace stereo uses a quad buffer, time sequential stereo style of left/right image generation. The format is natively supported by OpenGL but requires specific hardware support from the graphics device vendor, usually found within the workstation graphics range, e.g. NVIDIA Quadro and AMD FirePro. In non-stereoscopic 3D graphics applications, the rendering system uses a monoscopic frustum, i.e. a single pyramid of vision into the modelled world. This frustum defines the edges of vision and the near and far clipping planes. However, humans have two eyes and are therefore using two frustums, generating two images which are processed by the brain to create a sense of depth. For stereoscopic 3D applications to create the same sense of depth we also must use two frustums, see Figure 5.3. To render the stereo images, the scene is rendered in two frames, once for each eye, only visible through the complimenting lens of the polarised stereoscopic glasses. The matrices used to format the scene have different values for each eye, due to the viewing angles and space between the eyes, this mandates that they are recalculated for each frame, see Source Code 5.1.

To successfully operate the stylus we needed to implement a 3D cursor that followed the tip of the stylus and allowed the user access into the depth of the visualisation. There are at least three techniques for pointer length: fixed, variable, and hybrid [28]. We chose to employ a variable pointer to enable to user maximum range of movement when manipulating the scene. Due to earlier design choices, the stylus quickly integrated into the ray casting objecting picking scheme. The ray could instead be generated from the position and direction of the stylus. With the addition of the stylus direction, the necessity for the selected plane with mouse picking is removed and the vessel can be dragged freely around the scene, including towards and away from the user. Similarly, rotation of the vessel is also more natural as the stylus can be turned and rotated itself. The rotation of the stylus can be queried from the zSpace hardware and used to manipulate the vessel precisely.

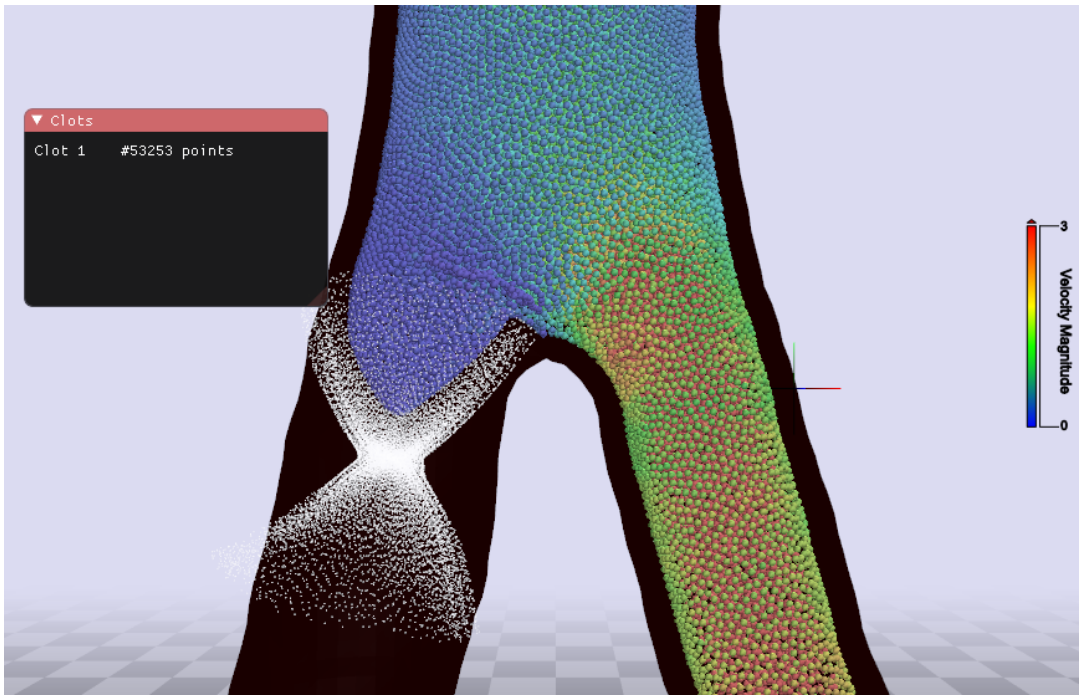
5.3 Simulation Interaction

For the clinicians to generate bespoke flow scenarios we endeavoured to create meaningful interaction with the blood flow. As our investigation was primarily concerned with the re-circulation behaviour, we aimed to manufacture flow re-circulation by allowing the user to create blockages. As a result of the simulation optimisation, this blockage-based interaction could be applied and removed from the geometry in real-time while the simulation was running. In practice, a user was able to render the centreline, and using their mouse or the zSpace stylus, point to a region of the geometry to create a blockage. Multiple blockages can be generated within one scenario, only limited by the size of the vessel loaded.

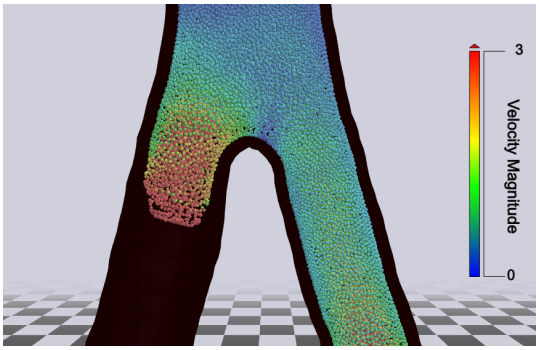
Initially we allowed the user to ‘draw’ a simple sphere along the centreline. These spheres were difficult to create, and they also generated distinctly artificial geometry. The structures generated were unnatural and unlike what is found in a patient with

Source Code 5.1 zSpace camera variants to retrieve the alternating view and perspective matrices. The zSpace camera inherits the normal mono view methods from the camera used for conventional screens.

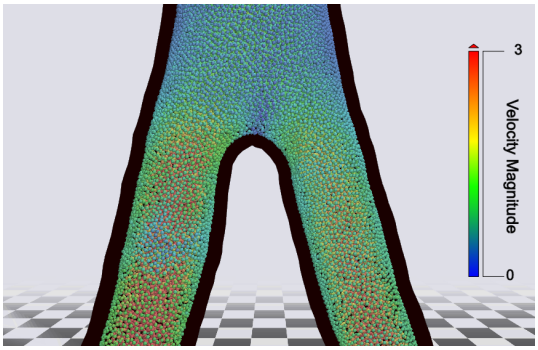
```
glm::mat4 GLZSpaceCamera::view() const
{
  // Get the view matrix from the zSpace StereoFrustum for the
   ↪ specified eye.
  ZSMatrix4 view_matrix;
  const auto error =
   ↪ zcGetFrustumViewMatrix(zc_context.frustum_handle,
   ↪ zc_context.eye, &view_matrix);
  const auto zs_view_matrix{ glm::make_mat4(view_matrix.f) };
  const auto mono_view{ super::view() };
  return zs_view_matrix * mono_view;
}

glm::mat4 GLZSpaceCamera::projection() const
{
  // Get the projection matrix from the zSpace StereoFrustum
   ↪ for a specified eye.
  ZSMatrix4 projection_matrix;
  ZCError error = zcGetFrustumProjectionMatrix(
   ↪ zc_context.frustum_handle, zc_context.eye,
   ↪ &projection_matrix);
  auto zs_proj = glm::make_mat4(projection_matrix.f);
  return zs_proj;
}
```



(a) Clot structure, rendered as white particles, within real patient geometry that features a bifurcation. Flow modelled from top to bottom.



(b) Flow developing moments after the clot structure is removed.



(c) Stable flow returning after the clot structure has been removed.

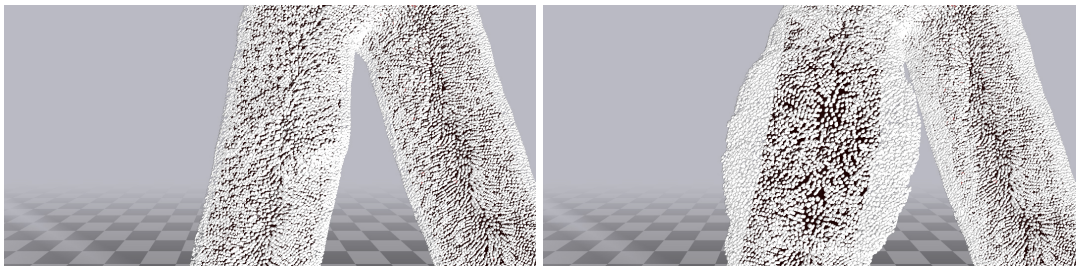
Figure 5.4 Evaluating the consequences of removing blockages in bifurcation geometry.

atherosis or thrombosis. Instead we developed a system to use the existing boundary particles. The blockage is created by transforming the boundary particles near the user's selection. Particles are selected from within a fixed radius of the user selection and those particles are copied. The position of the newly copied particle is altered to move along their previous normal vector, effectively closing the distance to the centreline. All the particles selected are moved closer to the centreline, but the magnitude of their displacement is dependent on their distance from the original user selection. This creates a natural gradient, which would more accurately model biological effects such as atheroma. The new particles are labelled and added into the boundary particle collection, meaning that the addition seamlessly enters into the general SPH algorithm, resulting in negligible effects on performance.

The 'clot' created then appears within the demonstrator applications 'clots' panel and can be removed at will, see Figure 5.4a. The simulated flow in Figure 5.4a has become trapped behind the clot and has been forced down through the second child vessel. Small areas of circulation behind the obstruction are also observable in and around the blocked regions of flow. When the clot is removed, there is an immediate shift in the flow, the previously blocked, blue particles, become red to highlight their increase in velocity, see Figure 5.4b. Subsequently, the flow within the second child vessel slows as the pressure is relieved, becoming blue and green in colour. Finally, the flow becomes completely stable, as shown in Figure 5.4c.

Following the blockage interaction, the second interaction desired by clinicians was narrowing the vessel geometry. This interaction was designed to imitate the build-up and potential release of atherosclerotic plaque. Users hovering their mouse or pointing their stylus over a region of the vessel may then scroll with their mouse-wheel or drag with their stylus. Subsequently, the vessel boundary particles in proximity to the user selection will begin to narrow or bulge around the desired region, depending on the direction of the scroll, see Source Code 5.2. The boundary particles affected are

then remapped into the correct grid cells and sorted, a process required by the normal fluid particles on each time-step and causing only minor performance degradation during the affected frame cycle. The interaction can be visualised by enabling the rendering of boundary particles, see Figure 5.5.



(a) White boundary particles rendered before any user interaction.

(b) White boundary particles rendered around expanded region of patient geometry.

Figure 5.5 Transformation of boundary particles after user narrowing/expanding interaction.

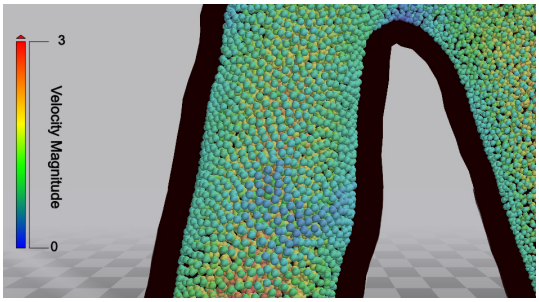
Figure 5.2 demonstrates the capabilities and resulting fluid behaviours of the prototype narrowing interaction operating in real-time. As previously explored in the cone test case, 4.2.2, the cross section shown in Figure 5.6b highlights the increase in relative velocity caused by the greatly reduced vessel radius. Additionally, we also observe that the region causes a bottleneck, resulting in a slow-moving region behind the partial blockage. Conversely, when a region of the vessel is manually expanded into a bulging state, the relative velocity is greatly reduced through the affected region, confining a portion of fluid within the extremities of the vessel, see Figure 5.6d.

5.4 Summary

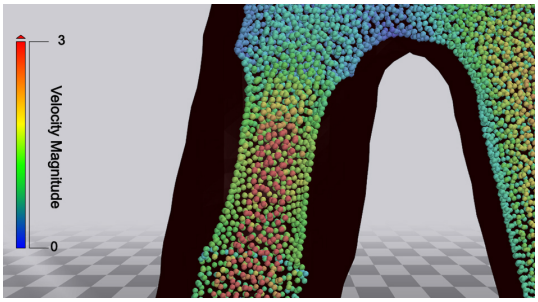
The usage and interaction with the prototype application implementing the new simulation engine has been described. These user controls and novel interactions were developed to enable clinicians to interactively create numerous scenarios and evaluate the haemodynamic consequences. In addition, the application of immersive

Source Code 5.2 CUDA functor to manipulate the boundary particles within a radius of a selected point.

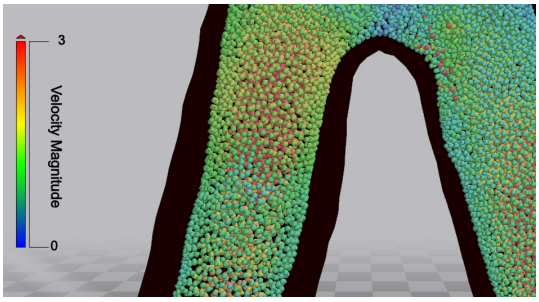
```
struct shrink_on_scroll_functor {
  float amount_to_move, float radius_of_effect;
  float3 point_of_interest;
  sph_parameters params_;
  explicit shrink_on_scroll_functor(const float3 point, const
   ↪ float radius, const float amount, sph_parameters params)
   ↪ : point_of_interest(point), amount_to_move(amount),
   ↪ radius_of_effect(radius), params_(params) {}

  template <typename Tuple>
  __device__ void operator()(Tuple t) {
    const float3 pos{ thrust::get<0>(t) }; // boundary position
    const float dist = sqrt(distance_squared(pos,
     ↪ point_of_interest));
    if (dist < radius_of_effect) {
      const auto direction = normalize(pos -
       ↪ point_of_interest);
      const auto percentage_between = dist / radius_of_effect;
      const auto new_pos = pos + (direction *
       ↪ (percentage_between * amount_to_move));
      thrust::get<0>(t) = new_pos;

      const auto ref = uniform_grid::calc_grid_hash(
        uniform_grid::calc_grid_cell(new_pos,
         ↪ params_.world_origin, params_.cell_size),
        params_.grid_size
      );
      ref > params_.num_cells ? thrust::get<1>(t) =
       ↪ UNDEFINED_CELL : thrust::get<1>(t) = ref;
    }
  }
};
```



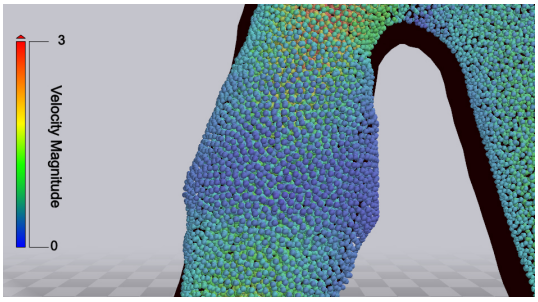
(a) Real patient geometry featuring the aortic bifurcation before applying narrowing interaction.



(b) Cross section of the narrowed region, demonstrating an increased relative velocity within the artificially narrowed region.



(c) Resulting expansion of the geometry while the user gradually releases the narrowed region of geometry.



(d) Real patient bifurcation expanded by user interaction. Example of boundary representation seen in Figure 5.5.

Figure 5.6 Evaluating the consequences of interactively narrowing and expanding a region of the bifurcation geometry.

technology, specially the zSpace hardware, was used to supplement the interactive capabilities of the application. The tool will be further updated for more intuitive clinical use, including hiding the advanced technical details from the user. The application was also demonstrated to an interventional radiologist. The feedback received from the demonstration was positive and a number of comments and suggestions were taken. In particular, similarities were drawn to the results seen from experience in the application of contrast medium during Angiography. The initial propagation of particles was noted to closely resemble the progression of contrast medium in X-ray imagery. A full clinical trial of the decision support tool is a suitable avenue for future work.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The decision-making process of cardiovascular clinicians approaching a patient with uncertain prospects is fraught with uncertainty and insufficient tooling. Potentially life-altering decisions are made on the basis of grainy medical image data and previous experience. There are no tools at the disposal of clinicians to rapidly prototype various treatment scenarios to aid their understanding of a specific problem domain. As computer hardware has evolved, we have seen the rapid acceleration of affordable computational power. This new-found computational potential can potentially provide benefits to the medical domain. Firstly, we can apply the new hardware to computational methods of simulating fluid dynamics, aiming to generate patient-specific results accurately in real-time. Secondly, we are seeing more availability and capability to render immersive environments, supporting new levels of interaction and visualisation. Previous implementations of accurate fluid simulations have required extended periods of processing time to generate results. In contrast, simulations that produce results in real-time have been designed principally for visually credible, high fidelity visualisations, unsuitable for validation.

The aim of this investigation was to explore this new potential, to develop and validate a bespoke patient-specific blood flow simulation and visualisation prototype that produces results in real-time. A prototype real-time blood simulation and visualisation tool was developed to test the thesis hypothesis:

> 'A real-time computer simulation and visualisation of blood flow through vascular structures can be developed, validated and integrated with the training and decision-making process of clinical interventions during an acute cardiac event.'

Furthermore, in Chapter 1 we outlined three research objectives that when met would address this hypothesis. Firstly, we aimed to produce the simulation engine for a modern prototype surgical planning tool to aid in the clinicians decision making processes by implementing a 3D real-time CFD simulation that operates within complex patient geometry. Following that objective, in Chapter 3 we have catalogued the development of a functional GPU accelerated real-time blood flow prototype. The CFD simulation operates using the SPH method, and by implementing an effective boundary condition, particle seeding configuration, and mesh preprocessing, the simulation will perform interactively inside real patient-geometry.

Secondly, we aimed to validate the results of the simulation, proving the accuracy of the CFD model for use by clinicians. Consequently, Chapter 4 demonstrated the results of the simulation from a variety of test cases were validated against analytical results and measurements from the literature. In addition, the simulation was tested within regions of real-patient geometry from the cardiovascular system. Interactive SPH applications are typically unsuitable for such measurements of accuracy, the quantity of particles is often too low, or the accuracy has been lost as a consequence of visual fidelity. We have engineered a balance between performance and accuracy sufficient for validation. The validation of the simulation was particularly important

for this research as clinicians must trust that the results of the simulation are indicative of reality to inform their treatment plans.

Our final objective was to visualise the simulation results in 3D to create an immersive and interactive experience for the user. Allowing the user to interact with the fluid flowing through the patient geometry and configure the aspects of the visualisation. By implementing the SPH fluid simulation method with GPU accelerated techniques we were able to perform the simulation and visualise the results in real-time. The application performs at interactive rates, the frame-rate allowing suitable visual continuity for an immersive experience. The prototype could potentially be used by clinicians to evaluate the haemodynamic consequences of scenarios generated in a non-invasive and immersive 3D environment, including the use of immersive 3D hardware if available to the user. Novel interactions designed for clinical use can be performed upon the simulated flow and geometry, allowing users to generate and remove blockages, dictating the pathways available for the blood flow. These scenarios can be based within real patient geometry, as surface meshes derived from the segmentation of medical imagery can be loaded into the application with no additional manual processing. Moreover, the workflow of the prototype application is semi-automated, automatically subjecting the loaded geometry to preprocessing, generating the vessel centreline, boundary particles, and particle seeding locations.

As a result, we can conclude that with modern hardware acceleration and optimisations for the SPH algorithm the development of an accurate real-time blood flow simulation and visualisation tool suitable for clinical and educational use is plausible. We have verified that the application is robust and can handle a selection of vastly different geometric structures, while previous simulations focus on one specific area of anatomy. Consequently, this development supports our hypothesis that real-time simulation and visualisation of blood flow has the potential to be applied to the decision-making process of clinical interventions.

6.2 Future Work

Although the potential to be an effective addition to the clinical decision-making process has been demonstrated, the software developed is a prototype for a subsequent full application, and this multidisciplinary investigation creates multiple avenues for future work. Firstly, the prototype application must be subject to further testing and development in a hospital setting. A clinical user study would provide greater insights into the applications clinical and educational effectiveness. A larger sample of clinicians expectations, behaviours, and investigative requirements would shape the prototype for clinical use. Furthermore, feedback from a user study may highlight desirable interactions that could be implemented, enabling a greater depth to scenario development.

Secondly, the simulation engine could be developed further to target a greater quantity of flow behaviours and improve those already developed. For example, collateral circulation is particularly important during some acute cardiac events and has yet to be investigated within the prototype application. A geometric model featuring a large quantity of potentially alternative blood flow routes to a common outlet could be developed and fabricated to replicate the behaviour, using patient specific geometry similar to that used in Section 4.3.3.

The boundary condition has been implemented and produces the required behaviours. However, the prototype application treats the boundary as a profoundly static entity. In real examples the vessel is highly reactive with regards to the blood flow. The simulation engine would benefit from the development of a type of elastic boundary that reacts to the flow in much the same way. To produce accurate results in real-time with such a boundary would be a large undertaking and is outside of the scope of this research endeavour. In addition, some geometric structure's inlet

produces an area of abnormally high density around the seeding area before stabilising, which could be improved upon with a more consistent seeding pattern along the inlet.

The nature of the cardiovascular system produces a pulsatile flow, the beating heart pumps blood into arteries which then propagates throughout the whole system. The behaviour of the blood flow during diastole and systole, the two phases of the cardiac cycle, are markedly different in some cardiovascular structures. However, the simulation generally models steady continuous flow through the patient geometry. Although the particle lifetime, seeding and re-seeding, does produce a pulsatile flow effect within the visualisation, users have limited control over the behaviour with current simulation parameters.

Similarly, the simulation parameters themselves would benefit from a form of calibration with empirical physiologic flow parameters. Some of the test cases developed are completely artificial and are therefore only subject to speculative parameters, or generally accepted density values for blood. As the patient-specific geometry is for research purposes we have no measurements for physiologic parameters such as inlet velocity and density. In a further clinical test, there may be opportunity to measure such parameters and to pair them with the associated geometry, creating a valuable simulation validation environment.

In conversation with clinicians the addition of contrast mediums was proposed to imitate the current physical processes of diagnosis. This is thought to enable a further aid for education and potentially validation. Contrast fluids could be implemented with a variety of methods. Firstly, a colour property could be added to the fluid, and would be subject to its own smoothing kernel, spreading the colour throughout the fluid. Secondly, SPH has seen some implementations with multiple fluid types, commonly using different mass for each type of fluid. Furthermore, there is some research into two-way coupled implementations of SPH, simulating diffuse regions such as sprays [99].

Alterations and additions to the prototype clinical tool could also be pursued. A key feature we imagine in future iterations would be the capability to move the simulation backwards in time, a difficult process for GPU accelerated algorithms. To remember the positions and properties of particles in previous time-steps could be an interesting challenge. Storing data to disk is generally a bottleneck for performant applications, especially as the data would need to stream from the GPU. Memory management on the GPU limits the amount of data that could be stored without affecting performance.

Software optimisation is an iterative process and all applications can be subject to further steps to increase performance. However, there are new unconventional routes to GPU usage to be explore. For instance, CUDA kernels, and other graphics programming frameworks, have recently been made available to run in cloud configurations, such as Google Cloud Compute [30] and Amazon EC2 P2 [143]. Potentially offloading the computational cost from the rendering machine and away from environments that would have a huge upfront cost, i.e. multiple GPU configurations. In particular, stereo rendering requires specific workstation GPUs which generally have a much higher cost than their consumer counterpart. The potential challenge to using the cloud in this application would be mitigating the latency created when transferring the results downstream to be rendered and modifications from user interactions back to the cloud.

In conclusion, we have developed and validated a modern, real-time, interactive, blood flow simulation and visualisation which can be applied to geometry from the cardiovascular domain. Feedback from clinicians supports the potential for further development into 3D clinical decision support applications for interventional cardiologists. The consistent improvements in computational hardware and immersive technologies will accelerate the integration of such tools into the medical arsenal, facilitating improved patient-specific care and clinical support.

References

[1] S Adami, XY Hu, and NA Adams. “A generalized wall boundary condition for smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 231.21 (2012), pp. 7057–7075.

[2] James Ahrens, Berk Geveci, and Charles Law. “Paraview: An end-user tool for large data visualization”. In: *The visualization handbook* 717 (2005).

[3] Nadir Akinci et al. “Coupling elastic solids with smoothed particle hydrodynamics fluids”. In: *Computer Animation and Virtual Worlds* 24.3-4 (2013), pp. 195–203.

[4] J.D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill international editions. McGraw-Hill, 1995. ISBN: 9780071132107. URL: https://books.google.co.uk/books?id=phG%5C_QgAACAAJ.

[5] Luca Antiga and David A Steinman. “Robust and objective decomposition and mapping of bifurcating vessels”. In: *IEEE transactions on medical imaging* 23.6 (2004), pp. 704–713.

[6] Luca Antiga et al. “An image-based modeling framework for patient-specific computational hemodynamics”. In: *Medical & biological engineering & computing* 46.11 (2008), p. 1097.

[7] Bassem F Armaly et al. “Experimental and theoretical investigation of backward-facing step flow”. In: *Journal of fluid Mechanics* 127 (1983), pp. 473–496.

[8] Christopher J Arthurs et al. “Reproducing patient-specific hemodynamics in the Blalock–Taussig circulation using a flexible multi-domain simulation framework: applications for optimal shunt design”. In: *Frontiers in pediatrics* 5 (2017), p. 78.

[9] L. Avila and M. Bailey. “Virtual Reality for the Masses”. In: *IEEE Computer Graphics and Applications* 34.05 (Sept. 2014), pp. 103–104. ISSN: 0272-1716. DOI: 10.1109/MCG.2014.103.

[10] Utkarsh Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015. ISBN: 978-1930934306.

[11] Portonovo S Ayyaswamy. “Introduction to biofluid mechanics”. In: *Fluid Mechanics*. Elsevier, 2012, pp. 779–852.

[12] Ragnar Bade, Felix Ritter, and Bernhard Preim. “Usability comparison of mouse-based interaction techniques for predictable 3D rotation”. In: *International Symposium on Smart Graphics*. Springer. 2005, pp. 138–150.

[13] Jon Barrilleaux. *3D user interfaces with Java 3D*. Manning Greenwich, 2001.

[14] Cx K Batchelor and GK Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[15] David W Bates et al. “Reducing the frequency of errors in medicine using information technology”. In: *Journal of the American Medical Informatics Association* 8.4 (2001), pp. 299–308.

[16] BBC News and Charlie Coleman. *Manchester Royal Infirmary’s new heart surgery suite completed*. June 2020. URL: https://www.bbc.co.uk/news/uk-england-manchester-27943441 (visited on 01/06/2020).

[17] Markus Becker and Matthias Teschner. “Weakly compressible SPH for free surface flows”. In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics*

symposium on Computer animation. Eurographics Association. 2007, pp. 209–217.

[18] J Belanger, P Venne, and JN Paquin. "The what, where and why of real-time simulation". In: *PES General Meeting*. IEEE, Oct. 2010, pp. 25–29.

[19] E Wes Bethel, Hank Childs, and Charles Hansen. *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*. CRC Press, 2012.

[20] René Botnar et al. "Hemodynamics in the carotid artery bifurcation:: a comparison between numerical simulations and in vitro MRI measurements". In: *Journal of biomechanics* 33.2 (2000), pp. 137–144.

[21] Doug Bowman et al. *3D User interfaces: theory and practice, CourseSmart eTextbook*. Addison-Wesley, 2004.

[22] Robert Bridson. "Fast Poisson disk sampling in arbitrary dimensions." In: *SIGGRAPH sketches*. 2007, p. 22.

[23] British Heart Foundation. *British Heart Foundation UK Factsheet*. Nov. 2018. URL: https://www.bhf.org.uk/-/media/files/research/heart-statistics/bhf-cvd-statistics---uk-factsheet.pdf?la=en (visited on).

[24] Morten Bro-Nielsen and Stephane Cotin. "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation". In: *Comput. Graph. Forum* 15 (1996), pp. 57–66.

[25] Brian H Brown et al. *Medical physics and biomedical engineering*. CRC Press, 2017.

[26] *Cardiovascular diseases (CVDs)*. 2017. URL: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[27] Elliot L Chaikof et al. "The Society for Vascular Surgery practice guidelines on the care of patients with an abdominal aortic aneurysm". In: *Journal of vascular surgery* 67.1 (2018), pp. 2–77.

[28] Clifford Champion et al. *Style Guide Best Practices*. 2017. URL: https://cdn.zspace.com/downloads/documentation/developer/style-guide-best-practices.pdf/ (visited on 03/14/2019).

[29] Jim X Chen and Niels da Vitoria Lobo. "Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations". In: *Graphical Models and Image Processing* 57.2 (1995), pp. 107–116.

[30] *Cloud GPUs*. URL: https://cloud.google.com/gpu/ (visited on 09/24/2019).

[31] John P Cooke et al. "Sex differences in control of cutaneous blood flow." In: *Circulation* 82.5 (1990), pp. 1607–1615.

[32] Nicu D Cornea, Deborah Silver, and Patrick Min. "Curve-skeleton properties, applications, and algorithms". In: *IEEE Transactions on Visualization & Computer Graphics* 13.3 (2007), pp. 530–548.

[33] NVIDIA Corporation. *NVIDIA FleX*. 2020. URL: https://developer.nvidia.com/flex (visited on 03/31/2020).

[34] NVIDIA Corporation. *PhysX*. 2020. URL: https://www.geforce.com/hardware/technology/physx (visited on 03/31/2020).

[35] Katia Cortese et al. "Chapter 6 - 3D HDO-CLEM: Cellular Compartment Analysis by Correlative Light-Electron Microscopy on Cryosection". In: *Correlative Light and Electron MIcroscopy*. Ed. by Thomas Müller-Reichert and Paul Verkade. Vol. 111. Methods in Cell Biology. Academic Press, 2012, pp. 95–115. DOI: 10.1016/B978-0-12-416026-2.00006-6.

[36] AJC Crespo, M Gómez-Gesteira, Robert A Dalrymple, et al. "Boundary conditions generated by dynamic particles in SPH methods". In: *CMC-TECH SCIENCE PRESS*- 5.3 (2007), p. 173.

[37] Sharen J Cummins and Murray Rudman. "An SPH projection method". In: *Journal of computational physics* 152.2 (1999), pp. 584–607.

[38] Gianluca De Santis et al. “Full-hexahedral structured meshing for image-based computational vascular modeling”. In: *Medical engineering & physics* 33.10 (2011), pp. 1318–1325.

[39] Michael F Deering. “The limits of human vision”. In: *2nd International Immersive Projection Technology Workshop*. Vol. 2. 1998.

[40] Denis Demidov et al. “Programming CUDA and OpenCL: A case study using modern C++ libraries”. In: *SIAM Journal on Scientific Computing* 35.5 (2013), pp. C453–C472.

[41] Peter J Denning and Ted G Lewis. “Exponential laws of computing growth”. In: *Communications of the ACM* (2017).

[42] Mathieu Desbrun and Marie-Paule Cani. “Smoothed Particles: A new paradigm for animating highly deformable bodies”. In: *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*. Ed. by Ronan Boulic and Gerard Hegron. Published under the name Marie-Paule Gascuel. Poitiers, France: Springer-Verlag, Aug. 1996, pp. 61–76. URL: https://hal.inria.fr/inria-00537534.

[43] Steven Diehl et al. “Generating optimal initial conditions for Smoothed Particle Hydrodynamics simulations”. In: *Publications of the Astronomical Society of Australia* 32 (2015).

[44] Jose M. Domínguez, Alejandro J.C. Crespo, and Moncho Gómez-Gesteira. “Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method”. In: *Computer Physics Communications* 184.3 (2013), pp. 617–627. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2012.10.015.

[45] Peng Du et al. “From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming”. In: *Parallel Computing* 38.8 (2012), pp. 391–407.

[46] DualSPHysics. *DualSPHysics: GPU and OpenMP based Smoothed Particle Hydrodynamics*. 2020. URL: https://dual.sphysics.org/ (visited on 03/31/2020).

[47] David H Eberly. *GPGPU Programming for Games and Science*. CRC Press, 2014.

[48] *Encounter the next era of computing*. 2018. URL: https://www.magicleap.com/ (visited on 09/24/2019).

[49] Facebook Technologies, LLC. *Oculus Rift*. URL: https://www.oculus.com/rift (visited on 03/31/2020).

[50] Nuno Faria, Rui Silva, and Joao L Sobral. "Impact of data structure layout on performance". In: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE. 2013, pp. 116–120.

[51] Mark L Field et al. "Hybrid theatres: nicety or necessity?" In: *Journal of the Royal Society of Medicine* 102.3 (2009), pp. 92–97.

[52] Mark Fillinger. "Who should we operate on and how do we decide: predicting rupture and survival in patients with aortic aneurysm". In: *Seminars in vascular surgery*. Vol. 20. 2. Elsevier. 2007, pp. 121–127.

[53] Nick Foster and Dimitri Metaxas. "Realistic animation of liquids". In: *Graphical models and image processing* 58.5 (1996), pp. 471–483.

[54] Alain Fournier and William T Reeves. "A simple model of ocean waves". In: *ACM Siggraph Computer Graphics* 20.4 (1986), pp. 75–84.

[55] Amit X Garg et al. "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review". In: *Jama* 293.10 (2005), pp. 1223–1238.

[56] David K Gartling. "A test problem for outflow boundary conditions-flow over a backward-facing step". In: *International Journal for Numerical Methods in Fluids* 11.7 (1990), pp. 953–967.

[57] Moncho Gomez-Gesteira et al. “State-of-the-art of classical SPH for free-surface flows”. In: *Journal of Hydraulic Research* 48.S1 (2010), pp. 6–27.

[58] Michael E Goss. “A real time particle system for display of ship wakes”. In: *IEEE Computer Graphics and Applications* 10.3 (1990), pp. 30–35.

[59] Henry Gray. *Anatomy of the Human Body*. 20th ed. Philadelphia: Lea & Febiger, 2001. ISBN: 1-58734-102-6.

[60] Chris Gregg and Kim Hazelwood. “Where is the data? Why you cannot debate CPU vs. GPU performance without the answer”. In: *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE. 2011, pp. 134–144.

[61] Sergey Grizan et al. “dJay: enabling high-density multi-tenancy for cloud gaming servers with dynamic cost-benefit GPU load balancing”. In: *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM. 2015, pp. 58–70.

[62] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. “Smoothed particle hydrodynamics on GPUs”. In: *Computer Graphics International*. Vol. 40. SBC Petropolis. 2007, pp. 63–70.

[63] Knud Henriksen, Jon Sporring, and Kasper Hornbæk. “Virtual trackballs revisited”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.2 (2004), pp. 206–216.

[64] B.G. Hinwood. *A Textbook of Science for the Health Professions*. Nelson Thornes, 1997. ISBN: 9780748733774. URL: https://books.google.co.uk/books?id=lWGESsimCggC.

[65] Jared Hoberock and Nathan Bell. *Thrust - Parallel Algorithms Library*. 2009. URL: https://thrust.github.io/ (visited on 03/18/2019).

[66] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 1988.

[67] HTC Corporation. *Room-Scale VR*. URL: https://www.vive.com/uk/product/vive/ (visited on 03/31/2020).

[68] Howard H Hu, Neelesh A Patankar, and MY Zhu. "Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique". In: *Journal of Computational Physics* 169.2 (2001), pp. 427–462.

[69] Eero Huotilainen et al. "Inaccuracies in additive manufactured medical skull models caused by the DICOM to STL conversion process". In: *Journal of Cranio-Maxillofacial Surgery* 42.5 (2014), e259–e265.

[70] Markus Ihmsen et al. "SPH Fluids in Computer Graphics". In: *Eurographics*. The Eurographics Association, 2014.

[71] Candace Imison et al. *Delivering the benefits of digital health care*. Nuffield Trust London, UK, 2016.

[72] Imperial College Healthcare NHS Trust. *Trust awarded 1.9m for state-of-the-art surgical and imaging theatre*. Dec. 2018. URL: https://www.imperial.nhs.uk/about-us/news/trust-awarded-funding-for-state-of-the-art-surgical-and-imaging-theatre (visited on 04/08/2019).

[73] COMSOL Inc. *COMSOL Multiphysics Modeling Software*. 2020. URL: https://uk.comsol.com/ (visited on 03/31/2020).

[74] Intel. *Raja Koduri Joins Intel as Chief Architect to Drive Unified Vision across Cores and Visual Computing*. (Visited on 09/27/2019).

[75] Keiichi Itatani et al. "Influence of surgical arch reconstruction methods on single ventricle workload in the Norwood procedure". In: *The Journal of thoracic and cardiovascular surgery* 144.1 (2012), pp. 130–138.

[76] Nigel W John et al. "The use of stereoscopy in a neurosurgery training virtual environment". In: *Presence: Teleoperators and Virtual Environments* 24.4 (2016), pp. 289–298.

[77] Barbara M Johnston et al. “Non-Newtonian blood flow in human right coronary arteries: steady state simulations”. In: *Journal of biomechanics* 37.5 (2004), pp. 709–720.

[78] Yannis G Kallinderis and Judson R Baron. “Adaptation methods for a new Navier-Stokes algorithm”. In: *AIAA journal* 27.1 (1989), pp. 37–43.

[79] Kamran Karimi, Neil G Dickson, and Firas Hamze. “A performance comparison of CUDA and OpenCL”. In: *arXiv preprint arXiv:1005.2581* (2010).

[80] Michael Kass and Gavin Miller. “Rapid, stable fluid dynamics for computer graphics”. In: *ACM Siggraph Computer Graphics*. Vol. 24. ACM. 1990, pp. 49–57.

[81] Abbas Khayyer et al. “An enhanced ISPH–SPH coupled method for simulation of incompressible fluid–elastic structure interactions”. In: *Computer Physics Communications* 232 (2018), pp. 139–164.

[82] Hyoung-Bum Kim et al. “Noninvasive measurement of steady and pulsating velocity profiles and shear rates in arteries using echo PIV: in vitro validation studies”. In: *Annals of biomedical engineering* 32.8 (2004), pp. 1067–1076.

[83] Torvid Kiserud, Leif Rune Hellevik, and Mark Adrian Hanson. “Blood velocity profile in the ductus venosus inlet expressed by the mean/maximum velocity ratio”. In: *Ultrasound in medicine & biology* 24.9 (1998), pp. 1301–1306.

[84] Torvid Kiserud, Svein Rasmussen, and Svein Skulstad. “Blood flow and the degree of shunting through the ductus venosus in the human fetus”. In: *American journal of obstetrics and gynecology* 182.1 (2000), pp. 147–153.

[85] Inc Kitware. “The VTK User’s Guide, 11 Edition”. In: *Kitware Inc* (2010).

[86] Inc. Kitware. *Home Page - Kitware, Inc.* URL: https://www.kitware.com/ (visited on 01/18/2020).

[87] Inc. Kitware. *ITK | Insight Toolkit*. URL: https://itk.org/ (visited on 01/18/2020).

[88] Inc. Kitware. *VTK - The Visualization Toolkit*. URL: https://vtk.org/ (visited on 01/18/2020).

[89] Scott Kulp et al. "Practical patient-specific cardiac blood flow simulations using SPH". In: *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE. 2013, pp. 832–835.

[90] Joseph J LaViola Jr et al. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[91] Hung Le, Parviz Moin, and John Kim. "Direct numerical simulation of turbulent flow over a backward-facing step". In: *Journal of fluid mechanics* 330 (1997), pp. 349–374.

[92] Lucian L Leape et al. "Systems analysis of adverse drug events". In: *Jama* 274.1 (1995), pp. 35–43.

[93] Victor W Lee et al. "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU". In: *ACM SIGARCH computer architecture news* 38.3 (2010), pp. 451–460.

[94] Yu-Chuan Li et al. "Neural network modeling for surgical decisions on traumatic brain injury patients". In: *International journal of medical informatics* 57.1 (2000), pp. 1–9.

[95] Elisa G Liberati et al. "What hinders the uptake of computerized decision support systems in hospitals? A qualitative study and framework for implementation". In: *Implementation Science* 12.1 (2017), p. 113.

[96] Gui-Rong Liu and Moubin B Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003.

[97] MB Liu and GR Liu. “Smoothed particle hydrodynamics (SPH): an overview and recent developments”. In: *Archives of computational methods in engineering* 17.1 (2010), pp. 25–76.

[98] William E. Lorensen and Harvey E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 163–169. ISSN: 0097-8930. DOI: 10.1145/37402.37422.

[99] Frank Losasso et al. “Two-way coupled SPH and particle level set fluid simulation”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.4 (2008), pp. 797–804.

[100] Leon B Lucy. “A numerical approach to the testing of the fission hypothesis”. In: *The astronomical journal* 82 (1977), pp. 1013–1024.

[101] I Scott MacKenzie and Colin Ware. “Lag as a determinant of human performance in interactive systems”. In: *Proceedings of the INTERACT́93 and CHÍ93 conference on Human factors in computing systems*. ACM. 1993, pp. 488–493.

[102] Miles Macklin and Matthias Müller. “Position based fluids”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 104.

[103] Monier Maher et al. *Physics processing unit*. Apr. 7, 2005. URL: http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%5C%2Fnetahtml%5C%2FPTO%5C%2Fsrchnum.html&r=1&f=G&l=50&s1=%5C%2220050075849%5C%22.PGNR.&OS=DN/20050075849&RS=DN/20050075849.

[104] Hani R Malone et al. “Simulation in neurosurgery: a review of computer-based simulation environments and their surgical applications”. In: *Neurosurgery* 67.4 (2010), pp. 1105–1116.

[105] S. Manini and Luca Antiga. 2013. URL: http://www.vmtk.org (visited on 03/18/2019).

[106] S. Manini and Luca Antiga. 2013. URL: http://www.vmtk.org/tutorials/Centerlines.html (visited on 03/18/2019).

[107] Rory McCloy and Robert Stone. "Virtual reality in surgery". In: *Bmj* 323.7318 (2001), pp. 912–915.

[108] Allison J McLarty et al. "Surveillance of moderate-size aneurysms of the thoracic aorta". In: *Journal of cardiothoracic surgery* 10.1 (2015), p. 17.

[109] S. Mendis et al. *Global Atlas on Cardiovascular Disease Prevention and Control*. Nonserial Publications Series. World Health Organization in collaboration with the World Heart Federation and the World Stroke Organization, 2011. ISBN: 9789241564373. URL: https://books.google.co.uk/books?id=ZRbKygAACAAJ.

[110] Duane Merrill. *CUB Documentation*. 2011. URL: https://nvlabs.github.io/cub/ (visited on 03/18/2019).

[111] Microsoft. *Hololens 2 Mixed reality is ready for business*. 2019. URL: https://www.microsoft.com/en-us/hololens (visited on 09/24/2019).

[112] Microsoft. *Vulkan 1.2 is Here!* Feb. 7, 2020. URL: https://docs.microsoft.com/en-gb/windows/win32/direct3d/ (visited on 03/31/2020).

[113] Jaques S Milner et al. "Hemodynamics of human carotid artery bifurcations: computational studies with models reconstructed from magnetic resonance imaging of normal subjects". In: *Journal of vascular surgery* 28.1 (1998), pp. 143–156.

[114] Rajat Mittal et al. "Computational modeling of cardiac hemodynamics: current status and future outlook". In: *Journal of Computational Physics* 305 (2016), pp. 1065–1082.

[115] Hideaki Miyata, Shinichi Nishimura, and Akira Masuko. “Finite difference simulation of nonlinear waves generated by ships of arbitrary three-dimensional configuration”. In: *Journal of Computational Physics* 60.3 (1985), pp. 391–436.

[116] Joe J Monaghan. “Simulating free surface flows with SPH”. In: *Journal of computational physics* 110.2 (1994), pp. 399–406.

[117] Bongki Moon et al. “Analysis of the clustering properties of the hilbert space-filling curve”. In: *IEEE Transactions on knowledge and data engineering* 13.1 (2001), pp. 124–141.

[118] Hernán G Morales et al. “Newtonian and non-Newtonian blood flow in coiled cerebral aneurysms”. In: *Journal of biomechanics* 46.13 (2013), pp. 2158–2164.

[119] Matthias Müller, David Charypar, and Markus Gross. “Particle-based Fluid Simulation for Interactive Applications”. In: *Proceedings of the 2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA ’03. San Diego, California: Eurographics Association, 2003, pp. 154–159. ISBN: 1-58113-659-5. URL: http://dl.acm.org/citation.cfm?id=846276.846298.

[120] Matthias Müller, Simon Schirm, and Matthias Teschner. “Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics”. In: *Technology and Health Care* 12.1 (2004), pp. 25–31.

[121] United Nations. “The Future We Want”. In: *The United Nations (UN), The United Nations (UN)* (2014). URL: https://sustainabledevelopment.un.org/futurewewant.html (visited on 03/18/2019).

[122] Maxwell Lewis Neal and Roy Kerckhoffs. “Current progress in patient-specific modeling”. In: *Briefings in bioinformatics* 11.1 (2009), pp. 111–126.

[123] Gibril J Njie et al. “Clinical decision support systems and prevention: a community guide cardiovascular disease systematic review”. In: *American journal of preventive medicine* 49.5 (2015), pp. 784–795.

[124] Tiago HC Nobrega, Diego Dias Bispo Carvalho, and Aldo von Wangenheim. “Simplified simulation and visualization of tubular flows with approximate centerline generation”. In: *2009 22nd IEEE International Symposium on Computer-Based Medical Systems*. IEEE. 2009, pp. 1–7.

[125] Georg Nollert, Sabine Wich, and Anne Figel. “The cardiovascular hybrid OR-clinical & technical considerations”. In: *The Cardiothoracic Surgery Network* 14 (2011). (Visited on 08/26/2019).

[126] NVIDIA Corporation. June 2018. URL: https://nvidianews.nvidia.com/news/nvidia-accelerated-supercomputers-hit-new-highs-on-top500-list (visited on 05/03/2019).

[127] NVIDIA Corporation. *Branch Statistics*. 2015. URL: https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/branchstatistics.htm (visited on 03/19/2019).

[128] NVIDIA Corporation. Feb. 2019. URL: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html (visited on 03/19/2019).

[129] NVIDIA Corporation. Aug. 2019. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/ (visited on 09/19/2019).

[130] Oculus VR. *Oculus PC SDK*. Apr. 2018. URL: https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-performance-guidelines/ (visited on 03/14/2019).

[131] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*. Nov. 2018. URL: https://www.openmp.org/ (visited on 03/18/2019).

[132] Wai-Man Pang et al. “Orthopedics surgery trainer with PPU-accelerated blood and tissue simulation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2007, pp. 842–849.

[133] BR Patel et al. “The solution of the Reynolds averaged Navier-Stokes equations in general curvilinear coordinates and its application to vehicular aerodynamics.” In: *1. International Conference on Computer Aided Design*. 1987, pp. 29–43.

[134] Darwyn R Peachey. “Modeling waves and surf”. In: *ACM Siggraph Computer Graphics*. Vol. 20. ACM. 1986, pp. 65–74.

[135] Andreas Peer et al. “An implicit SPH formulation for incompressible linearly elastic solids”. In: *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, pp. 135–148.

[136] John S Pellerito et al. “Doppler sonographic criteria for the diagnosis of inferior mesenteric artery stenosis”. In: *Journal of Ultrasound in Medicine* 28.5 (2009), pp. 641–650.

[137] Darrell W Pepper and Juan C Heinrich. *The finite element method: basic concepts and applications with MATLAB, MAPLE, and COMSOL*. CRC press, 2017.

[138] kaiser permanente. *Fast facts*. URL: https://about.kaiserpermanente.org/who-we-are/fast-facts (visited on).

[139] Josselin Petit and Roland Brémond. “A high dynamic range rendering pipeline for interactive applications”. In: *The Visual Computer* 26.6-8 (2010), pp. 533–542.

[140] Marina Piccinelli et al. “A framework for geometric analysis of vascular structures: application to cerebral aneurysms”. In: *IEEE transactions on medical imaging* 28.8 (2009), pp. 1141–1155.

[141] F Pinto and M Meo. “Design and manufacturing of a novel shear thickening fluid composite (STFC) with enhanced out-of-plane properties and damage suppression”. In: *Applied Composite Materials* 24.3 (2017), pp. 643–660.

[142] Serban R Pop et al. “A directed particle system for optimised visualization of blood flow in complex networks.” In: *Studies in health technology and informatics* 184 (2012), pp. 330–336.

[143] *Powerful, Scalable GPU instances for high-performance computing*. URL: https://aws.amazon.com/ec2/instance-types/p2/ (visited on 02/24/2020).

[144] Martin Prosi et al. “Influence of curvature dynamics on pulsatile coronary artery flow in a realistic bifurcation model”. In: *Journal of biomechanics* 37.11 (2004), pp. 1767–1775.

[145] PukiWiki. *SPH法の重み関数 [modal weight functions]*. Feb. 2014. URL: http://www.slis.tsukuba.ac.jp/~fujisawa.makoto.fu/cgi-bin/wiki/index.php?SPH%5C%CB%5C%A1%5C%A4%5C%CE%5C%BD%5C%C5%5C%A4%5C%DF%5C%B4%5C%D8%5C%BF%5C%F4 (visited on 06/17/2019).

[146] Y Qian et al. “Computational hemodynamic analysis in congenital heart disease: simulation of the Norwood procedure”. In: *Annals of biomedical engineering* 38.7 (2010), pp. 2302–2313.

[147] Jing Qin et al. “Particle-based simulation of blood flow and vessel wall interactions in virtual surgery”. In: *Proceedings of the 2010 Symposium on Information and Communication Technology*. ACM. 2010, pp. 128–133.

[148] Amanda Randles et al. “Massively parallel models of the human circulatory system”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2015, p. 1.

[149] Bastian E Rapp. *Microfluidics: Modeling, Mechanics and Mathematics*. William Andrew, 2016.

[150] Paul Read and Mark-Paul Meyer. *Restoration of motion picture film*. Butterworth-Heinemann, 2000.

[151] Fabian Rengier et al. "3D printing based on imaging data: review of medical applications". In: *International journal of computer assisted radiology and surgery* 5.4 (2010), pp. 335–341.

[152] John J Ricotta et al. "Cardiovascular disease management: the need for better diagnostics". In: *Medical & biological engineering & computing* 46.11 (2008), pp. 1059–1068.

[153] Royal Liverpool and Broadgreen University Hospitals NHS Trust. *Home Departments Medical specialisms Surgery, theatres and operations Surgery, theatres and operations*. 2019. URL: https://www.rlbuht.nhs.uk/departments/medical-specialisms/surgery-theatres-and-operations/ (visited on 04/24/2019).

[154] Onkar Sahni et al. "Efficient anisotropic adaptive discretization of the cardiovascular system". In: *Computer methods in applied mechanics and engineering* 195.41-43 (2006), pp. 5634–5655.

[155] Richard M Satava. "Medical applications of virtual reality". In: *Journal of Medical Systems* 19.3 (1995), pp. 275–280.

[156] Michael Satran and Mike Jacobs. *HLSL*. URL: https://docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl (visited on 04/14/2019).

[157] Will Schroeder, Lydia Ng, and Josh Cates. "The ITK software guide". In: (2003).

[158] Neal E Seymour et al. "Virtual reality training improves operating room performance: results of a randomized, double-blinded study". In: *Annals of surgery* 236.4 (2002), p. 458.

[159] M.S. Shadloo, G. Oger, and D. Le Touze. "Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: Motivations, current state, and challenges". In: *Computers & Fluids* 136 (2016), pp. 11–34. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2016.05.029.

[160] Songdong Shao and Edmond YM Lo. "Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface". In: *Advances in water resources* 26.7 (2003), pp. 787–800.

[161] Kang G Shin and Parameswaran Ramanathan. "Real-time computing: A new discipline of computer science and engineering". In: *Proceedings of the IEEE* 82.1 (1994), pp. 6–24.

[162] Haroon Siddique. "NHS to receive 487m technology boost". In: *The Guardian* (July 20, 2017). URL: https://www.theguardian.com/society/2018/jul/20/nhs-to-receive-487m-technology-boost-matt-hancock (visited on 01/10/2019).

[163] Mark W Siebert and Petru S Fodor. "Newtonian and non-newtonian blood flow over a backward-facing step–a case study". In: *Proceedings of the COMSOL Conference, Boston*. 2009, p. 27.

[164] Tobias Sielhorst, Marco Feuerstein, and Nassir Navab. "Advanced medical displays: A literature review of augmented reality". In: *Journal of Display Technology* 4.4 (2008), pp. 451–467.

[165] Ida Sim et al. "Clinical decision support systems for the practice of evidence-based medicine". In: *Journal of the American Medical Informatics Association* 8.6 (2001), pp. 527–534.

[166] Jos Stam. "Stable fluids". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1999, pp. 121–128.

[167] Jos Stam. *The Art of Fluid Animation*. CRC Press, 2015.

[168] Jos Stam and Eugene Fiume. “Depicting fire and other gaseous phenomena using diffusion processes”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. 1995, pp. 129–136.

[169] Jos Stam and Eugene Fiume. “Turbulent wind fields for gaseous phenomena”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. 1993, pp. 369–376.

[170] John A Stankovic. “Misconceptions about real-time computing: A serious problem for next-generation systems”. In: *Computer* 21.10 (1988), pp. 10–19.

[171] David A Steinman. “Assumptions in modelling of large artery hemodynamics”. In: *Modeling of physiological flows*. Springer, 2012, pp. 1–18.

[172] Martin Strandberg-Larsen et al. “Is the Kaiser Permanente model superior in terms of clinical integration?: a comparative study of Kaiser Permanente, Northern California and the Danish healthcare system”. In: *BMC Health Services Research* 10.1 (2010), p. 91.

[173] Ching-Lung Su et al. “Overview and comparison of OpenCL and CUDA technology for GPGPU”. In: *2012 IEEE Asia Pacific Conference on Circuits and Systems*. IEEE. 2012, pp. 448–451.

[174] Jacob E. Sunshine et al. “Association of Adverse Effects of Medical Treatment With Mortality in the United States: A Secondary Analysis of the Global Burden of Diseases, Injuries, and Risk Factors Study”. In: *JAMA Network Open* 2.1 (Jan. 2019), e187041–e187041. ISSN: 2574-3805. DOI: 10.1001/jamanetworkopen.2018.7041. eprint: https://jamanetwork.com/journals/jamanetworkopen/articlepdf/2720915/sunshine_2019_oi_180294.pdf.

[175] Wen Tang et al. “A realistic elastic rod model for real-time simulation of minimally invasive vascular interventions”. In: *The Visual Computer* 26.9 (2010), pp. 1157–1165. DOI: 10.1007/s00371-010-0442-1.

[176] David Tarditi, Sidd Puri, and Jose Oglesby. "Accelerator: using data parallelism to program GPUs for general-purpose uses". In: *ACM SIGPLAN Notices* 41.11 (2006), pp. 325–335.

[177] Charles A Taylor and CA Figueroa. "Patient-specific modeling of cardiovascular mechanics". In: *Annual review of biomedical engineering* 11 (2009), pp. 109–134.

[178] Charles A Taylor and David A Steinman. "Image-based modeling of blood flow and vessel wall dynamics: applications, methods and future directions". In: *Annals of biomedical engineering* 38.3 (2010), pp. 1188–1203.

[179] Next Limit Technologies. *RealFlow Home - RealFlow*. URL: http://www.nextlimit.com/realflow/ (visited on 03/31/2020).

[180] Jonathan M Teich and Maureen M Wrinn. "Clinical decision support systems come of age". In: *MD computing* 17.1 (2000), pp. 43–46.

[181] The Khronos Group. *Khronos Releases Vulkan 1.0 Specification*. Feb. 2016. URL: https://www.khronos.org/news/press/khronos-releases-vulkan-1-0-specification (visited on 03/18/2019).

[182] The Khronos Group, Inc. *Core Language (GLSL)*. Sept. 26, 2017. URL: https://www.khronos.org/opengl/wiki/Core_Language_(GLSL) (visited on 04/14/2019).

[183] The Khronos Group, Inc. *Core Language (GLSL)*. Sept. 13, 2019. URL: https://www.khronos.org/opengl/wiki/Type_Qualifier_(GLSL) (visited on 04/14/2019).

[184] The Khronos Group, Inc. *OpenGL Overview*. URL: https://www.khronos.org/opengl/ (visited on 03/31/2020).

[185] The Khronos Group, Inc. *Shader data types and precision*. Aug. 19, 2019. URL: https://docs.unity3d.com/Manual/SL-DataTypesAndPrecision.html (visited on 04/14/2019).

[186] The Khronos Group, Inc. *SPIR Overview*. URL: https://www.khronos.org/spir/ (visited on 04/14/2019).

[187] The Khronos Group, Inc. *Vulkan 1.2 is Here!* URL: https://www.khronos.org/vulkan/ (visited on 03/31/2020).

[188] The Pennine Acute Hospitals NHS Trust. July 2015. URL: https://www.pat.nhs.uk/news/New-hybrid-theatre-at-The-Royal-Oldham-Hospital-to-enhance-patient-care.htm (visited on 05/03/2019).

[189] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[190] Pauline Y Tsó and Brian A Barsky. "Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping." In: *ACM Transactions on Graphics (TOG)* 6.3 (1987), pp. 191–214.

[191] Daniel Valdez-Balderas et al. "Towards accelerating smoothed particle hydrodynamics simulations for free-surface flows on multi-GPU clusters". In: *Journal of Parallel and Distributed Computing* 73.11 (2013), pp. 1483–1493.

[192] Franck Patrick Vidal et al. "Principles and Applications of Computer Graphics in Medicine". In: *Comput. Graph. Forum* 25 (2006), pp. 113–137.

[193] Pierre-Frédéric Villard et al. "Interventional radiology virtual simulator for liver biopsy". In: *International journal of computer assisted radiology and surgery* 9.2 (2014), pp. 255–267.

[194] Damien Violeau and Benedict D Rogers. "Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future". In: *Journal of Hydraulic Research* 54.1 (2016), pp. 1–26.

[195] Steven Vogel. *Life in moving fluids: the physical biology of flow*. Princeton University Press, 1996.

[196] Sarah L Waters et al. "Theoretical models for coronary vascular biomechanics: progress & challenges". In: *Progress in biophysics and molecular biology* 104.1-3 (2011), pp. 49–76.

[197] Pierre Wellner, Wendy Mackay, and Rich Gold. "Computer-augmented environments: back to the real world". In: *Communications of the ACM* 36.7 (1993), pp. 24–27.

[198] Kenric B White, David Cline, and Parris K Egbert. "Poisson disk point sets by hierarchical dart throwing". In: *2007 IEEE symposium on interactive ray tracing*. IEEE. 2007, pp. 129–132.

[199] Elizabeth Wilkins et al. *European Cardiovascular Disease Statistics 2017*. English. Belgium: European Heart Network, Feb. 2017.

[200] PT Williams and AJ Baker. "Numerical simulations of laminar flow over a 3D backward-facing step". In: *International Journal for Numerical Methods in Fluids* 24.11 (1997), pp. 1159–1183.

[201] P. A Yushkevich and G Gerig. Apr. 2018. URL: http://www.itksnap.org/pmwiki/pmwiki.php (visited on 03/18/2019).

[202] Qi Zhang, Roy Eagleson, and Terry M Peters. "Volume visualization: a technical overview with a focus on medical applications". In: *Journal of digital imaging* 24.4 (2011), pp. 640–664.

[203] zSpace, Inc. *Learning Through AR / VR Experiences*. URL: https://zspace.com/ (visited on 03/14/2019).

[204] zSpace, Inc. *Native SDK*. URL: https://developer.zspace.com/docs/native-sdk-intro (visited on 03/14/2019).

[205] zSpace, Inc. *zSpace zView*. URL: https://zspace.com/edu/info/zview (visited on 03/14/2019).

[206] zSpace, Inc. *zView - Connecting to a DSLR Camera*. Aug. 2017. URL: https://support.zspace.com/hc/en-us/articles/115003659346-zView-Connecting-to-a-DSLR-Camera (visited on 03/14/2019).

Appendix A

Demonstration Feedback from Clinician

Clinician	Professor Derek Gould
Speciality	Interventional radiology
Years of Experience	35

‘I did see the simulation and my recollection is of a nice simulation of laminar flow through a vessel and an aneurysm.’

‘I was clear about the need to simulate the real world non-laminar impacts of the changed configuration of the aneurysm. If it is partially filled with clot then laminar flow might still be possible as it could still maintain a tubular configuration through the solidified thrombus.’

‘However the natural dilated configuration of an aneurysm may contain less thrombus than this, perhaps none; this then creates a non-laminar flow pattern with back eddies and it would seem appropriate and necessary to reflect this in the simulation.’

‘We did also review some recorded data performed in the Liverpool Department of Clinical Engineering’s laboratory with Dr Then How using a vascular flow model

(rapid prototyped) that I think did show back eddies highlighted by thin beam illumination of suspended particles.'

Appendix B

Clinician Interview Notes

Clinician	Professor Derek Gould and Dr Thien Howe
Speciality	Interventional radiology
Years of Experience	35

B.1 Prepared Questions

B.1.1 Where could a fluid flow framework fit in the decision making process of interventions?

Simulating an interventional task. Mapping the flow of contrast agent, how much contrast agent to use and at what speed to inject the agent? Looking to see a faithful representation of blood flow after you alter the anatomy. To identify when risk of intervention is less than the risk of doing nothing, cerebral aneurysm analysis would also be a good example. Deploying the stent to see it restore normal laminar flow.

B.1.2 How would the decision-making process be helped by an implementation of the framework?

Mapping from CT to vascular tree and predict what a contrast injection would look like. This can be helpful with deployment of interventional methods in the vessel. The closer you can get to seeing how the anatomy looks the better for simulation purposes.

This could be also be used to determine local pressures. This would benefit cases with an aortic aneurysm to work out pressure distribution and predict the risk of a rupture. High pressure areas in complex geometry (irregular bulging) and categorise the high risk aneurysm.

Simulate flow in small aneurysms, repeat over time to see how the system evolves. Where is pressure dangerously high? Today's decision guidelines are purely on the size of the aneurysm, a threshold primarily based on statistics.

B.1.3 What is an example of the sequence of events in diagnosis?

Pre-operative planning. Risk assessment

Inter operative decision making provided data from patient can be acquired and then fed into the real time simulation. E.g. Duplex can give info on properties of plaque. Also inlet velocity. Reduced velocity from embolic. Other complications.

B.1.4 What kind of interactions with flow would be helpful in decision making?

Changing the pressure. Specifying flow rates. In and out flow. Using additional data from other sources, ultrasound for example. Monitoring peripheral emboli - side effects resulting from the procedure that you are carrying out.

B.2 Additional Feedback

Ability to calculate shear forces then relate shear stress maps with known areas where atherosclerosis develops. Calculate where the area of low shear and at risk of developing stenosis. Then using this information to develop preemptive treatment.

Flow velocity is generally low - how do you define this in simulator? Liverpool do have data from previous experiments which were used in finite element simulations.

We see the velocity increase in the other side of the bifurcation when a blockage is added, because of the complex vascular network with all of the other branches possible, it might be the case that the flow does not change that much. It would also be useful to be able to slow down the particle flow and specify the inlet conditions for investigating some clinical scenarios.

Appendix C

Example Application Configuration JSON File

```
{
  "boundary layers": 3,
  "cell size": 1,
  "gravity": {
    "x": 0.0,
    "y": -1.0,
    "z": 0.0
  },
  "initial spacing": 0.08,
  "kernel radius": 1,
  "maximum particle neighbours": 100,
  "maximum particles": 60000,
  "rest density": 1000,
  "sim scale": 1,
  "stiffness": 250,
  "time step": 0.005,
  "viscosity": 18
}
```

- Boundary layers: the number of particle layers generated when preprocessing the loaded geometry.
- Cell size: The size of each cell in the spatial uniform grid. The grid cells are equal in each dimension.
- Gravity: gravity force / external directing force to be universally applied to the fluid.
- Initial spacing: the distance between the particles when they are seeded into the geometry.
- Kernel radius: the SPH smoothing kernel radius length.
- Maximum particle neighbours: Maximum number of boundary particles to take into consideration on each time. Only available if the simulation is compiled with the maximum particle neighbours macro.
- Maximum particles: The maximum amount of particles to be seeded into the simulation
- Rest density: the resting density of the fluid used in the pressure state equation
- Sim scale: the simulation takes place in some arbitrary geometric structures, this property scales all interactions within the simulation accordingly.
- Stiffness: the stiffness coefficient used in the pressure state equation
- Time step: the amount of simulation time to integrate on each step.
- Viscosity: the viscosity coefficient used in the force calculation step.