Towards Real-Time Hydrodynamics and Fluid Simulation for Marine Robotics

By:
Yosef Shmuel Guevara Salamanca

Supervisors:
Profesor Rodrigo Ventura. Institute for Systems and Robotics (ISR-Lisbon)
Giang Nguyen. Institute for Artificial Intelligence, University of Bremen.

Thesis proposal to obtain the Master in Engineering Degree in:
Mestrado Bolonha em Engenharia Electrotécnica e de Computadores

Instituto Superior Técnico and Universität Bremen
Erasmus Mundus Joint Master's degree in:
Marine and Maritime Intelligent Robotics Master
2023/2024

Statement

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Abstract

This thesis proposal explores the development of an advanced simulation tool for marine robotics, focusing on real-time hydrodynamics and fluid simulation. The main goal is to create an open-source asset for game engines, enabling the simulation of open oceans, shallow waters, and underwater scenarios in marine simulators. The study employs various fluid simulation techniques, including finite differences for open seas, smoothed particle hydrodynamics for shallow waters, and geometry-based methods for underwater simulations. The research addresses the challenge of balancing realistic physics with hardware limitations, offering solutions for non-visual water behaviors and small-scale phenomena like splashes and bubbles. The thesis contributes to the field by providing a unique, customizable, and accessible tool for simulating marine environments, integrating advancements in robotics, computing, and rendering technologies.

Key Words: Marine Robotics, Real-Time Hydrodynamics, Fluid Simulation, Open-Source asset, Marine Environments, Smoothed Particle Hydrodynamics, Underwater Simulation, Computational Fluid Dynamics, Physics-based Modeling, Robotics and Simulation Technology

Resumo

Esta tese explora o desenvolvimento de uma ferramenta de simulação avançada para robótica marinha, focando em hidrodinâmica e simulação de fluidos em tempo real. O principal objetivo é criar um recurso de código aberto para motores de jogos, possibilitando a simulação de oceanos abertos, águas rasas e cenários subaquáticos em simuladores marinhos. O estudo emprega várias técnicas de simulação de fluidos, incluindo diferenças finitas para mares abertos, hidrodinâmica de partículas suavizadas para águas rasas e métodos baseados em geometria para simulações subaquáticas. A pesquisa aborda o desafio de equilibrar física realista com limitações de hardware, oferecendo soluções para comportamentos de água não visuais e fenómenos em pequena escala como respingos e bolhas. A tese contribui para o campo ao fornecer uma ferramenta única, personalizável e acessível para simulação de ambientes marinhos, integrando avanços em robótica, computação e tecnologias de renderização.

Palavras-chavem: Robótica Marinha, Hidrodinâmica em Tempo Real, Simulação de Fluidos, Recurso de Código Aberto, Ambientes Marinhos, Hidrodinâmica de Partículas Suavizadas, Simulação Subaquática, Dinâmica de Fluidos Computacional, Modelaçem Baseada em Física, Tecnologia de Robótica e Simulação.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In the field of engineering and research, the importance of simulators is indisputable [1]. These virtual tools are capable of replicating real-world situations, offering functionalities such as modeling of physical systems, sensor integration, as well as the ability to program and interact with various objects, among others. However, in the field of marine and maritime robotics, there is a noticeable lack of advanced simulators [2] that implement real-time physics and fluid interaction, which are critical aspects in areas such as control algorithms and artificial intelligence. The few available options often rely on proprietary software, which poses challenges in terms of acquisition, development, and maintenance, and often requires specific hardware, limiting accessibility.

Understanding the interaction of marine vehicles with water is a key aspect of marine engineering, as the various effects of forces and moments on these vehicles are fundamental to their design [3]. The study of how water interacts with the hull can drive the development of more efficient marine vehicles, with improvements in mobility, energy consumption, cost-effectiveness, and adaptability to different weather conditions. For example, high resistance during navigation increases energy consumption and reduces autonomy, which is undesirable for designers [4].

Simulating extensive bodies of water such as seas, lakes, and even pools is crucial, as their presence is common in various virtual and real-world environments. Achieving convincing realism without proper water simulation is complex [5]. Often, simplified or unrealistic techniques are used due to the high cost of full 3D fluid simulations [6, 7]. In addition, accurate simulation demands high computational power and faces challenges in effective integration with visualization tools, as well as model validation and verification [8, 6].

As for real-time simulation, it is imperative to stay within the established processing time limits of 15 to 60 frames per second (FPS), adjusting resolution and visual quality as needed [9]. This implies that methods used in non-real-time simulations are not directly transferable to real-time applications. a distinction is made between simulators focused on visual representation, such as game engines that prioritize graphical fidelity, and those where physical accuracy is essential [6].

Real-time water simulations must be capable of handling a wide range of user interactions with predefined simulation parameters. Although full 3D fluid simulations are powerful, their high computational demand makes them prohibitive for real-time applications, especially for simulating large bodies of water. This thesis focuses on the development of an asset for a game engine to simulate open oceans, shallow waters, and underwater vehicles to be integrated into marine simulators, using methods that replicate realistic physics and visualize marine phenomena. It combines fluid simulation techniques such as finite differences for open seas, smoothed particle hydrodynamics (SPH) for shallow waters, and geometry-based methods for underwater simulations.

The thesis also addresses hardware limitations and the necessary simplifications in these simulations, considering the removal of non-visual water behaviors and the use of additional formulations to emulate small-scale phenomena such as splashes and bubbles. The stability of real-time simulations versus offline simulations is analyzed, which can be adjusted until satisfactory results are obtained.

Finally, the focus on creating an open-source asset provides a unique opportunity to advance in this field, overcoming existing limitations, promoting collaboration, customization, and enabling interdisciplinary integration between robotics, computing, physical simulation, and rendering capabilities. This leads to the creation of visually stunning and accurate marine environments.

CHAPTER 2

State of the art

Contents

Ocean and hydrodynamic variable simulation plays a crucial role in various applications, from the maritime industry to scientific research. In many robotics scenarios, the choice between real-time simulations, suitable for applications demanding immediate and interactive responses, and offline simulations, preferable when seeking greater accuracy, comprehensive analysis, and modeling of complex events, is a key consideration Jeschke et al. in 2018 [10]. To address these simulation needs comprehensively, we will now delve into four specific approaches used to simulate oceans, hydrodynamic variables, and their interaction with objects.

Therefore, we present four distinct approaches used in the simulation of oceans, hydrodynamic variables, and their interaction with objects. The first section focuses on simulating the ocean surface in open sea section (2.1), primarily using finite-difference-based methods, and explores how the interaction of waves with objects in the sea and their rendering process is simulated. The second section (2.2) addresses geometric methods for approximating the hydrodynamic variables of fully submerged objects and the equations of motion of a fully submerged vehicle. The third section (2.3) centers on the use of Computational Fluid Dynamics (CFD) tools to simulate variables such as pressure and drag in the direction of fluid flow using numerical mesh methods. Finally, the last section explores the simulation of shallow waters using Smoothed Particle Hydrodynamics (SPH) methods section (2.4) for submerged vehicles and hybrid methods that combine finite-difference-based approximations (2.4.1).

2.1 Approaches for simulating surface waves.

In real-time two-dimensional water wave simulation, developers must choose between utilizing efficient Fourier-based methods that lack interactions with moving obstacles or opting for more expensive finite-difference or finite-element methods capable of accommodating environmental interactions [10].

The methods used to simulate surface waves primarily focus on two approaches: one involves discretizing wave height and momentum at each point on a grid, as is shown in Figure 2.1, known as finite-difference methods presented by Tessendorf in 2004 and 2017 [11, 5] or spectrum-based method; while the other approach entails discretizing wave amplitudes as a function of frequency and direction, referred to as Fourier-based methods [12], However, since these methods are derived based on several assumptions about the underlying flow, they typically have limitations, such as their inability to realistically interact with complex boundaries. On the other hand, modern approaches, such as wavelet transformation, which discretizes wave amplitudes considering not only space but also frequency and direction in combination [10]

Spectrum-based methods use analytical solutions to the Navier-Stokes equations on a 2D height field to represent ocean motion as sine and cosine functions, sacrificing the ability to simulate arbitrary fluid motion [5, 11]. To handle complex boundaries, it is necessary to extend these methods by simulating wavefront propagation within a static environment [10].

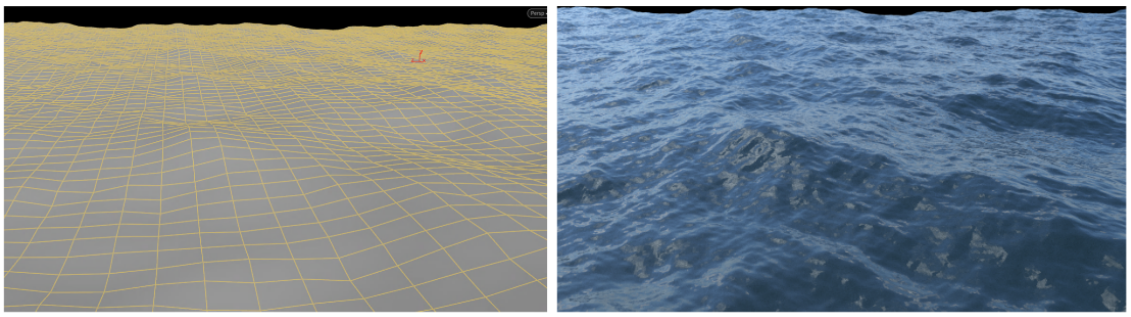


Figure 2.1: Visualization of the ocean surface (left) and the corresponding rendered image (right)[13].

As shown in this chapter, simulating the complexities of natural ocean phenomena and achieving accurate representation is a challenging task. The ocean's highly dynamic behavior includes various complexities, ranging from calm seas to turbulent oceans, and from small ripples to large shore breaks. These dynamic ocean movements are influenced by various phenomena occurring at different scales, present in 2011 by Darles et al. [14]. Therefore, a typical rendering process workflow is described in the as follows.

Ocean Waves typical pipeline

Some common approaches use eWave interaction to simulate the iWave problem [11]. eWave employs a modified propagation algorithm based on FFTs and supports object interactions [5]. It utilizes a 2D obstruction map to represent interactions, aiding in source identification for surface disturbances. The eWave layer can be combined with spectral ocean surfaces, creating a unified displaced ocean.

The figure 2.2 below illustrates the rendering workflow using Gilligan-Maya to construct the ocean environment in Bait, producing essential simulation data such as the ocean surface mesh and displacement map textures. Meanwhile, the Gilligan-Houdini workflow enhances ocean simulation by incorporating non-planar geometry binding and automated floating animations for secondary objects.

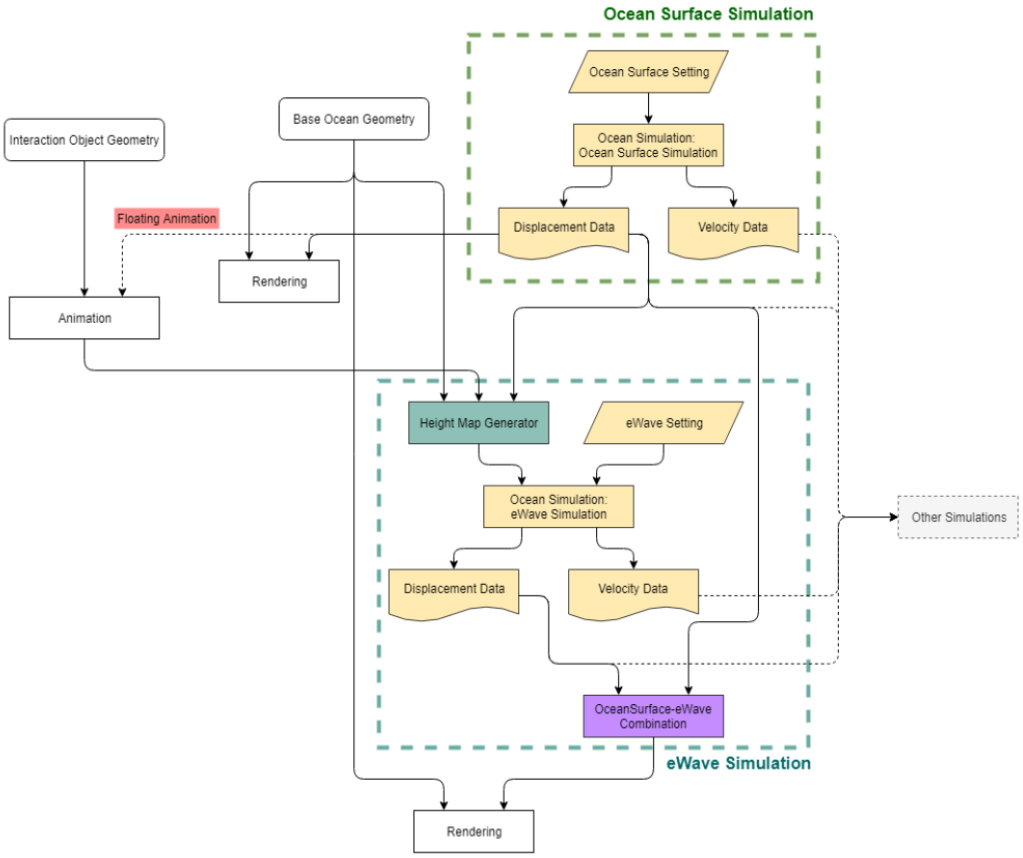


Figure 2.2: Gilligan-Houdini workflow.[13]

When it comes to simulating ocean interaction with characters or objects, a common practice is to create a distinct simulation near the characters, as is shown in Figure 2.3, a custom blend node is employed to merge separate particle simulations with ocean height fields like the presented by Moana Technical team in 2017[15]. On the other hand, FLIP tanks, enable the simulation of the entire ocean surface by initializing it from existing ocean surfaces or the ones presented by Hopper et al. in 2003 [16].

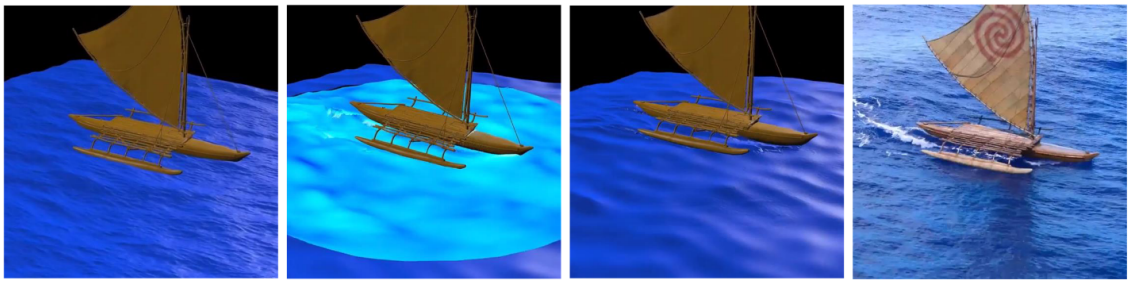


Figure 2.3: Ocean Interaction in Moana (2016), Disney Animation Studios[17].

2.1.1 Object interactions with water and hydrostatic forces

In general, in real-time graphical environments, rigid bodies are used to simulate vehicles or objects above or below the water surface, and as fixed bodies, such as land masses or icebergs. Rigid bodies in graphic environments and their interaction with water, typically mesh-based objects, requires additional mathematical methods to simulate forces like buoyancy and ocean currents presented by Mendonça et al. in 2013 [18].

The Buoyant force

The buoyant force $\mathbf{F}_b$ occurs due to the surrounding water and acts in an upward direction. Its magnitude equals the weight of the water displaced by the submerged hull. In mathematical terms, the magnitude of the buoyant force vector $\mathbf{F}_b$ is expressed as:

$$\|\mathbf{F}_b\| = \rho \cdot v_f \cdot g, \tag{2.1}$$

Here, g represents the gravitational constant, ρ denotes the density of water, and v represents the displaced water volume, and $\mathbf{F}_g$ have equal magnitudes but opposite directions.

Wind and Water Currents

The movement and orientation of a rigid object on the water's surface are influenced by propeller activation and external forces, such as wind, water currents, and waves. Given the absence of sails in most modern marine vehicles, the effects of wind can be neglected, except when considering their impact on the vehicle's hull. Lateral winds cause oscillating roll torque, whereas frontal winds result in oscillating pitch torque [18].

La wave amplitude as a function of time t and wind amplitude ν using the Beaufort scale. This approach simplifies complex wave interactions. Pitch and roll angles of the surface vehicle, ϕ_{pitch} and ϕ_{roll}, are modified based on their alignment with the wind directional vector $\mathbf{p}_w$ and user-defined factors α_{pitch} and α_{roll} [18].

$$\begin{aligned}\phi_{\text{pitch}}\left(\mathbf{p}_m^y, \mathbf{p}_w, \nu, t\right) &= \left(\mathbf{p}_m^y \cdot \mathbf{p}_w\right) h(\nu, t)\alpha_{\text{pitch}};\\ \phi_{\text{roll}}\left(\mathbf{p}_m^x, \mathbf{p}_w, \nu, t\right) &= \left(\mathbf{p}_m^x \cdot \mathbf{p}_w\right) h(\nu, t)\alpha_{\text{roll}};\end{aligned}$$

Other methods allow the simulation of arbitrarily shaped 3D models using Proportional-Integral-Derivative controllers for the ship's motion, by introducing non-hydrostatic forces, one example of a hydrostatic force algorithm is the introduced in Zheleznyakova et al. in 2020[19], which results with in different wind conditions, can be seen in the Figure 2.4, and a rigid body motion algorithm described by House et al. in 2016 [20]. However, it cannot accurately capture non-linear effects or complex phenomena like breaking waves or splashes [21].

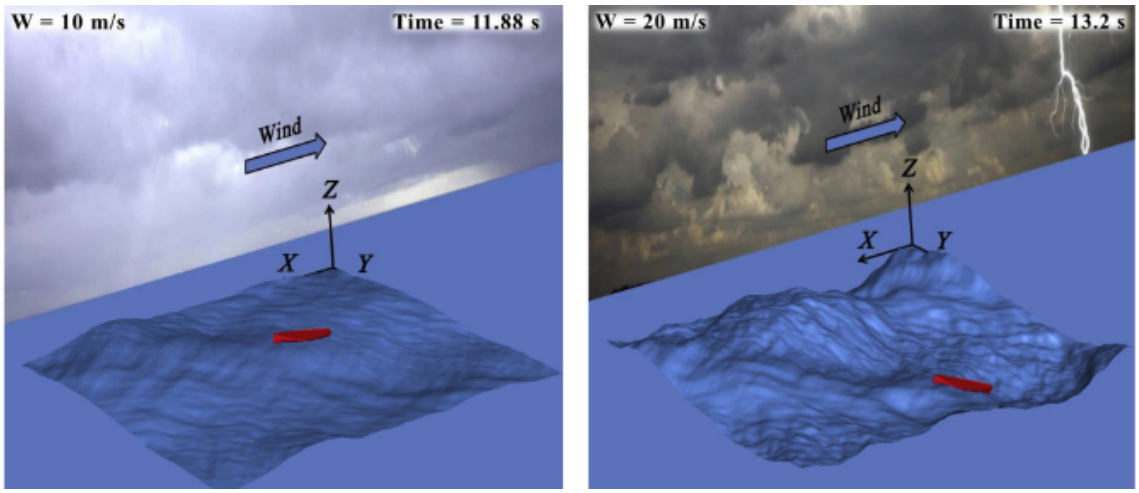


Figure 2.4: A. Zheleznyakova algorithm approach results under different wind conditions[19].

2.2 Geometrical-Based Methods and Fossen Equations

Geometric methods encompass various approaches, ranging from assuming the shape generated by the object to applying the slender body theory, which treats objects as two-dimensional surfaces as the ones presented by Severholt et al. in 2017 [22]. These methods are often combined to analyze the behavior of objects in motion within a fluid, and their significance lies in their ability to simplify and enhance our understanding of fluid dynamics and hydrodynamics in various applications, without any excessive computational requirements.

These methods help to understand the forces applied to the vehicle by the surrounding fluid medium can be roughly broken down into added mass, lift, drag, and hydrostatic forces as Jakuba et al. explained in 2003 [3]. For example, some hydrodynamic characteristics of vehicles are usually described using the Myring profile equations, which aim to reduce the resistance exerted by the fluid on the general shape of the vehicle like the presented in Le et al. in 2021 [23].

The shape of the vehicle affects how water flows around it. In the case of underwater vehicles, when the vehicle accelerates, it displaces a certain amount of water. This displacement leads to what is known as added mass, which causes the vehicle to exert additional force to move the fluid in the direction of the vehicle's movement, as is explained in by Zeraatga et al. in 2019 [24]. Simultaneously, the resistance of the fluid to the vehicle's motion, referred to as hydrodynamic resistance or drag, is determined by factors such as pressure distribution and flow separation along the vehicle's hull as is explained by Tober et al. 2020 [25].

The energy required to move the fluid in the direction of the vehicle's displacement acts as a pressure on the hull, and given that pressure operates normal to the surface, the geometry of the body becomes crucial in this context, as Perrault et al. discuss in 2002 [26].

In general, an Underwater Vehicle can be divided into three main components[3], The hull, control planes, propeller. One widely employed method in naval architecture is the slender body theory, which assumes a slender body shape and offers the advantage of working with a more accurate body shape without excessive simplification, as Newman et al. proposed in 1977 [27]. These calculations have the advantages of low computational power requirements, enabling quick simulations, and having a general formulation applicable to arbitrary hull and control surface shapes, which can be verified against experimental data.

In calculating rigid body dynamics for a 6 DOF underwater vehicle, we use the movement equation proposed by Fossen's book in 2011 [28]:

$$M\dot{v} + C(v)v + g(\eta) + g_o = \tau \tag{2.2}$$

Where:

M : Inertia Matrix (including added mass)
$C(v)$: Matrix of Coriolis Centripetal term (Including added mass)
$D(v)$: Damping matrix
$g(\eta)$: Vector of gravitational forces and moments
τ : Vector of control inputs

However, this model simplifies the following assumptions and limitations [22]:

1. Deep submersion of the body, ensuring constant added mass coefficients.
2. Neglects waves, currents, and disturbances.
3. Assumes an inviscid, frictionless fluid with no circulation, applying potential flow theory.
4. Vehicle's rotational symmetry around the x-axis with identical cross sections in the XY and XZ planes.
5. Treats vehicle components in isolation, ignoring interactions.

This approach offers advantages such as analytical coefficient determination without excessive computational complexity. It is not limited to a small region around the vehicle's equilibrium, as some methods are, and it leverages the local symmetry of individual components, even if the entire vehicle may not be symmetric, including the control planes. In the context of determining added mass and inertia, experimental methods such as the planar motion mechanism (PMM) test, rotating arm test, circular motion test (CMT), and variations are often employed, as Seong-Keon Lee et al. explain in 2011 [29].

2.3 Computational fluid dynamics (CFD) Method.

Computational Fluid Dynamics (CFD) is a field within fluid mechanics that uses numerical analysis and algorithms to solve and analyze problems involving fluid flows. CFD works by numerically solving the equations that govern fluid flow, primarily the Navier-Stokes equations. These equations describe how fluids move based on various parameters such as pressure, velocity, and viscosity. In practice, CFD involves dividing the space through which the fluid moves into a complex network, known as a mesh, consisting of countless small volumes or elements. This section focuses on the application of Computational Fluid Dynamics (CFD) within the context of marine robotics. We will explore how CFD is employed to model and analyze fluid behavior in marine environments.

For studying fluid mechanics, particularly in microchannels and sterilization chambers [23]. In the field of marine robotics, it has been applied to calculate hydro-drag and lift coefficients for underwater gliders and unmanned underwater vehicles, revealing their dependency on steady-state velocity and enabling performance predictions under varying conditions [23] , as is shown in Figure 2.5.

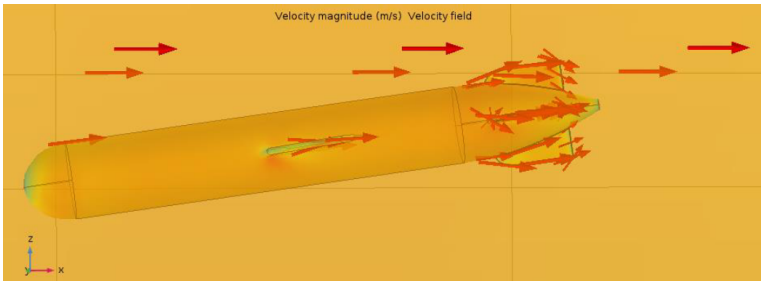


Figure 2.5: The velocity fields in a glider velocity $U = 1m/s$ and angle of attack $\theta = -10°$ [4].

CFD allows investigating the hydrodynamic characteristics for underwater vehicles. For example, the influence in the angle of attack in influencing drag and lift coefficients. The results provide valuable insights for improving underwater vehicle design and are applicable to real-world scenarios, as is explained in Hang Su et al. in 2018 [30].To calculate the interaction of the water, it is necessary to define a continuous domain around the studied body, subdivide it into small control volumes, forming a discrete domain known as the numerical mesh [4], as is shown in Figure 2.6.

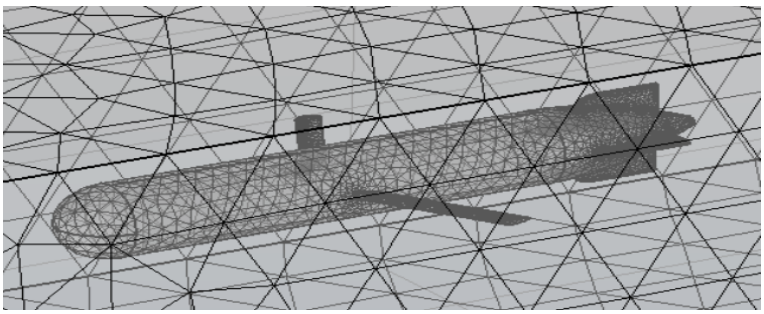


Figure 2.6: The dense mesh around the torpedo surface used for computation [4].

For example, the CFD analysis of underwater vehicles, particularly an AUV without rudders, shows minor variations in the predicted and experimental volumetric drag coefficients, with maximum and average differences of 8.1% and 7.2% , respectively. These discrepancies can be ascribed to a combination of experimental and numerical errors.[4]

With the development of more and more advanced computers and computer models, methods such as CFD and boundary element methods are getting increasingly more common. Although these methods are very accurate, they require more computational power and are quite complicated. In an extensive simulation program, it is of importance to use simpler models to make the model less demanding and require less computational power, especially when the shape of the hull is as simple as AUV hulls often are.

2.4 Smoothed Particle Hydrodynamics Methods

Smoothed Particle Hydrodynamics (SPH) is a computational method developed in the 1970s, initially for astrophysics, that simulates the dynamics of fluids and solids. Using a Lagrangian approach, it tracks the movement of individual particles in a fluid. In marine robotics, SPH has been used to model how fluids interact with marine vehicles, allowing for realistic simulations of their movements in aquatic environments.

For SPH simulations with about 10,000 particles can run in near real-time. However, for larger applications requiring mathematical precision, up to 1,000,000 particles may be needed, posing real-time simulation challenges as is mentioned in Angelidis et al. 2022 [31], where an accuracy level of 30% to 40% is deemed satisfactory, as long as the simulation process remains reasonably efficient as is explained in Gartner et al. in 2023[32]. Recent improvements in SPH models have made them a competitive alternative to other fluid simulation methods, overcoming challenges such as neighborhood search algorithms and pressure solvers as is exaplained by Koschier et al. in 2022[7].

In marine robotics, incompressible SPH (ISPH) has proven effective in simulating underwater vehicle motion, including buoyancy and speed limits, aligning with real-world physics achieving to replicate hydrodynamic variables as the added mass and the drag [32] as is shown in Figure 2.7.

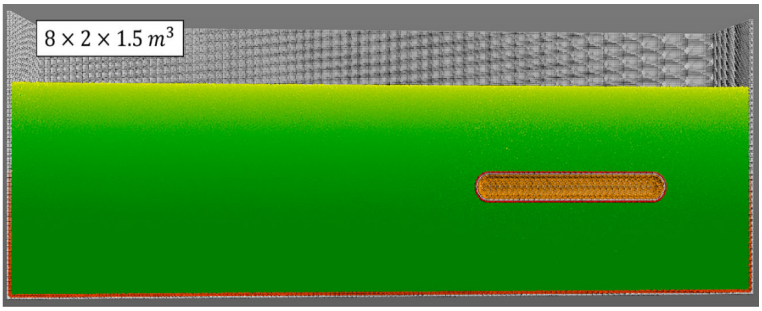


Figure 2.7: Capsule shape AUV inside the tank of particles [32].

On the other hand, as is shown in Angelidis et al.[31] Gazebo robotics integrates the SPlisHSPlasH SPH method to simulate fluid interactions with rigid bodies. This enables synchronization between fluid and rigid body simulations, with applications like dam break simulations and autonomous swimming, opening possibilities in soft robotics and swimming robots. as is shown in Figure 2.8.

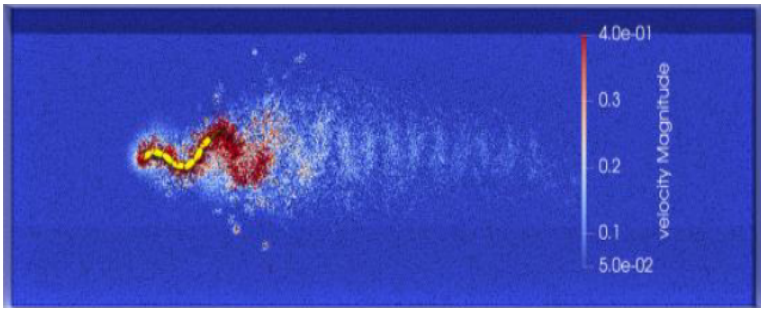


Figure 2.8: Simulation of an amphibot robot in a particle tank [31].

The SPH methods are used to simulate the interaction of objects both below and above the surface of a fluid and to calculate hydrodynamic variables accurately. However, due to the significant computational requirements needed to simulate a large number of particles, their applications are limited by processing time, preventing real-time results.

For this reason, other particle-based approaches have chosen to employ hybrid approaches that combine methods based on finite differences, as described in section 2.1 to generate the ocean surface, with the SPH methods for object iteration or small details like foam or splashes. These hybrid wave and particle approaches will be explained in detail on the next page.

2.4.1 Hybrid Wave and particle Approaches

To achieve real-time simulation of large volumes of water, Jensen and Goiás in 2001 [12] employed grid-based Eulerian methods to model deep ocean waves and their interaction with objects. However, their method involves numerical calculations and artificial damping. Hagen et al. [33] achieved significant acceleration by simulating non-linear shallow-water waves on the GPU, achieving up to a 30-fold speedup compared to the CPU. Kim et al. [34] used the GPU to calculate buoyant forces on arbitrary models, enabling real-time interactions between the fluid and objects, achieving 16 frames per second with 50 floating objects.

Harris in 2004 [35] demonstrated how to implement a 2D simulation of the Navier-Stokes equations for incompressible flow on the GPU, allowing real-time simulations. Crane et al. [36] presented a complete 3D simulation of water on the GPU, achieving high frame rates (120-180 FPS) at a grid resolution of 64x64x128. Cords in 2007 [37] proposed an approach that separates water surface simulation into two components, using a 3D simulation for low-frequency fluid flow and a 2D simulation of high-frequency surface waves. Clavet et al. in 2005 [38] implemented a Lagrangian simulation of viscoelastic fluids, achieving interactive frame rates using particles. Treuille et al. in 2006 [39] introduced a model reduction method that precomputes multiple fluid simulations and projects them into a low-dimensional solution space, enabling high-resolution real-time simulations, although with a long precomputation time. In the FLIP method proposed by Brackbill et al. in1986 [40], the fluid is represented using Lagrangian fluid elements known as particles, which carry information about mass, momentum, internal energy, and constitutive properties and move through a grid.

Some particle wave methods are better to simulate large bodies of water, such as oceans and lakes, with a focus on small-scale like foam and splashes details for enhanced realism, as is shown in Figure 2.9. It employs a specialized shallow water solver, tailored to variations in topography and water depth, enabling the simulation of phenomena like currents, breaking waves, and waterfalls. This method transforms hard-to-represent elements like foam and spray into particles and is implemented using CUDA for optimal performance on modern GPUs. It is ideally suited for the entertainment industry, such as in video games and environmental visualizations, where detailed and varied representations of bodies of water are required as Chentanez and Müller achieved in 2010 [8].

On the other hand, the Wave Particles method focuses on using wave particles to simulate the dynamics of water surface waves and their interaction with floating objects. This approach creates a simplified method for representing the movement and deformation of waves, transforming wave particles into a water surface height field that simulates the flow induced by waves, as is shown in Figure 2.10. Additionally, it allows for easy integration of interactions between water and floating objects, and utilizes advanced graphic technology for real-time simulation and rendering. It is particularly suitable for games and virtual reality applications that require realistic simulations of water-object interactions in a marine environment, as was achieved in 2010 by Cem Yuksel [6].

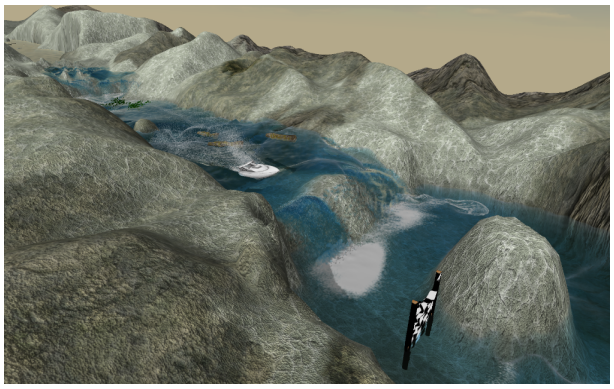


Figure 2.9: Small waterfalls, swirling vortices, foam, spray, splashes, small-scale waves, interaction of water with rigid and soft bodies.[8].

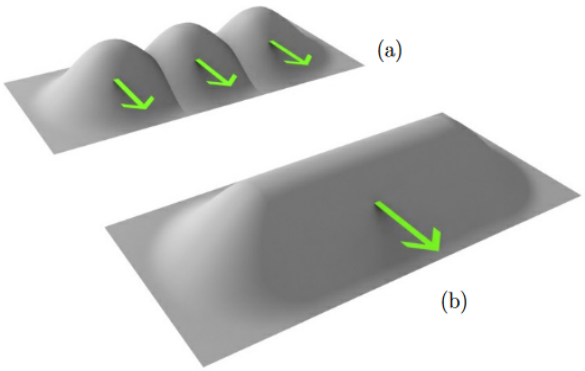


Figure 2.10: (a) Individual wave particles (b) Wavefront formed by these wave particles [6].

CHAPTER 3

Methodology

Contents

As previously addressed, the main objective of this master is thesis is to develop an asset that enables real-time simulation of large volumes of water, both in shallow waters and open seas, for use in surface and underwater marine vehicle simulators. The simulation will be designed to seamlessly integrate with the **Multiverse Simulation Framework**[1], which combines various advanced physics engines with photorealistic rendering capabilities. Therefore, the first section will focus on selecting the appropriate game engine upon which the asset will be built, determining which of the available engines is most suitable for simulation creation.The second section (3.1) will cover the asset creation using Blender, which is a freely available 3D modeling tool that allows for direct model exports to various game engines. The third section (3.3) will address the application of the algorithm for simulating open seas, shallow waters, and underwater scenarios. Finally, the fourth section (3.4) will encompass the integration with the Multiverse Framework.

3.1 Choosing the Right Game Engine

The first part of the analysis focuses on tests related to the game engine and consists of three main parts:

1. Possibility of integration with physics engines such as Mujoco, Nvidia Physics, pybullet, Chrono, Dojo, Dualphysics, among others.
2. Capability and ease of simulating fluids.
3. Compatibility with ROS integration.
4. Compatibility with platforms such as Gazebo, CopeliaSim, Webots, and Isaac.

Additionally, the following aspects will be evaluated:

1. Ability to achieve hyper-realistic graphic quality.
2. Capability to run on multiple operating systems.
3. Availability of using external resources.

To accomplish this, tests will be conducted using three popular game engines: Godot, Unity, and Unreal Engine, which logos are shown in the Figure 3.1, to help the reader to recognize them. These three game engines will be explained in more detail in the following subsections.

Figure 3.1: Game Engines Logos.

[1] https://github.com/Multiverse-Framework

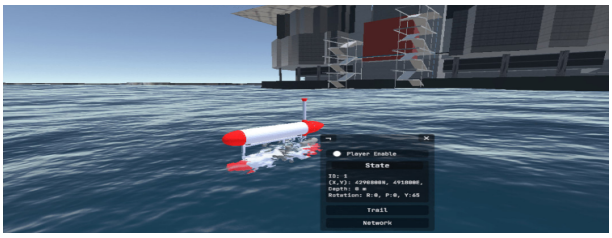
3.1.1 Godot

Godot is a versatile game engine that is well-suited for both 2D and 3D game development. It provides cross-platform support, ensuring compatibility with multiple operating systems, such as macOS, Windows, and Android. One of its remarkable attributes is being a Linux-native engine, operating under the permissive MIT license. However, despite its integration with ROS, its adoption within the scientific community remains relatively small. UAn example of simulating an ocean surface shader can be seen in Figure 3.2.

Figure 3.2: Realistic Water shader from UnionBytes [41]

3.1.2 Unity

A large number of marine simulators have been developed using Unity [42, 43, 44], some examaple of marine robotics simulators made in this engine are shown in the Figure This choice is due to the significant advantages that Unity offers in terms of versatility, as it is fully integrated with various physics engines and is compatible with multiple platforms. In addition to its use in the mentioned fields, Unity has also proven its utility in other robotics applications, including medical simulators and terrestrial robotics simulators [2].



(a) Vehicle mred executing station keeping, NetMarSyS Simulator [43]

(b) Example of visual fidelity achievable, MARUS Simulator[42]

Figure 3.3: Simulators made in Unity.

3.1.3 Unreal engine

Currently, it is the game engine with the best visuals, capable of easily generating large scenarios, which is one of its standout features. It has been used as a visualization layer for some simulators like CARLA [45], a popular tool for autonomous vehicles, and Nvidia Isaac, a tool for simulating synthetic data in physically realistic virtual environments [2]. It has also been employed in some marine simulators for underwater robotics environments, such as HoloOcean [46] and UNavSim [47].

(a) UNav-Sim Simulator [47].

(b) HoloOcean Simulator [46].

Figure 3.4: Simulators made in Unreal.

3.2 Creating Simulation Assets

To address the need for creating assets that allow us to assess the results of our simulation and verify the interaction of rigid bodies with water, we have chosen to use Blender instead a Computer-Aided Design (CAD). Blender is an open-source and freely available 3D graphics software that provides various essential tools and features for this purpose, which will be explained further.

The following figures demonstrate how Rhinoceros software [48] generates various types of meshes for abstract shapes. However, when using these meshes in other applications, it often becomes necessary to export them and make additional adjustments to ensure optimal performance, which can complicate the process, even when working with simple shapes like cubes.

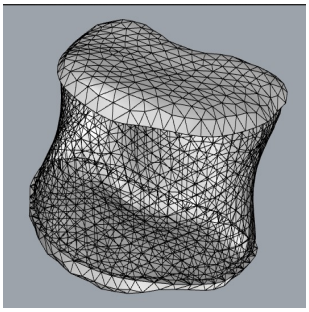


Figure 3.5: Messy mesh.

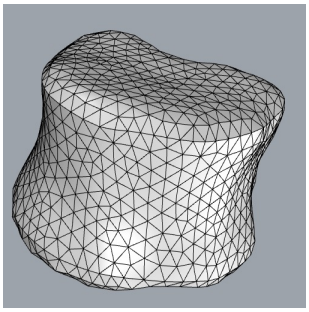


Figure 3.6: Tri-mesh Figure.

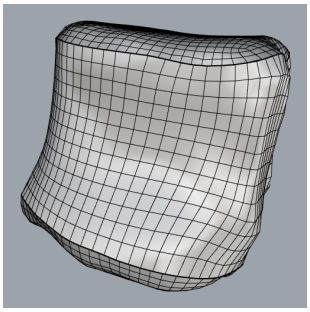


Figure 3.7: Quad mesh Figure.

Firstly, Blender[49] allows us to optimize the number of polygons generated for each vehicle. This means that we can ensure that simple shapes have an appropriate number of polygons, preventing excessive computational resource usage during the simulation.

Secondly, Blender simplifies this task by offering an export format designed to maintain clean and efficient meshes, composed of tri-mesh (3-vertex polygons) as shown in Figure 3.6 or quad-mesh (4-vertex polygons) as shown in Figure 3.7, as illustrated in Figure. This feature ensures that the model retains its optimal shape upon export, without requiring additional steps. In contrast, when exporting from other platforms such as SolidWorks, Autodesk Inventor, or Rhinoceros, as shown in Figure 3.5, additional adjustments are often needed, which can compromise the quality of the exported model.

3.3 Algorithm Implementation for Simulation

3.3.1 Algorithm for Open Sea

To model waves in a virtual open sea environment, we will employ finite difference methods, following the proposal by Jensen [12] in his work on animation and rendering of deep waters. This approach aligns with the principles established in Tessendorf's work [5], which also addresses the impact of fluid on objects in a simulated environment.

Shader Implementation

To achieve an accurate representation of the open sea, we will develop a specific shader. This shader must be efficient enough to render extensive ocean scenes without the need for loading screens; however, this approach may limit the overall size of the available area initially. Shader efficiency is achieved through a well-structured mesh that is divided both horizontally and vertically, with sufficient resolution to affect different areas of objects located on the surface.

Finite Difference Methods and Wave Dynamics

We will implement finite difference methods to simulate wave dynamics. This will allow us to generate waves with realistic characteristics, including crests and troughs, as seen in [12, 5].

3.3.2 Shallow Water Algorithm

The algorithm to be implemented is a combination of mesh-based and particle-based methods [8]. It is capable of handling terrains with varying slopes and water depths. This solver is essential for simulating both shallow water scenes and transforming particles into liquid, which cannot be represented using a height field, into spray, splashes, and foam when interacting with different objects.

Technical Implementation

The implementation is carried out using CUDA or GPU-supported parallelism, allowing real-time performance on modern GPUs. Shallow water equations are employed for fluid simulation, involving the integration of velocities and heights, as well as specific algorithms for handling turbulence and fluid-object interactions.

Small-scale Details Algorithm

The algorithm generates and simulates various types of particles [8], such as spray, splashes, and foam, to add realistic details to the simulation as mentioned earlier. These particles are used to represent aspects of water that cannot be captured with the mesh-based approach seen in Jensen [12].

3.3.3 Underwater Algorithm

Visualizing the interaction between underwater vehicles and the surrounding fluid is nearly impossible underwater. However, all underwater vehicles are affected by factors such as pressure, drag, and added mass, as indicated by Jakuba [3]. One way to model this interaction would be through the SPH method, as discussed previously in section 2.4. However, as pointed out by Gartner et al. [32], simulation results significantly depend on the quantity of particles (density) in a given area and their size. These simulations are far from real-time [32] and may differ from the expected theoretical results.

Therefore, the goal is to develop an approach using colliders offered by game engines, which provide well-known geometric shapes, to easily estimate approximate values of added mass and drag, as observed in [22] using geometric methods.

For this part, geometric calculations for the Sparus II AUV [50] will be used as a reference, as it shares the typical torpedo-shaped form common to many underwater vehicles. The technical characteristics of this vehicle can be seen in figure 3.8.

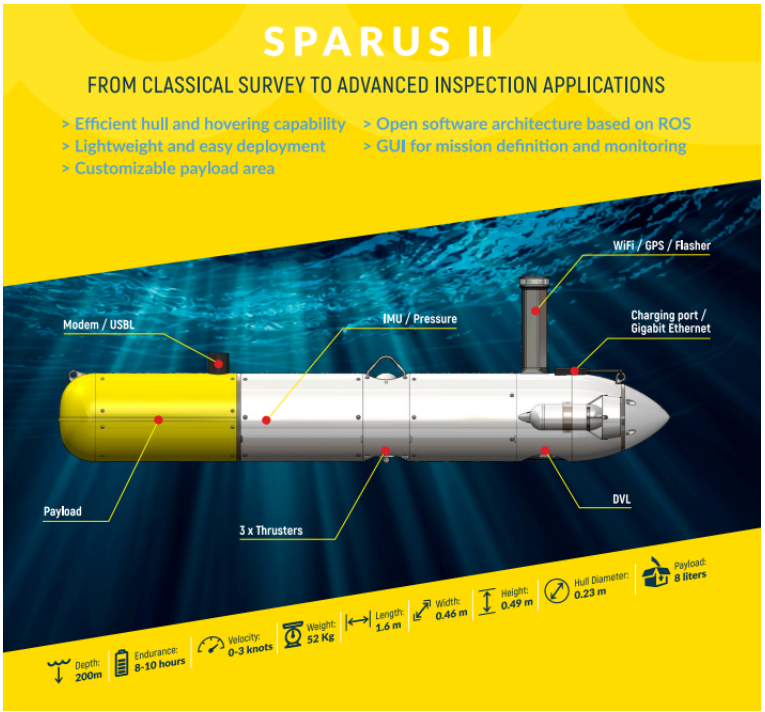


Figure 3.8: Sparus II technical overview [50].

Mass matrix computation

The mass matrix represents the distribution of mass in a rigid body. The diagonal elements of the matrix represent the mass of the body in each degree of freedom, while the off-diagonal elements capture the cross-coupling effects between the degrees of freedom, such as the interaction between translation and rotation. This matrix plays a fundamental role in the equations that define the movement of the vehicle, allowing us to calculate the accelerations of a given system given the forces and torques acting on it. It is a crucial factor in understanding the dynamics of the system.

Added mass matrix computation

The hydrodynamic added mass is a virtual mass added to a system due to the displacement of fluid as an object accelerates or decelerates. It is essential to consider this added mass for accurate calculations of the system's dynamics, as the object and fluid cannot occupy the same physical space simultaneously. This added mass significantly impacts the dynamics of the AUV, altering the vehicle's response to forces and inputs.

One method for calculating added mass is to employ slender body theory, which takes into account the vehicle's geometry and fluid properties, including density and viscosity. This theory assumes that the object is long and slender, meaning its length is significantly greater than its width and height [22].

Drag matrix computation

To estimate the effect of the main body shape, which resembles a sphere when moving in water, we consider it to have a hemisphere at the front and a cone at the back. For our calculations, we treat the cone as another hemisphere. The drag experienced by the AUV in water depends on various factors, including its shape, size, and velocity. Understanding and accurately computing the drag matrix is crucial for predicting the vehicle's motion characteristics.

Fossen movement equation application

The generation of Fossen movement equations involves deriving the mathematical equations that describe the motion of an underwater vehicle. These equations consider various forces and moments acting on the vehicle, including hydrodynamic forces, added mass effects, and external inputs, mentioned previously.

3.3.4 Rendering Techniques

To achieve the visual representation of the simulation, we employ noise-based techniques to provide a general texture to the ocean, which is applied to the shader mesh used to create the ocean's surface. Additionally, we apply the finite difference methods mentioned in [5] to simulate wave dynamics.

We will conduct an analysis to determine whether it is necessary to implement dynamic rendering techniques that prevent calculations in areas where there is no interaction between the fluid and objects in the ocean.

Fluid-Object Interaction

Building upon some of the methodologies mentioned in section 2.2, we will integrate simulations that consider how objects interact with the fluid. This includes the ability of objects to alter the flow and wave formation, as well as the object's response to water forces.

3.4 Multiverse integration

By developing an Integration interface or API that facilitates communication and data exchange between the selected physics and graphics engines, we aim to achieve seamless integration within the **Multiverse Simulation Framework.**[2]

[2] https://github.com/Multiverse-Framework

CHAPTER 4

Preliminar results

Contents

In this chapter, we present the preliminary results of our research, highlighting the feasibility of creating a basic fluid simulation within a video game engine and addressing the challenges encountered. The chapter's structure is divided into several sections: the first section focuses on the selection of the video game engine and its implications (Section 4.1), the second section deals with the initial configuration parameters of the mesh for simulating the ocean surface (Section 4.2), the third section centers on simulating buoyancy forces on an Unmanned Surface Vehicle (USV) (Section 4.3), the fourth and fifth sections detail the preliminary results of rendering the ocean surface and performance metrics of the simulation, including rendering times, computed primitives, and frames per second (FPS) (Sections 4.4 and 4.5), respectively. The sixth section is dedicated to initial geometric calculations related to the mass matrix of a underwater vehicle (Section 4.6). Finally, we conclude the chapter with a discussion of the findings and challenges encountered during the implementation of the simulation, highlighting the lessons learned in the process.

4.1 Choice of Game Engine

The preliminary results of our research, as detailed in the comparison table shown in Table 4.1 made by Collins et al. [2] in 2021 as a review of robotics simulators in different fields, highlight the features and capabilities of several popular simulators. Unity and Unreal Engine offer a wide range of functionalities, including support for multiple physics engines and ROS integration. However, a major limitation we identified is their direct compatibility with Linux-based systems, which are widely used in the field of robotics. This can lead to optimization and compatibility issues, as Linux-based systems are prevalent in this domain.

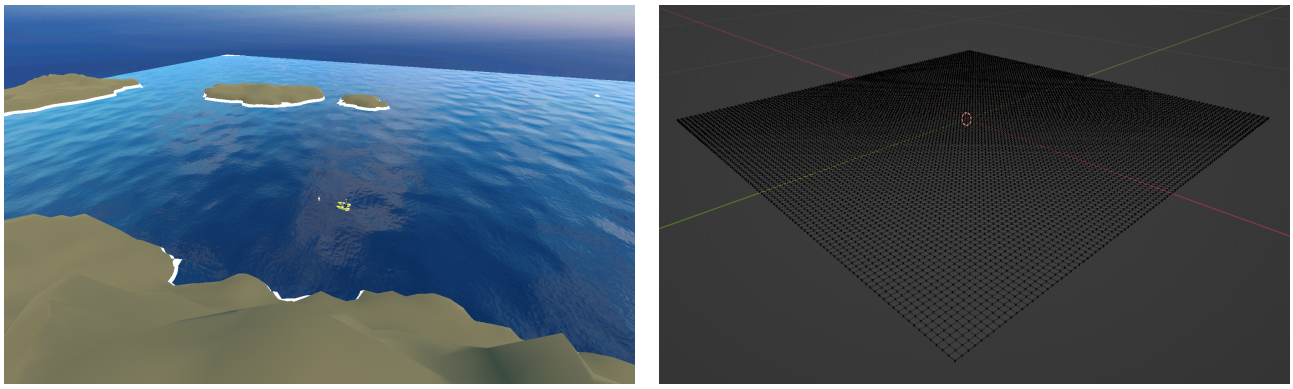
Simulator	RGBD + LiDAR	Force Sensor	Linear + Cable Acutator	Multi-Body Import	Soft-Body Contacts	DEM Simulation	Fluid Mechanics	Headless Mode	ROS Support	HITL	Teleoperation	Realistic Rendering	Inverse Kinematics
Airsim	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓, unreal	✗
CARLA	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓, unreal	✗
CoppeliaSim	✓	✓	Linear only	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓
Gazebo	✓	✓	Linear only	✓	✗	Through Fluidix	Through Fluidix	✓	✓	✓	✓	✗	✓
MuJoCo	✓	✓	✓	✓	✓	✓	Limited	✓	✗	HAPTIX only	HAPTIX only	✗	✗
PyBullet	✓	✓	Linear only	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓
SOFA	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓, Unity	✗
UWSim	RGBD only	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓, custom	✗
Chrono	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓, offline	✓
Webots	✓	✓	linear	✓	✗	✗	Limited	✓	✓	✗	✓	✗	✗

Table 4.1: Feature comparison between popular robotics simulators [2].

Considering our project's requirements and the importance of Linux compatibility, we initially opted for Godot, an entirely open-source engine native to Linux [41]. While this choice may entail sacrificing some expected final graphical quality, it offers the distinct advantage of better optimization for the operating system and the use of a lightweight graphics engine. Although other engines may provide higher graphical fidelity in the end, Godot's lightweight framework allows for superior optimization within the Linux environment, a critical consideration for our project.

4.2 Water surface parametrization

First, the shader was configured, which involved creating a grid measuring 150 meters by 150 meters, as shown in Figure 4.1a, subdivided into 1500 squares along both dimensions, as shown in Figure 4.1b. This setup provided a resolution of 10 centimeters, resulting in a total of 2,250,000 points used to determine the water surface resolution and interaction points with objects that interact with it. It was concluded that further increasing the resolution would be too demanding without parallelization, and that working with a much smaller grid would not yield any advantages in terms of visual appearance, computational capacity, or object interaction.



(a) Ocean surface preliminary extension.

(b) Ocean's base mesh.

Figure 4.1: Ocean surface preliminary visualization.

4.3 Fluid-Object Interaction preliminary results

Given that we have only worked with the shader mesh so far, the only interaction we can currently simulate is through the pivot points to replicate the buoyancy of the marine vehicle. This method differs from what is shown in [18] and in section 2.1.1. Instead of applying a buoyancy force to the center of gravity of the bounding box and its volume, in our case, each pivot point exerts a distributed buoyancy force. This means that we assume the vehicle's mass is evenly distributed across each of the flotation points, which are the vertical green lines located along the vehicle's surface as depicted in Figure 4.2.

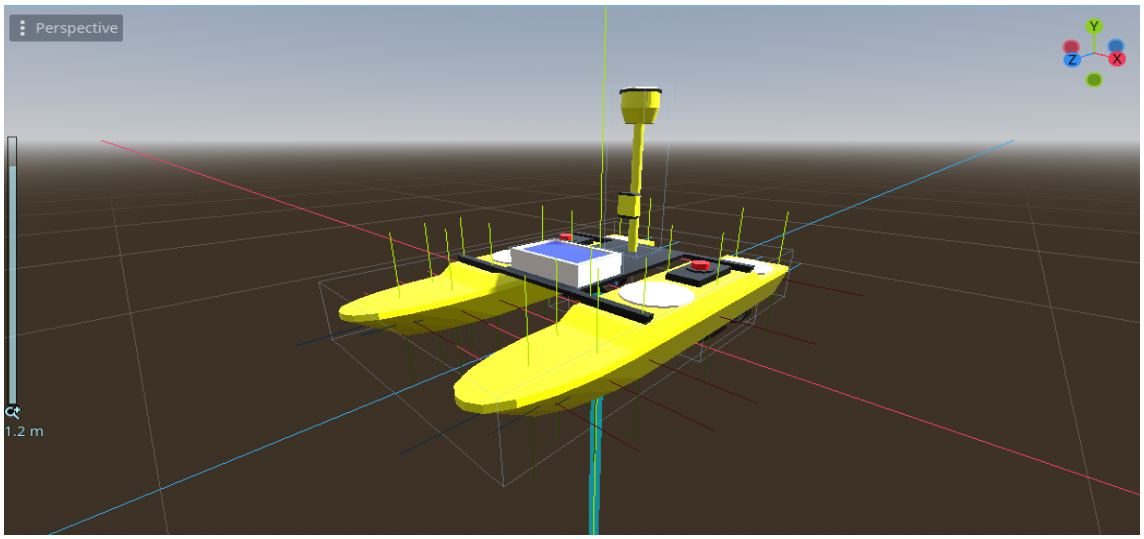
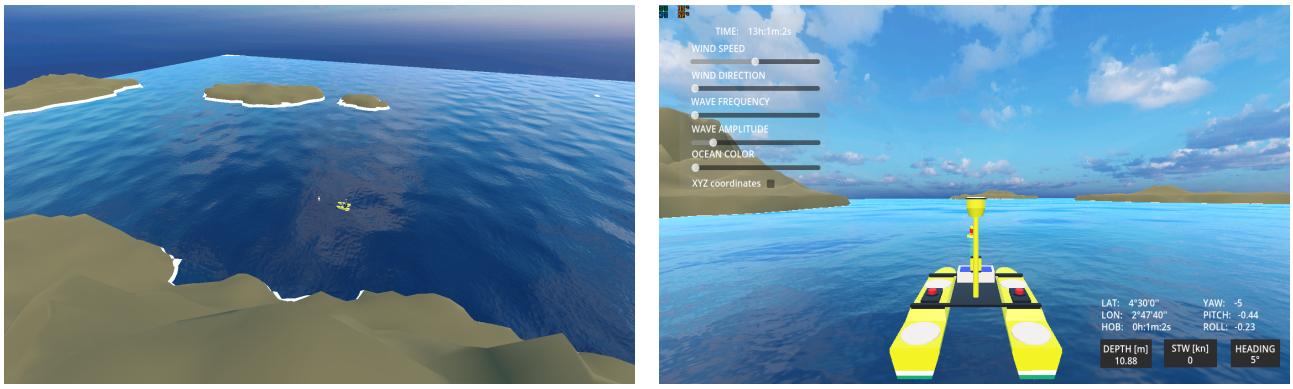


Figure 4.2: Contact points between the vehicle surface and the water.

4.4 Ocean rendering preliminary results

The shader techniques employed have resulted in a visually convincing representation of water, complete with light reflections, refractions, and wave movements, initially based on white noise height levels as can be seen in the Figure 4.3a.The shader's parameters can be been adjusted to reflect varying environmental conditions, allowing for real-time changes in the simulation by the adjustments in the use interface, as can be seen in the Figure 4.3b, also it is possible to see how the waves influence the yaw pitch and roll angles and the orientation in the same figure.



(a) Ocean render result.

(b) Environmental interface

Figure 4.3: Ocean rendering preliminary results.

4.5 Performance metrics

By evaluating, the real-time performance of the simulation in the Figure 4.4. A high frame rate of 240 FPS ensures a smooth visual experience, which is critical for accurate real-time simulations. The rendering times are notably low, with the GPU and CPU times recorded at 4.15 ms and 0.416 ms, respectively. These low rendering times suggest that the simulation is highly efficient, with minimal processing delay, which is crucial for maintaining real-time interaction within the simulation. The number of draw calls, at 75, indicates an optimized rendering pipeline, as fewer draw calls generally lead to better performance on a wide range of hardware. With 75 objects and 4,590,257 primitives, the simulation demonstrates its capability to represent a complex and detailed oceanic environment without compromising the performance.

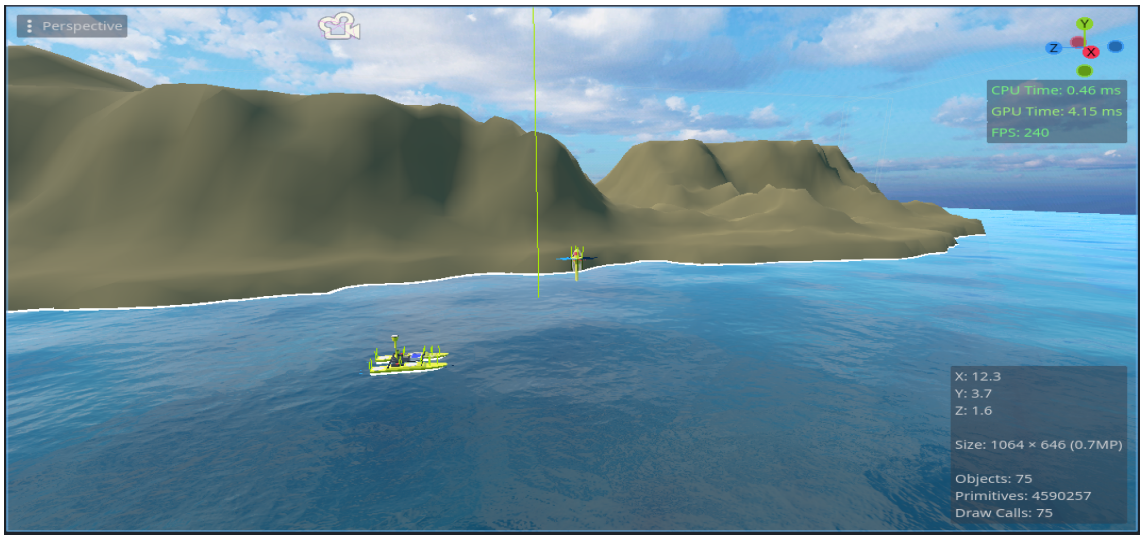
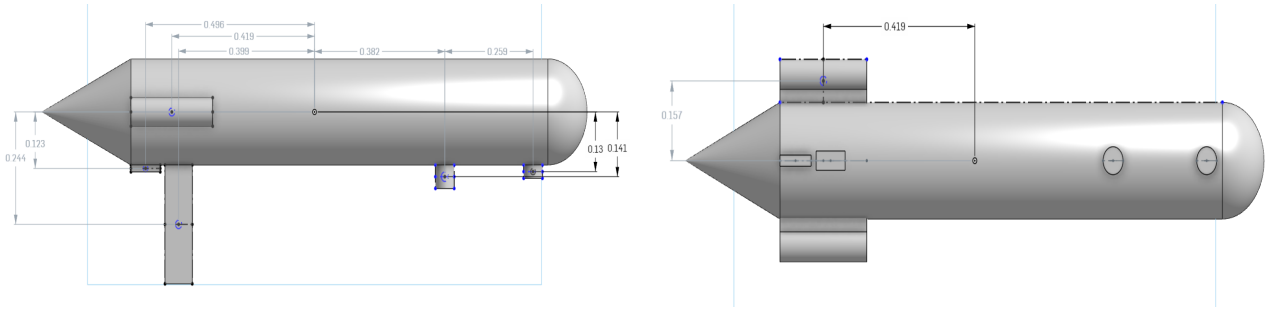


Figure 4.4: Preliminary performance metrics.

4.6 Geometric methods

For the geometric methods, we have opted to use the Sparrus II AUV as the reference model, as mentioned in Section 2.1.1. The mass matrix for this model has been computed using geometric methods described in [22, 3]. The mass matrix represents the distribution of mass in a rigid body. The diagonal elements of the matrix represent the mass of the body in each degree of freedom, meanwhile, the cross-coupling effects between the DOF, such as the interaction between translation and rotation. This matrix plays a fundamental role in the equations that define the movement of the vehicle, allowing us to calculate the accelerations of a given system given the forces and torques acting on it, making it a crucial factor in the dynamics of the system.



(a) Sparrus II UAV side view measurements model.

(b) Sparrus II UAV side top measurements model.

Figure 4.5: Sparus II 3D models.

If we consider the density of the vehicle as constant, we can use the CAD simplified model to calculate the vehicle's approximate volume and density as in shown in the Figure 4.5.

$$Density = \frac{Mass}{Volume} \tag{4.1}$$

$$Density = \frac{52\ kg}{0.06\ m^3} = 866.67\ kg/m^{-3}$$

In other hand, The Sparus allows defining the main axes and the position of the gravity center. Therefore, we consider the origin of the base **body** in the gravity center of the Sparus. It is placed at the x-position of the central thruster and at the middle of the cylinder.

$$M_{RB}^{CG} = \begin{bmatrix} 52.000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 52.000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 52.000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.404 & 0 & -0.057 \\ 0 & 0 & 0 & 0 & 8.930 & 0 \\ 0 & 0 & 0 & -0.057 & 0 & 8.922 \end{bmatrix}$$

In addition, by using a similar procedure as before, by isolating each part of the Sparrus and by translating the center of mass of each part to the center of gravity of the AUV using CAD software, we can make an independent and comparative analysis of the effect of each mass matrix on the dynamics of the vehicle.

4.7 FINAL REMARKS.

Based on the empirical data we have gathered, it has become evident that while Godot offers a huge advantage as our chosen game engine for the ocean simulation project, been capable of simulating vast extension of oceans and achieving photorealistic results keeping meanwhile keeping a physics simulation; however, there are notable challenges to address. Specifically, our experiments revealed issues in the integration of Godot, particularly in versions 4.XX, with some graphics cards, including the GTX 1070, RTX 3050 and a RTX 3050 Ti, where compatibility with the new Vulkan graphics engine appeared to be inconsistent. These compatibility concerns, coupled with loading problems observed on the Linux platform, even when the simulation was inactive, increased the concerns about system stability and performance. Furthermore, challenges in implementing necessary parallel computing components were encountered. These challenges indicate a need re-evaluation and potential consideration of an alternative game engine, a decision that will be done throughout the thesis phase. These findings will serve as guiding principles as we continue the simulation in alignment with our research objectives.

CHAPTER 5

Workplan

In this chapter, the initial schedule for my thesis proposal is defined. It is important to acknowledge that this project involves various aspects from different knowledge areas. To successfully complete this thesis within a six-month timeframe, a structured and detailed work schedule is necessary. This schedule will encompass all crucial activities and checkpoints required to advance the research in a disciplined and efficient manner.

5.1 Thesis Timeline

Table 5.1: Future Thesis timeline workflow.

February	Review and writing of the state of the art.
March	Second evaluation of the graphic engine and computing algorithm for open sea.
April-May	Computing the shallow water algorithm.
June	Computing algorithm for underwater vehicles and Integration with the Multiverse framework.
July	Writing the final thesis document.
August	Thesis defense.

Continuing, we provide a more detailed description of the work plan.

5.2 Tasks and Chronology

February: Review and Writing of the State of the Art

- Investigate and summarize current developments and relevant technologies.
- Identify gaps in existing research.

March: Evaluation of the Graphic Engine and Algorithm for Open Sea

- Evaluate and select appropriate simulation tools.
- Develop and test preliminary algorithms.

April-May: Development of the Shallow Water Algorithm

- Implement and optimize the algorithm.
- Conduct initial tests and adjustments.

June: Algorithm for Underwater Vehicles and Integration with the Multiverse Framework

- Develop simulation for underwater vehicles.
- Integrate with the existing simulation system.

July: Writing the Final Thesis Document

- Consolidate findings and analysis.
- Write methodology, results, and discussion chapters.

August: Thesis Defense

- Prepare the final presentation.
- Review and practice the defense.

5.3 Risks and Contingency Plan

- **Delays in Algorithm Development:** Plan additional time and consult with experts.
- **Challenges in System Integration:** Conduct regular tests and consider integration alternatives.
- **Preparation for Thesis Defense:** Practice sessions and peer review.

Note: During all this time, there will be continuous work on the writing of the final thesis document.

Bibliography

[1] Mabel M. Zhang et al. "DAVE Aquatic Virtual Environment: Toward a General Underwater Robotics Simulator". en. In: *2022 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. Singapore: IEEE, Sept. 2022, pp. 1–8. ISBN: 978-1-66541-689-4. DOI: 10.1109/AUV53081.2022.9965808. URL: https://ieeexplore.ieee.org/document/9965808/ (visited on 01/03/2024).

[2] Jack Collins et al. "A Review of Physics Simulators for Robotic Applications". en. In: *IEEE Access* 9 (2021), pp. 51416–51431. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3068769. URL: https://ieeexplore.ieee.org/document/9386154/ (visited on 01/03/2024).

[3] Michael V. Jakuba. *Modeling and control of an autonomous underwater vehicle with combined foil/thruster actuators*. en. Woods Hole, MA: Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, 2003. DOI: 10.1575/1912/2460. URL: https://hdl.handle.net/1912/2460 (visited on 12/03/2023).

[4] João Victor Nunes De Sousa et al. "On the Study of Autonomous Underwater Vehicles by Computational Fluid-Dynamics". en. In: *Open Journal of Fluid Dynamics* 10.01 (2020), pp. 63–81. ISSN: 2165-3852, 2165-3860. DOI: 10.4236/ojfd.2020.101005. URL: https://www.scirp.org/journal/doi.aspx?doi=10.4236/ojfd.2020.101005 (visited on 12/03/2023).

[5] Jerry Tessendorf. *Gilligan: A prototype framework for simulating and rendering maritime environments*. 2017. URL: https://people.cs.clemson.edu/~jtessen/papers_files/simdoc.pdf.

[6] Cem Yuksel. "Real-time Water Waves with Wave Particles". PhD thesis. Texas AM University, 2010. URL: https://hdl.handle.net/1969.1/ETD-TAMU-2010-08-8566.

[7] Dan Koschier et al. "A Survey on SPH Methods in Computer Graphics". en. In: *Computer Graphics Forum* 41.2 (May 2022), pp. 737–760. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.14508. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14508 (visited on 12/10/2023).

[8] Nuttapong Chentanez and Matthias Müller. "Real-time simulation of large bodies of water with small scale details". en. In: (2010). URL: https://api.semanticscholar.org/CorpusID:11375775.

[9] Matthias Müller et al. "Real time physics: class notes". en. In: *ACM SIGGRAPH 2008 classes*. Los Angeles California: ACM, Aug. 2008, pp. 1–90. ISBN: 978-1-4503-7845-1. DOI: 10.1145/1401132.1401245. URL: https://dl.acm.org/doi/10.1145/1401132.1401245 (visited on 01/03/2024).

[10] Stefan Jeschke et al. "Water surface wavelets". en. In: *ACM Transactions on Graphics* 37.4 (Aug. 2018), pp. 1–13. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3197517.3201336. URL: https://dl.acm.org/doi/10.1145/3197517.3201336 (visited on 01/03/2024).

[11] Jerry Tessendorf. "Simulating Ocean Water". In: 2004. URL: https://api.semanticscholar.org/CorpusID:1674789.

[12] L. S. Jensen and R. Goliáš. "Deep-Water Animation and Rendering". In: *Proceedings of the Game Developer's Conference*. 2001. URL: http://www.gamasutra.com/gdce/2001/jensen/jensen_01.htm.

[13] Jingcong Zhang. "Implementation and Applications of Art-directable Ocean Simulation Tools". In: 2018. URL: https://api.semanticscholar.org/CorpusID:209019259.

[14] Emmanuelle Darles et al. "A Survey of Ocean Simulation and Rendering Techniques in Computer Graphics". In: *CoRR* abs/1109.6494 (2011). arXiv: 1109.6494. URL: http://arxiv.org/abs/1109.6494.

[15] Sean Palmer et al. "The Ocean and Water Pipeline of Disney's Moana". In: *ACM SIGGRAPH 2017 Talks*. SIGGRAPH '17. Los Angeles, California: Association for Computing Machinery, 2017. ISBN: 9781450350082. DOI: 10.1145/3084363.3085067. URL: https://doi.org/10.1145/3084363.3085067.

[16] Rob Hopper and Kai Wolter. "The Water Effects of Pirates of the Caribbean: Dead Men Tell No Tales". In: *ACM SIGGRAPH 2017 Talks*. SIGGRAPH '17. Los Angeles, California: Association for Computing Machinery, 2017. ISBN: 9781450350082. DOI: 10.1145/3084363.3085044. URL: https://doi.org/10.1145/3084363.3085044.

[17] Moana Technical Team. *Moana Technical Team Instagram Page*. Apr. 2017. URL: https://www.instagram.com/disneyanimation/.

[18] Ricardo Mendonça et al. "Kelpie: A ROS-Based Multi-robot Simulator for Water Surface and Aerial Vehicles". In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. 2013, pp. 3645–3650. DOI: 10.1109/SMC.2013.621.

[19] Alexandra L. Zheleznyakova. "Physically-based method for real-time modeling of ship motion in irregular waves". In: *Ocean Engineering* 195 (2020), p. 106686. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2019.106686. URL: https://www.sciencedirect.com/science/article/pii/S0029801819308017.

[20] David House and John C. Keyser. *Foundations of Physically Based Modeling and Animation*. 1st. A K Peters/CRC Press, 2016. DOI: 10.1201/9781315373140. URL: https://doi.org/10.1201/9781315373140.

[21] Reagan Samuel Burke. "Rigid Body Dynamics of Ship Hulls via Hydrostatic Forces Calculated From FFT Ocean Height Fields". en. In: ().

[22] Josefine Severholt. "Generic 6-DOF Added Mass Formulation for Arbitrary Underwater Vehicles based on Existing Semi-Empirical Methods". In: (2017). URL: https://api.semanticscholar.org/CorpusID:201876843.

[23] Thanh-Long Le and Duc-Thong Hong. "Computational Fluid Dynamics Study of the Hydrodynamic Characteristics of a Torpedo-Shaped Underwater Glider". en. In: *Fluids* 6.7 (July 2021), p. 252. ISSN: 2311-5521. DOI: 10.3390/fluids6070252. URL: https://www.mdpi.com/2311-5521/6/7/252 (visited on 12/03/2023).

[24] Hamid Zeraatgar, Aliasghar Moghaddas, and Kazem Sadati. "Analysis of surge added mass of planing hulls by model experiment". In: *Ships and Offshore Structures* 15 (Aug. 2019), pp. 1–8. DOI: 10.1080/17445302.2019.1615705.

[25] Hampus Tober. "Evaluation of Drag Estimation Methods for Ship Hulls". 30 Credits. Second Cycle. Stockholm, Sweden: Stockholm University, 2020.

[26] Doug Perrault et al. "Sensitivity of AUV added mass coefficients to variations in hull and control plane geometry". In: *Ocean Engineering* 30 (Apr. 2003), pp. 645–671. DOI: 10.1016/S0029-8018(02)00041-0.

[27] Massachusetts Institute of Technology. *Marine Hydrodynamics*. 40th anniversary edition. Cambridge, MA: The MIT Press, 2017. ISBN: 9780262534826. URL: https://home.hvl.no/ansatte/gste/ftp/MarinLab_files/Litteratur/Marine_Hydrodynamics_Newman_2018.pdf.

[28] Thor I. Fossen. "Handbook of Marine Craft Hydrodynamics and Motion Control". In: (2016). DOI: 10.1002/9781119994138.

[29] Seong-Keon Lee et al. "Evaluation of the added mass for a spheroid-type unmanned underwater vehicle by vertical planar motion mechanism test". In: *International Journal of Naval Architecture and Ocean Engineering* 3.3 (Sept. 2011), pp. 174–180. DOI: 10.2478/ijnaoe-2013-0060.

[30] Hang Su et al. "Computational Fluid Dynamics Study of Autonomous Underwater Vehicle with Vectorial Thrusters". en. In: *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*. Kobe: IEEE, May 2018, pp. 1–5. ISBN: 978-1-5386-1654-3. DOI: 10.1109/OCEANSKOBE.2018.8559226. URL: https://ieeexplore.ieee.org/document/8559226/ (visited on 12/03/2023).

[31] Emmanouil Angelidis et al. "Gazebo Fluids: SPH-based simulation of fluid interaction with articulated rigid body dynamics". en. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 11238–11245. ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9982036. URL: https://ieeexplore.ieee.org/document/9982036/ (visited on 12/04/2023).

[32] Nicolas Gartner et al. "Can smoothed particle hydrodynamics simulate physically realistic movements of underwater vehicles?" en. In: *Advanced Robotics* 37.20 (Oct. 2023), pp. 1283–1300. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2023.2263046. URL: https://www.tandfonline.com/doi/full/10.1080/01691864.2023.2263046 (visited on 12/04/2023).

[33] T. R. Hagen et al. "Visual Simulation of Shallow-Water Waves". In: *Simulation Modelling Practice and Theory* 13.8 (2005), pp. 716–726.

[34] J. Kim et al. "Fast GPU Computation of the Mass Properties of a General Shape and Its Application to Buoyancy Simulation". In: *The Visual Computer* 22 (2006), pp. 856–864.

[35] Mark Harris. "Fast Fluid Dynamics Simulation on the GPU". In: *GPU Gems*. Ed. by Randima Fernando. Addison-Wesley, 2004. Chap. 38, pp. 637–665.

[36] Keenan Crane, Ignacio Llamas, and Sohaib Tariq. "Real-Time Simulation and Rendering of 3D Fluids". In: *GPU Gems 3*. Ed. by Hubert Nguyen. Addison-Wesley, 2007. Chap. 30, pp. 633–675.

[37] H. Cords. "Mode-Splitting for Highly Detailed, Interactive Liquid Simulation". In: *GRAPHITE '07: Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. 2007, pp. 265–272.

[38] Simon Clavet, Pierre Beaudoin, and Pierre Poulin. "Particle-Based Viscoelastic Fluid Simulation". In: *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. New York, NY, USA: ACM, 2005, pp. 219–228.

[39] Adrien Treuille, John Lewis, and Zoran Popovic. "Model Reduction for Real-Time Fluids". In: *ACM Transactions on Graphics* 25.3 (2006), pp. 826–834.

[40] Jeremiah U. Brackbill and H. M. Ruppel. "FLIP: A Method for Adaptively Zoned, Particle-in-Cell Calculations of Fluid Flows in Two Dimensions". In: *Journal of Computational Physics* 65 (1986), pp. 314–343.

[41] UnionBytes. *Beautiful realistic water shader from UnionBytes*. 2021. URL: https://godotshaders.com/shader/realistic-water/.

[42] Ivan Loncar et al. "MARUS - A Marine Robotics Simulator". en. In: *OCEANS 2022, Hampton Roads*. Hampton Roads, VA, USA: IEEE, Oct. 2022, pp. 1–7. ISBN: 978-1-66546-809-1. DOI: 10.1109/OCEANS47191.2022.9976969. URL: https://ieeexplore.ieee.org/document/9976969/ (visited on 01/03/2024).

[43] Shubham Garg et al. "NetMarSyS - A Tool for the Simulation and Visualization of Distributed Autonomous Marine Robotic Systems". en. In: *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)(50043)*. St Johns, NL, Canada: IEEE, Sept. 2020, pp. 1–5. ISBN: 978-1-72818-757-0. DOI: 10.1109/AUV50043.2020.9267922. URL: https://ieeexplore.ieee.org/document/9267922/ (visited on 01/03/2024).

[44] Ricardo Mendonca et al. "Kelpie: A ROS-Based Multi-robot Simulator for Water Surface and Aerial Vehicles". en. In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. Manchester: IEEE, Oct. 2013, pp. 3645–3650. ISBN: 978-1-4799-0652-9. DOI: 10.1109/SMC.2013.621. URL: http://ieeexplore.ieee.org/document/6722374/ (visited on 01/03/2024).

[45] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[46] E. Potokar et al. "HoloOcean: An Underwater Robotics Simulator". In: *Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA*. Philadelphia, PA, USA, May 2022.

[47] Abdelhakim Amer et al. *UNav-Sim: A Visually Realistic Underwater Robotics Simulator and Synthetic Data-generation Framework*. 2023. arXiv: 2310.11927 [cs.RO].

[48] Thuynh4 - RhinocerosForum. *Is there a way to remesh STL files while maintaining connectivity?* 2021. URL: https://discourse.mcneel.com/t/is-there-a-way-to-remesh-stl-files-while-maintaining-connectivity/105077/1.

[49] Blender. *The Freedom to Create* . 2021. URL: https://www.blender.org/about/.

[50] iquarobotics. *Sparus II technical overview*. 2023. URL: https://iquarobotics.com/wp-content/uploads/2018/03/SPARUS_II_brochure.pdf.