Digital Fishes

An Interactive Virtual Aquarium

Master's degree in Computer Engineering - Mobile Computing

David Cervantes Pérez

Leiria, September of 2022

Digital Fishes

An Interactive Virtual Aquarium

Master's degree in Computer Engineering - Mobile Computing

David Cervantes Pérez

Project Report under the supervision of Professor Nuno Rodrigues and Professor Rita Ascenso

Leiria, September of 2022

Originality and Copyright

This project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Dedication

To my lovely wife and family for everything.

Acknowledgments

Thanks to the teachers and organizers of the Master for their professionalism and support. To the tutors of this work who have accompanied it from the beginning with great dedication.

This work is supported by AUIP - Asociación Universitaria Iberoamericana de Postgrado.

Abstract

This work describes the creation of an interactive virtual aquarium, Digital Fishes, with a physical representation in a cubic fish tank with synchronized visual information and animated elements, such as fish and vegetation, which can be used for educational purposes. Here the solution to simulate the movement of marine flora and fauna added to our virtual Aquarium is described. An implementation of the fish behavior, to move in groups using the Boids algorithm, is created, in addition to autonomous behaviors such as chasing prey and fleeing from predators, interactive behaviors such as following the finger on the screen and going towards the food that the user may throw in the pond. The solution described in this work has managed to bring into the hands of children an interactive virtual representation of an interactive as one more learning tool at their disposal. These children who participated in the Digital Fishes Interactive Aquarium's evaluation provided positive feedback, showing great enthusiasm and positively assessed the solution.

Keywords: Digital Fishes, Interactive Virtual Aquarium, Boids Behavior, User Interaction, Ubiquitous Interfaces, Computer Graphics

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AABB	Axis-Aligned Bounding Box
API	Application Programming Interface
CPU	Central Processing Unit
DOTS	Data-Oriented Technology Stack
ESTG	School of Technology and Management
GLSL	OpenGL Shader Language
GPGPU	General-Purpose computing on Graphics Processing Units
GPU	Graphics Processing Unit
HLSL	High-Level Shader Language
ICGI	International Conference on Graphics and Interaction
MAODS	Metallic Ambient Occlusion and Smoothness
OCI	Oracle Cloud Infrastructure
REST	Representational State Transfer
UI	User Interface
URP	Universal Render Pipeline
VM	Virtual Machine
VR	Virtual Reality

1. Introduction

Covering two-thirds of the earth's surface, the seas host sea life that not everyone will have the opportunity to see. Institutions such as aquarium museums try to bring the beauty of marine nature to the person without having to go sailing. These institutions also aim to educate the public about caring for the sea and nature by offering beautiful educational activities for children. In these, children learn about fishes and their habitats.

More recently, with the development of technology and increasing computing power, technologies that were expensive or impossible to implement in the past are now becoming accessible on laptops and mobile phones. As a result, these technologies are revolutionizing how we interact with the world, learn, and teach. Virtual Reality (VR) is one of these technologies that are becoming more accessible every time.

Virtual Reality (VR) has great potential to transform how people and businesses interact with each other and the environment. While this innovative technology has traditionally been associated with the gaming industry, it is surprisingly becoming more popular. Being applied in various fields to manipulate the physical environment as virtual reality allows its users to experience the natural setting more immersively.

Some software development companies have already begun creating virtual environments that recreate marine and aquatic life. This inspired us to create an interactive virtual aquarium with a physical representation in the form of a cubic fish tank. Also, the visual information and animated elements, such as fish and vegetation, are synchronized between the lateral faces of the cubic tank. In addition, it can be used for educational purposes and thus contribute to a better understanding of marine life in Portugal.

1.1.Motivation and goals of the research

The primary motivation for this work is to build an application for educational purposes that resembles a natural aquarium where we can enjoy a seabed and the marine life of the region of Portugal, which behaves as naturally as possible and with which we can interact.

The main goal of the work is to ***build an Interactive Virtual Aquarium***.

The following particular goals were met during the research to fulfill the main goal:

- A website with the aquarium and decorative elements.
- Implement or adapt an algorithm that allows fish to follow paths, avoid obstacles, and each other according to their natural behavior, as much as possible.
- Implement the parameterization and configuration features of the aquarium elements and interaction.
- Create the necessary mechanisms to synchronize the various views of the aquarium on several terminals.

Along the project's development, scientific research was performed to identify adequate methodologies and select the technologies. The work methods used in this research were: a systemic method for the development of computer systems and ensuring that the elements that are part of the actual application are a whole that works together as a unit; the historical-logical method for the critical study of previous works, and use these as a point of reference and comparison of the results obtained, the analytical-synthetic method by breaking down the research problem into separate elements and deepening the study of each one them, to then synthesize them in the proposed solution.

1.2.Document structure

The document consists of an introductory chapter describing the context of the master thesis and presenting the work to be carried out, the research structure, and the project's main objectives and expected results.

The second chapter (State of the Art of Interactive Virtual Aquariums) presents the theoretical and practical elements that constitute the investigation to select the framework and determine the work's essential elements. In this chapter, different Interactive Virtual Aquariums were studied using a set of indicators, and the basic features of these solutions were analyzed. Also, a sample of species of fish and algae was selected through a study carried out in the marine environment of Portugal.

The third chapter (Methodology) presents the methodological part before developing the solution. Therefore, the chapter begins by presenting the methodologies used in the development process. After these, the requirements for the solution are defined and described. Finally, the architecture and technologies to carry out the requirements of this solution are presented.

The fourth chapter (Development) presents the implementation phase. The chapter describes the work done in the implementation phase to fulfill the requirements, including the description of the solution's graphic design, the user interface, color selection, and 3d models of fish and algae.

The fifth chapter (Evaluation) describes and exposes the result and evaluation of the obtained solution. Appendix A presents the System Usability Scale (SUS) questionnaire used for this evaluation.

The document includes the publication and conclusions derived from the study carried out and recommendations that include elements in future works.

2. A state-of-the-art of Interactive Virtual Aquarium

The objective of this chapter is to present the theoretical foundations of the research. It shows the study on a group of interactive virtual aquariums based on defining a group of indicators.

These indicators were defined to characterize the interactive virtual aquariums. The analysis of these solutions allowed us to determine the essential features an interactive virtual aquarium must offer.

2.1. Interactive Virtual Aquariums

According to J. R. Li [1], Virtual reality (VR) is a synthetic, three-dimensional, interactive environment typically created by a computer. It provides a unique avenue to enhance the visualization of complex three-dimensional objects and environments with real-time, more interactive, and spatial ability. Furthermore, as Robertson [2] says, a Desktop VR system uses animated interactive 3D graphics to build virtual worlds with desktop displays and without head tracking.

So we can say that an **Interactive Virtual Aquarium** is the 3D representation of a natural aquarium with which we can interact. In our case, the goal is to build a virtual aquarium with the immersive characteristics of a Desktop VR system with which we can interact through a device's screen.

Lee and colleagues [3] presented a method to build a virtual aquarium environment system that can be controllable in real-time by expressing the movement of algae using virtual fish and control points based on the spring-mass model. In addition, a fluid expression generator is presented based on fluid mechanics and control fluid flow. In our case, the solution will be more straightforward, without fluid dynamics, and the movement of the algae and fish is performed using vertex displacement along the direction axis of the fish with a sinusoidal function to simulate the movement of locomotion. Finally, Lypovyi and Montusiewicz [4]

use Unity as a platform to develop simulation experiments with different configurations of force coefficients that control a model based on Boid behavior.

A group of solutions was studied to analyze the interactive virtual aquariums, characterizing them with indicators. This characterization lets us know what elements we must consider creating an interactive virtual aquarium.

The defined indicators shown in Table 1 have been defined as a way to identify the main characteristics that an interactive virtual aquarium must have. Finally, Table 2 is presented, summarizing each of the solutions' characteristics and identifying whether or not the solution has that characteristic.

Indicator	Description
Type (TP)	It refers to the format in which the solution is presented. A. Screensaver (SS) B. Real, live stream (RTS) C. Interactive for large surfaces and attractions (ILA)
Target audiences (TA)	It refers to the audience to which the solution is intended, age limit. A. Adults B. Everyone
Decorative elements (DCE)	It refers to the decoration of an aquarium. A. Tropical reefs B. River bottoms C. Artificial decoration for aquariums D. The habitat of the species recreated E. None

Fish behavior (FB)	It refers to the interaction of the fish with each other and the elements surrounding them. A. Highly realistic B. Realistic C. Real D. Individual movement E. Individual and groups movement F. Predator / Prey
Interaction with the fish (IF)	It refers to the user's interaction with the fish and how they react to it. A. Allows feeding B. They flee or chase the user through contact with the surface of the device. C. None D. Unknown
Didactic elements (DIE)	It refers to the didactic level that the solution presents. A. Simple description of the fish B. A detailed description of the fish C. None D. Unknown
Configuration (CG)	It refers to the solution's ability to allow some configuration. A. Habitat configuration, example: decorative elements B. Configuration of the fish, example: quantity and species C. Fish behavior D. None E. Unknown
Remote Configuration (RCG)	Refers to the solution's ability to store its configuration in a remote location. A. Yes B. No C. Unknown

Real-time synchronization (RTS)	It refers to the characteristic of the solution to display the information on the screen in a synchronized way in real-time. A. Yes B. No C. Unknown
Platform (PLF)	It refers to the platform for which the solution is released. A. Windows B. macOS C. Game consoles D. Mobile E. Web F. Unknown

Table 1. Interactive Virtual Aquarium analysis set of indicators

The following virtual aquariums solutions were studied: Dream Aquarium[1], Aqua TV[2], Marine Aquarium[3], Amazing 3D Aquarium[4], Sim Aquarium[5], Aqua Real[6], SEA LIFE – Melbourne, Virtual Aquarium[7], New England Aquarium, Virtual Visit[8], Vertigo systems, Living aquarium[9], TeamLab, Sketch Aquarium[10].

In each case, an analysis of the virtual aquariums was made to fill out a summary table. The final table helped identify and select the elements that these types of systems usually have.

[1] https://www.dreamaquarium.com
[2] https://aqua-tv.co.uk
[3] https://www.prolificpublishinginc.com/store/product.php?productid=64
[4] https://amazing-3d-aquarium.en.softonic.com
[5] http://simaquarium.com/en
[6] https://www.baixaki.com.br/download/digifish-aqua-real-2-steam.htm
[7] https://www.visitsealife.com/melbourne/whats-inside/virtual-aquarium
[8] https://www.neaq.org/visit/at-home-events-and-activities
[9] https://www.vertigo-systems.de/en/products/interactive-aquarium/living-aquarium
[10] https://futurepark.teamlab.art/en/playinstallations/sketch_aquarium_connected_world

2.1.1. Dream Aquarium

Beautiful graphics and many great features are offered by this virtual aquarium & screensaver, Dream Aquarium. This is one of a virtual aquarium's most realistic fish motions and behavior. The fishes have articulated fins, moving eyes, gills, and mouths that create a realistic behavior overall. Also, it has beautiful shifting light rays, ground ripples, gently waving plants that fish can swim into, soft shadows cast by fish, and configurable bubble streams. In addition, the application offers several different aquariums that can be customized by adding different species of fish, which can be added well over a hundred from 27 species.

Regarding the configuration, it can be switched between different aquariums, and adjust the monitor settings. Can change the location and speed of bubbles and adjust the light rays. Also, the user can feed the fish at any time or use the automatic feeding option.

Figure 1. Dream Aquarium [5]

2.1.2. Aqua TV

Aqua TV is a fully customizable virtual aquarium with beautifully created and animated 3D fishes. It has a simple user interface. First up, choose a tank from the three available: Small, Medium, and Large, and start to populate it. The smaller the tank, the fewer fish can fit in it, and some fish can only be used in tanks suitable for their breed. Other than this, the background can be changed, walls and decoration. Furthermore, all tank creations are stored on iCloud and can be shared with Apple TV to enjoy on the big screen. Also, the tanks can be viewed from the front or above and have a random mode to switch between them.

Figure 2. Aqua TV [6]

2.1.3. Marine Aquarium

Marine Aquarium is a virtual aquarium that works as a screensaver. It offers excellent graphics and 28 fish species that can be held up to 30 fish simultaneously. It has only one environment, but for the fish, this aquarium offers a wide range of customization, can be added a specific fish to an aquarium, or a random option can be used that will always add different species of fish. In addition, specific species of fish can also be disabled from appearing, or specific fish sets can be created and toggled between them. Regarding the fish, each species' name can be seen along with its location, diet, and size.

Figure 3. Marine Aquarium [7]

2.1.4. Amazing 3D Aquarium

This virtual aquarium and screensaver offer solid graphics and decent customization options, and it allows you to add up to 10 fish to the aquarium. Regarding the fish, the Amazing 3D Aquarium application supports about 20 species but can also be downloaded more from the developer's website. The aquarium can also have random fishes in it or can be chosen which species are wanted. Amazing 3D Aquarium also supports different backgrounds that can be chosen or used randomly.

Figure 4. Amazing 3D Aquarium [8]

2.1.5. Sim Aquarium

Sim Aquarium offers high-quality 3D graphics and is one of the best-looking virtual aquariums. The application offers about 30 species of fish that can be added and about ten different environments. The amount and speed of bubbles and the number of light rays can be controlled. Also, the application has a valuable stats manager on the left that will show the number of specific fish in the aquarium. It can also work as a screensaver replacement. Sim Aquarium is a virtual aquarium with excellent graphics. As a result, it might use many system resources, but it offers many settings related to video quality.

Figure 5. Sim Aquarium [9]

2.1.6. Aqua Real

The Aqua Real virtual aquarium uses advanced computational 3D rendering to create detailed representations of the swimming styles, living organisms, and habits of different types of fish. It also shows realistic movements of coral, sea anemones, and fish. They claim that it is an eco-simulated program, a virtually created reality built with a self-created biological engine after extensive research into the domain. The behavior of the fish change depending on the spice; some live alone, and some swim in groups. Occasionally they attack one another due to the basic instinct of territorial aggression. They also swim for their lives when they sense sharks lurking nearby.

They developed a realistic and efficient anemone-clown fish cline behavior to make coral reefs more realistic, including a physics operation animalism behavior. It represents natural libration movements while it swims through an anemone's tentacles.

There are 29 types of tropical fish and six types of coral reef scenes and can be added fishes and scenes from the company's website and perform interactions such as touch, feed, and call a shark. In addition, the combination of fish types and scenes can be changed in the settings.

Figure 6. Aqua Real [10]

2.1.7. SEA LIFE – Melbourne, Virtual Aquarium

Virtual Aquarium is an initiative to bring the attractions of the SEA LIFE – Melbourne natural aquarium to homes through live streams. The site also offers various educational resources to learn about Marine life through coloring games and initiatives to build coral reefs with crafts.

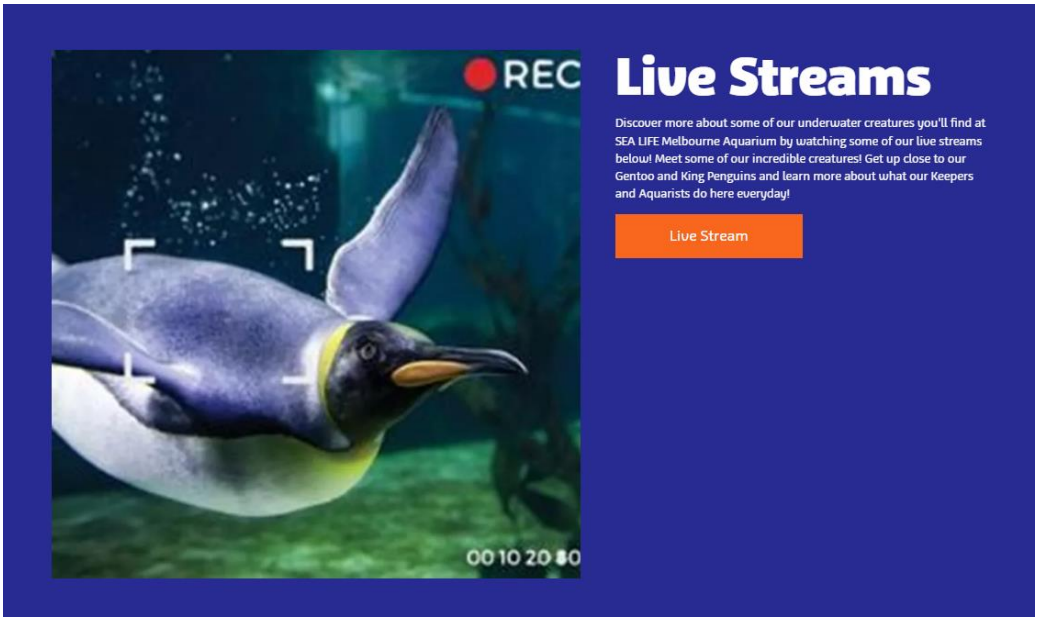

Figure 7. Web screenshot of SEA LIFE Melbourne, Virtual Aquarium Live stream [11]

Figure 8. Web screenshot of SEA LIFE Melbourne, Virtual Aquarium Educational activities [11]

2.1.8. New England Aquarium, Virtual Visit

Virtual Visit offers a service like that of Melbourne, Virtual Aquarium, which offers educational activities designed to be carried out from homes. They also offer live streams of enclosures and fish tanks of marine animals.

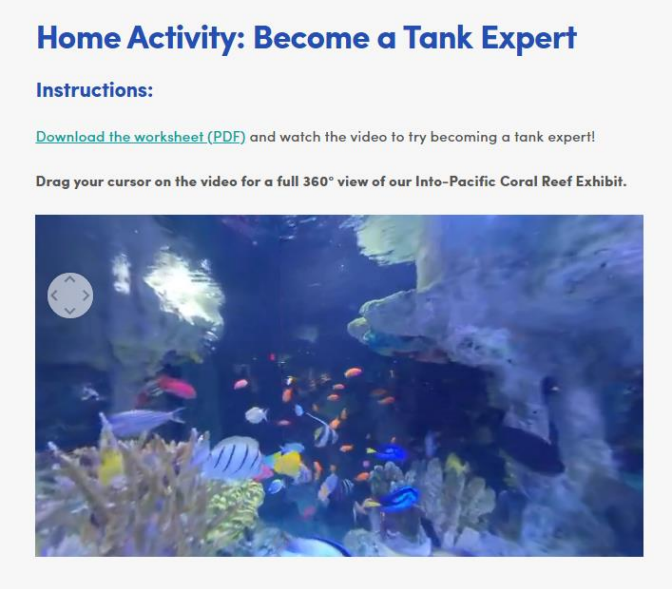

Figure 9. Web screenshot of New England Aquarium, Virtual Visit Educational activities [12]

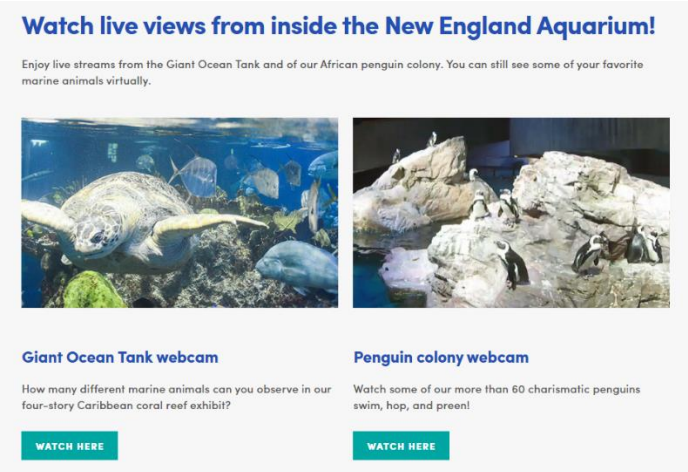

Figure 10. Web screenshot of New England Aquarium, Virtual Visit Live Stream [12]

2.1.9. Vertigo systems, Living aquarium

Inside the interactive virtual aquarium, fish, water, and other genuine sea creatures react to a person's movements. For example, if the fish are touched, they try to escape or follow a person's movements. The aquarium is available in various sizes for floor, wall, table, bar, and counter. The 3D aquarium can be individually designed. There is a vast selection of realistic 3D fish, cartoon fish, and unique creatures such as mermaids, dolphins, sharks, and others. All creatures behave very realistically. It can be changed the fish set, choose other living environments, and integrate logos, images, or adverts into the aquarium. Also, at the touch of a fish, it shows his name over his head.

This virtual aquarium solution also has a particular configuration called **paint2life** that makes it possible to color the animals, scan them and release them into the aquarium. This feature is mainly for kids. The creatures created can move around and react to the kids. The children can play with and even feed them.

Figure 11. Vertigo systems, Living aquarium floor [13]

Figure 12. Vertigo systems, Living aquarium fish labels [13]

Figure 13. Vertigo systems, Living aquarium, paint2life [13]

2.1.10. TeamLab, Sketch Aquarium

Children can observe the power of their creative imagination through Sketch Aquarium. Each participant is invited to color a drawing of a sea creature of his or her preference. Once completed, the paper is scanned, and the image is projected onto a giant virtual aquarium. Children will be able to see their creations come to life and swim with all the other sea creatures. Children may also touch the fish to see them swim away or touch the virtual food bag to feed the fish.

Figure 14. TeamLab, Sketch Aquarium [14]

2.2. Summary of the studied solutions

Table 2 summarizes the indicator's value for every Interactive Virtual Aquarium analyzed. The possible value for each indicator is defined in Table 1. The letters (**A, B, C, D, E, and F**) refer to each possible value's defined identifier. Note that the product site or documentation is not explicitly stated when the value refers to Unknow. Also, an indicator can meet more than one value. Table 1 shows the description of the indicators that were taken into account: Type (TP), Target audiences (TA), Decorative elements (DCE), Fish behavior (FB), Interaction with the fish (IF), Didactic elements (DIE) Configuration (CG), Remote Configuration (RCG), Real-time synchronization (RTS) and Platform (PLF), below the summary (Table 2).

Interactive Virtual Aquariums	Indicator									
	TP	TA	DCE	FB	IF	DIE	CG	RCG	RTS	PLF
Dream Aquarium	A	B	B	A	A	C	A|B	B	B	A
Aqua TV	A	B	A|C	B|D	C	A	A|B	A	B	A|C
Marine Aquarium	A	B	A|C	B|D	C	B	A|B	B	B	A|D
Amazing 3D Aquarium	A	B	A	B|E	C	C	B	B	B	A
Sim Aquarium	A	B	A	B, E	C	A	A|B	B	B	A
Aqua Real	A	B	A	A|E|F	A|B	C	A|B	B	B	A
SEA LIFE Melbourne, Virtual Aquarium	B	B	D	C	C	C	C	B	A	E
New England Aquarium, Virtual Visit	B	B	D	C	C	C	C	B	A	E
Vertigo systems, Living aquarium	C	B	A	B|E	A|B	A	B	C	C	F
TeamLab, Sketch Aquarium	C	B	A	B|E	A|B	C	B	C	C	F

Table 2. Summary of indicators for Interactive Virtual Aquariums studied, indicators' possible values are defined in Table 1

From the information in Table 2, we can infer that the characteristic that an Interactive Virtual Aquarium may have, is an environment that has elements from a coral reef and fishes.

The fishes may look as realistic as possible, swim in groups or alone, and react to user interactions through contact with the devices' screen. The behavior during the interaction can vary between fleeing or approaching the origin of the contact. It is also expected to present some characteristic that allows it to feed the fish. Furthermore, in a few cases, it found solutions that exhibit predator/prey behavior in fish. They may also contain some feature that allows the user to learn to identify fish by name.

Only a few solutions have unique features that allow creating new fish models by scanning a drawing and adding it to the aquarium as a service in facilities that are prepared for that.

It can be said that all of them present some configuration that allows changing the number and species of fish shown in the tank. In addition, it can be configured how the aquarium looks. However, only in one case could it be seen that the configurations are stored remotely, but no case could be verified where the remote configuration was allowed.

It can also be seen that few solutions have educational features, but all these solutions can be enjoyed by all audiences and are mainly released as a screensaver for windows.

In particular cases, the services provided by natural aquariums were found to bring the virtual experience of fish tanks to homes throw live streams.

2.3.Portugal’s marine life

The richness of Portugal's marine life has led researchers to study it in depth. As a result, the species that inhabit the Portuguese seas have been continuously documented.

The selection made of these species of fish and algae to have a representation in the Interactive Virtual Aquarium is described below.

2.3.1. Algae selection.

In this study by Freitas and colleagues [15], the edible algae found on the Portuguese coast are documented. Of these, the *Grateloupia Turuturu* has been chosen, which is an edible red alga. Although it is an invasive species of these seas, it is rich in protein and fiber, with

beneficial effects on health. Has commercial interest as a producer of carrageenan, whose properties include anticoagulant activity.

Figure 15. Grateloupia Turuturu [15]

Because of its presence in marine estuaries of the Portuguese coast, as emphasized in this study of protected areas [16], the *Zostera Marina* has been selected to be represented and be part of the developed aquarium. This alga is not considered edible but plays an essential role in the ecosystem for other species inhabiting it.

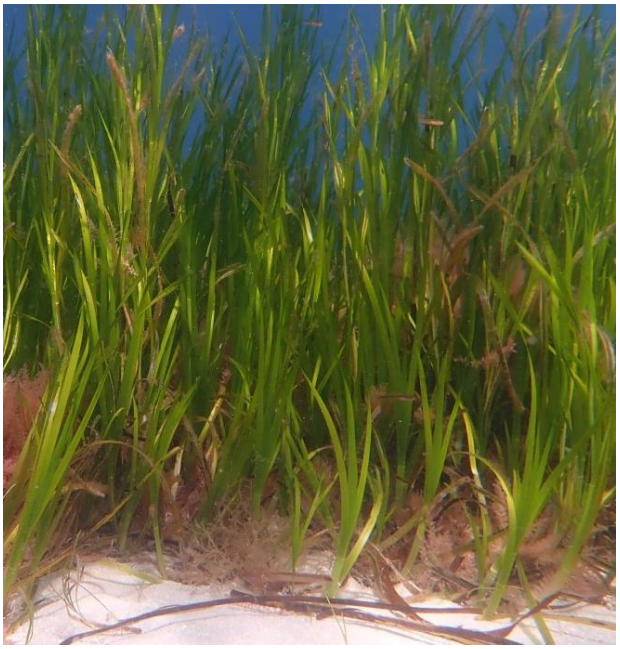

Figure 16. Zostera Marina

2.3.2. Fishes selection

In this article of *Ciência Viva* [17], a selection of the most popular species of fish from the Portuguese seas is described, of which we have chosen the following classified by size and habitual social behavior:

Small fish shoals*: Salmonete - Mullus surmuletus; Pescada-branca - Merluccius merluccius; Goraz - Pagellus bogaraveo.*

Big fish shoals: *Atum-albacora - Thunnus albacares*

Solitary or small shoals fishes: *Dourada – Sparus aurata*

Solitary fishes: *Robalo - Dicentrarchus labrax; Espadarte - Xiphias gladius; Cherne - Polyprion americanus;*

Figure 17. Fishes selection [17].

For the developed aquarium, the models were constructed and introduced, respecting the properties of each fish species in nature.

2.4. Chapter Summary

The second chapter addresses the fundamental elements that allow knowing the characteristics of Interactive Virtual Aquariums solutions, starting by defining an Interactive Virtual Aquarium. A group of indicators is also defined in the body of the chapter to perform the analysis of a group of Interactive Virtual Aquarium solutions selected. The analysis of several Interactive Virtual Aquarium solutions using the defined indicators is included. The chapter also summarizes the characteristics and indicators to be selected for implementation in the solution Digital Fishes.

After the state-of-the-art made in this chapter, the researcher reaches the following conclusions:

- An **interactive Virtual Aquarium** is the 3D representation of a natural aquarium with which the user can interact. In our case, the goal is to build a virtual aquarium with the

immersive characteristics of a Desktop VR system with which we can interact through a device's screen.

- The basic functionalities that an **Interactive Virtual Aquarium** solution must have are decorative elements (in most cases, coral reefs), the fish must have realistic locomotion, and present behaviors that show interactions between them and the user. In addition to characteristics that allow feeding the fish and seeing scientific information about each fish. Also, the solutions should present some configuration that allows at least to define the fish shown in the aquarium.
- It was noted that most of these solutions are presented as screensavers and as a plus to create pleasant surroundings since an aquarium can affect people. However, this tells us that they do not usually interact with the user, and few have educational characteristics like minimum detailed descriptions of the fishes.
- It can also be appreciated that these solutions are usually local and do not contain remote configuration.
- A solution was not found to present a real-time synchronization of the aquarium on multiple devices, except in the live stream services that some centers offer, unlike the proposed solution in this paper, where this is one of its main characteristics.
- A study was carried out, which resulted in a selection of species of fish and algae representative of the Portuguese seas.

The following chapter will address the methodological part of the development process to create the Interactive Virtual Aquarium, Digital Fishes.

3. Methodology

The objective of this chapter is to present the methodological part prior to the development of the solution. Therefore, the chapter begins by presenting the methodologies used in the development process. After these, the requirements for the solution are defined and described. Finally, the architecture and technologies to carry out the requirements of this solution are presented.

3.1. Software Development Methodology

The Scrum framework was the software development methodology for developing our Interactive Virtual Aquarium. As its creators defined it, “Scrum is a framework within which people can address complex adaptive problems while productively and creatively delivering products of the highest possible value” [18].

Scrum is an agile methodology used in software development based on iterative and incremental processes. It was designed to deliver value to the customer throughout the project development. Agile methodologies aim to deliver the right product, with incremental and frequent delivery of small chunks of functionality, through small cross-functional self-organizing teams, enabling frequent customer feedback and course correction as needed. Agile software development methodologies are iterative, dividing the work into iterations [19].

Figure 18 shows the Scrum Framework artifacts and Events. Scrum’s artifacts represent work or value to provide transparency and opportunities for inspection and adaptation [19].

Artifacts defined by Scrum are specifically designed to maximize the transparency of essential information so that everybody has the same understanding of the artifact. The Scrum Artifacts are Product Backlog, Sprint Backlog, and Increment [19].

The Scrum Events are Sprint, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective [19].

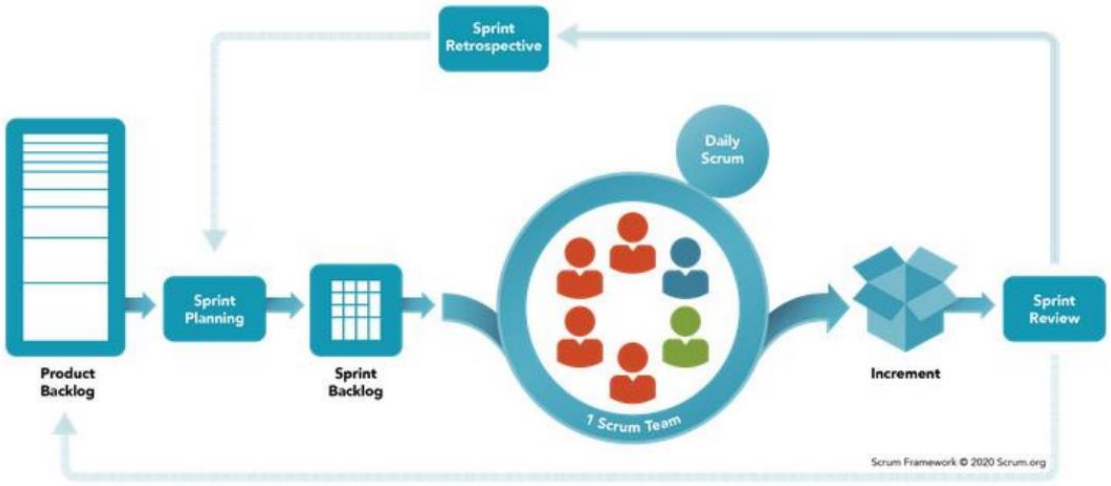

Figure 18. Scrum Framework [18]

The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team. Product Backlog items that the Scrum Team can do within one Sprint are deemed ready for selection in a Sprint Planning event. They usually acquire this degree of transparency after refining activities [19].

The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how). One of its main features is that it allows small team development where roles are well identified but can be exchanged. The main roles are the product owner, the scrum master, and the scrum team. In addition, an Increment is a concrete stepping stone toward the Product Goal.

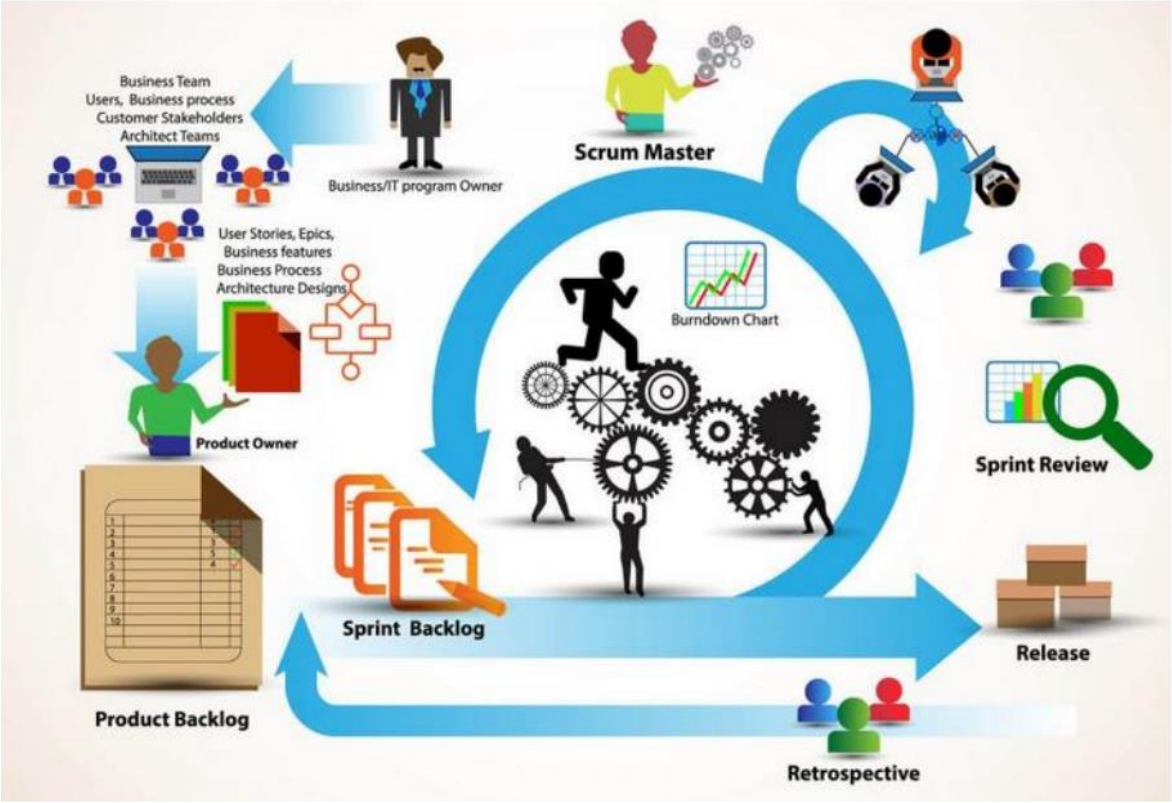

Figure 19. Scrum roles and their relationship

For this work's development, the Sprint duration was established as two weeks. At the end of each Sprint, the team members met during the Sprint Review with the tutors of this work.

The researcher performed all the roles defining the methodology in this research. According to the questions and answers on the reference website[11] of this methodology, it is possible if the person is a multi-skilled individual. Of course, the situation is not ideal mainly because the methodology was designed for a team, but the work can be done.

[11] www.scrum.org

3.2. Requirements

To develop this solution, the following functional and non-functional requirements have been defined, allowing the Interactive Virtual Aquarium to have the main characteristics found during the state-of-the-art study plus those proposed by the authors of this work.

3.2.1. Functional requirements

F01 - Fish behavior: Fish and algae should show movement as naturally as possible. The fish must avoid obstacles and each other according to their natural behavior, as far as possible, and present some behavior that indicates fleeing or chasing each other, simulating prey/predator behavior.

F02 - Interaction with the fish: The fish must react to stimuli from the user, such as feeding them or touching the screen of the terminals.

F03 - Didactic elements: The virtual aquarium must be able to display information about the fish selected.

F04 - Configuration: It must be possible to parameterize and configure the behavior of the fishes.

F05 - Remote Configuration: The aquarium must be able to be configured remotely.

F06 - Real-time synchronization: The virtual aquarium must show the fish synchronized in the different views of the aquarium in several terminals.

3.2.2. Non-functional requirements

NF01 - Target platform: The virtual aquarium should be presented as a web application and optionally as an Android application.

NF02 - Decorative elements: The virtual aquarium must have decorative elements of the marine environment.

NF03 - Target audiences: The virtual aquarium should have a simple interface and interaction with the fish so that children can use it.

Functional and non-functional requirements were defined, helping to focus on what was essential for the final solution.

3.3. Architecture

The solution consists of a virtual representation of a 3D aquarium as a web or android application loaded in a terminal, and the visual information displayed is synchronized between each one.

Note that the characteristics of this solution wrap around problems that are well known in the development of Video Games, such as the need to develop a 3D solution with user interactions that maintain synchronized information between them in real-time. Therefore, it was decided to use a video game engine, which led us to use a widely used solution in the video games industry, **Unity**.

A multiplayer game architecture with a dedicated server is proposed for this solution, unlike a P2P multiplayer architecture (client/server hosted by the player, Figure 20), where one of the clients oversees synchronizing and performing the calculations. For a client running a web app in a tablet browser, this comes at a high computational cost and having to deal with server migration if it goes down.

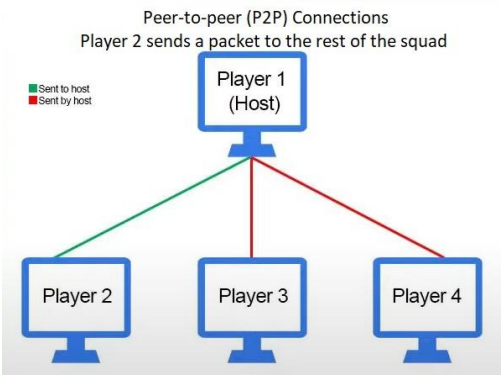

Figure 20. Peer-to-peer multiplayer architecture

This dedicated server architecture (Figure 21) allows us to solve these problems, control fish movements and interaction calculations in one place, and ensure that we always have a server for any representation of this aquarium we want. Of course, the server only represents the behavior of a single aquarium. However, in any case, we will have it available in a web browser from any device.

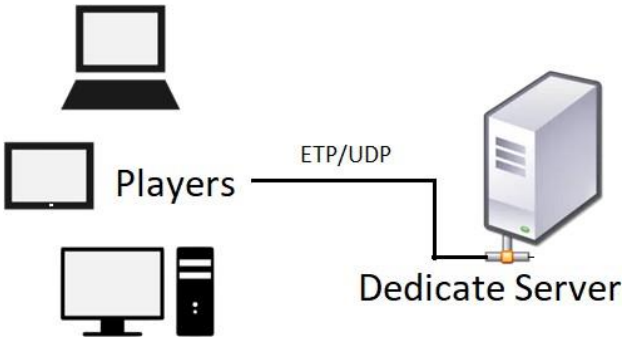

Figure 21. Multiplayer dedicated server architecture

We also need at least one solution to modify configuration parameters remotely. This will be very useful in modifying the behavior and species of the fish and the size of the schools of fish. In general, for any behavior that we want to parameterize (Figure 22).

Just as we need the remote configuration, we also need some way to facilitate the process of adding new models for the new species that we want to add. So, we need to download these models from our Web app at run-time from remote storage, which we can easily manage (Figure 22).

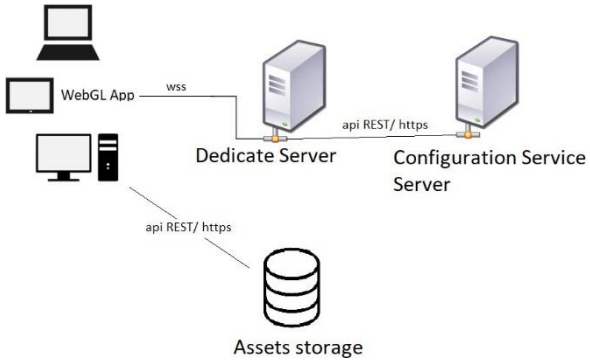

Figure 22. Multiplayer dedicated server architecture with configuration service

In summary, for this solution, a multiplayer game architecture with a dedicated server is proposed to guarantee the requirement of real-time synchronization between the different terminals, which represent a 3D virtual aquarium through a web or android application. In addition, a service is proposed through a REST API to manage remote configurations. Finally, it was proposed to place the 3d models of the fish in remote storage from which they can be loaded at run time as a fundamental step to facilitate the process of incorporating new fish, see (Figure 22).

3.4. Selection of technologies

This section describes the technologies selected for development according to the needs required by the proposed solution and the selected architecture.

3.4.1. Graphic engine

As the central technology for developing the solution, we use Unity. It allows us to compile our app for web platforms with WebGL 2.0 graphics API.

Unity is a cross-platform game engine developed by Unity Technologies. The engine has been gradually extended to support a variety of desktop, mobile, console, and virtual reality platforms. The engine can create three-dimensional (3D) and two-dimensional (2D) games, interactive simulations, and other experiences. In addition, industries have adopted the engine outside video gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces [20].

Unity engine offers a primary scripting API in C# using Mono for both the Unity editor in the form of plugins and games themselves, as well as drag-and-drop functionality. In addition, it supports building games for more than 19 platforms, including mobile, desktop, consoles, and virtual reality [20].

WebGL™ is a cross-platform, royalty-free open web standard for a low-level 3D graphics API based on OpenGL ES[12], exposed to ECMAScript[13] via the HTML5 Canvas element. WebGL is a Shader-based API using GLSL[14], with semantically similar constructs to the underlying OpenGL ES API. It stays very close to the OpenGL ES specification, with some concessions for developers' expectations from memory-managed languages such as JavaScript. WebGL 1.0 exposes the OpenGL ES 2.0 feature set; WebGL 2.0 exposes the OpenGL ES 3.0 API [21].

WebGL brings plugin-free 3D to the web, implemented right into the browser. Major browser vendors, Apple (Safari), Google (Chrome), Microsoft (Edge), and Mozilla (Firefox), are members of the WebGL Working Group.

3.4.2. Multiplayer networking solution

Since Unity was chosen as the graphics engine and considering the need to communicate in real-time with a web app, this leaves us only in the Unity environment using Web sockets as the transport layer. However, the idea is not to have to implement our transport protocol, so we use Fish-Networking (Fish-Net) of the available solutions. Fish-Net is a high-level networking library for Unity created by FirstGearGames, optimized for ease of use & probability of success. Some of the characteristics that led us to its use:

- Support for a dedicated server architecture.
- Compatible with over a dozen low-level Transports, including Web Sockets.
- A growing library of Script Templates to make learning and coding more accessible.
- It has Remote Procedure calls and context control via Attributes.
- It supports self-managed Linux server deployment.
- FirstGearGames grants a worldwide, non-exclusive, no-charge, and royalty-free license to reproduce, modify, and use the software for developing a game or other content. Several complete Examples are included.

[12] https://www.khronos.org/opengles
[13] https://www.ecma-international.org/publications-and-standards/standards/ecma-262
[14] https://www.khronos.org/opengl/wiki/Core Language (GLSL)

- Constant improvements and enhancements every month.
- Full-time support is available on their Discord platform.

3.4.3. Remote configuration solution

For a remote configuration service, we could choose whether to develop a solution that includes an API, that would store these configurations and deliver them to our server when they are requested and a portal to manage these configurations quickly. Alternatively, an existing solution that offers us these characteristics can be used. Luckily, Unity has a cloud service named ”Unity Remote Config,” which already has a REST API to integrate it into our Unity project.

As Unity claims, and to answer the needs for this project, this service allowed us to:

- Adapt an application to different types of users or tune the difficulty curve in real-time.
- Alter graphics quality based on device segmentation to optimize performance.
- Roll out new features gradually while monitoring impact.
- Tailor game settings to different geographical regions.
- Run tests, comparing the impact of different colors, styles, prices, copies, and others.
- Turn seasonal, holiday, or other time-sensitive events on or off remotely. Enable or disable features for specific player segments.

All that features are great, but in this case, it is used for its main feature, which allows us to change our configuration remotely, and its REST API can be used if we develop a simple portal to perform these actions.

3.4.4. Assets load solution

For uploading assets from remote storage, ”Unity Addressable” is used. The Addressable Asset System allows the developer to ask for an asset via its address. Once an asset is marked ”addressable,” it generates an address that can be called anywhere. Wherever the asset resides (local or remote), the system will locate it and its dependencies, then return it.

Addressable use asynchronous loading to support loading from any location with any collection of dependencies, providing a more straightforward way to make the app more dynamic.

3.4.5. Shader building tool

To develop the solution, we need to implement some shaders such as water surfaces, caustics, seaweeds, and fishes to add effects of objects within a volume of water. To facilitate this work, we use the "Unity Shader Graph."

Unity Shader Graph is a visual scripting tool used to create Shaders instead of writing code, ensuring the effects needed to seem like a natural aquarium in this case.

3.4.6. User interface

For the render of the user interface, "Unity UI Toolkit" allows the creation of the user interfaces in .uxml and .uss format, which are implementations of the .xml and .css standards by Unity. This feature allows us to have cleaner and more responsive interfaces and much flexibility to create them using the "Unity UI Builder" tool.

3.4.7. 3D Modeling

For the modeling of the fish and algae in 3D, Blender 3.2.2 was used. Blender[15] is a free and open-source 3D creation suite. It supports the entirety of the 3D pipeline modeling, rigging, animation, simulation, rendering, compositing and motion tracking, and even video editing and game creation [22]. Another fundamental step to finishing the 3D models is the work with textures. To make them, the Gimp 2.10[16] image editor was used.

[15] https://www.blender.org
[16] https://www.gimp.org

3.4.8. Testing

As support during development to measure and perform performance tests, the Unity Profiler tool is used, which allows us to have a detailed view of how many resources have been used and the execution time during each frame.

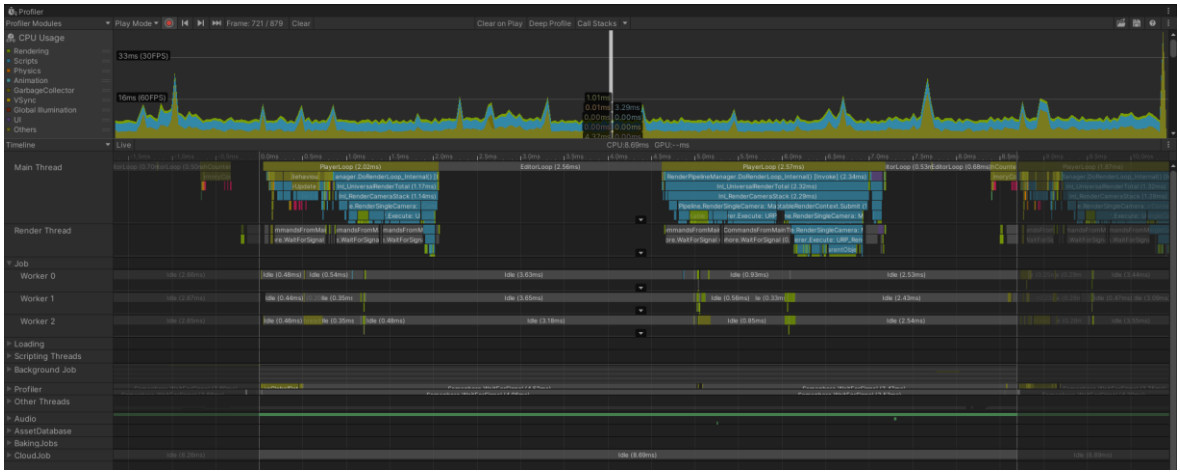

Figure 23. Unity Profiler

3.4.9. Deployment

As a base technology for the deployment, Docker[17] is used to creating the images that will contain the server and the static host for the web app. Docker is an open platform for developing, shipping, and running applications. Docker enables separate the applications from the infrastructure so it can deliver software quickly. With Docker, the infrastructure can be managed in the same ways that manage the applications. Taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, thus can significantly reduce the delay between writing code and running it in production.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow running many containers simultaneously on a given host. In addition, containers are lightweight and contain

[17] https://docs.docker.com

everything needed to run the application, so is not needed to rely on what is currently installed on the host [23].

3.5. Chapter Summary

As a summary of the methodology phase, we can summarize the following:

- Based on the characteristics of the solution, a methodology was used that allowed an agile process to be carried out, where the iterations are focused on the delivery of functional requirements.
- The selected and implemented architecture based on multiplayer dedicated server architecture allows us to have control of the aquarium and makes it possible to have clients released for various platforms.
- All the technologies selected for the development of this solution were chosen according to the needs of our architecture and design, sitting mainly around the Unity video game development platform.

The following chapter will address the development process to create the Interactive Virtual Aquarium, Digital Fishes.

4. Development

The goal of this chapter is to present the work that has been done in the implementation phase. The chapter begins by describing the graphic design of the solution, referring to the user interface, color selection, and 3d models of fish and algae. Finally, the programming process is explained to obtain the virtual aquarium representing Portuguese marine life.

4.1. Digital Fishes UI Design

The solution consists of a virtual aquarium with a physical representation made with a framework that supports ideally 4 Tablets (Figure 24). For each Tablet, a web or android application with a virtual representation of a 3D aquarium is loaded. The visual information displayed is synchronized between each terminal (Tablet), showing a face of the aquarium in each of them to obtain a better user experience and immersion.

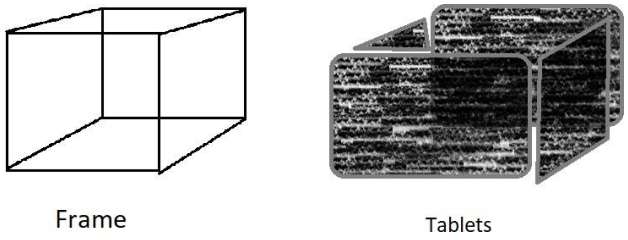

Figure 24. Prototype physical representation design

Due to the limited devices available(3), a physical prototype (stand) for the tablets that fit three devices was built. For this reason, so opted to lean against the wall and cover it from above (like some natural aquariums).

Figure 25 shows an example of the Interactive Virtual Aquarium assembled in the built physical prototype.

Figure 25. Prototype physical representation

In the case of the user interface, it is not necessary to be very complex so that graphic elements are avoided as much as possible to maximize the area of interaction with the fish and maximize immersion. Therefore, it only needs a waiting screen with a start button and a screen on which the fish will be displayed to interact. A way to obtain information on each species was also requested. It is then to opt for a second mentioned screen (main screen) with a button to return to the waiting screen, a button to feed the fish, navigation buttons that change the perspective of the aquarium, and an information panel that remains hidden until a fish is selected (Figure 26).

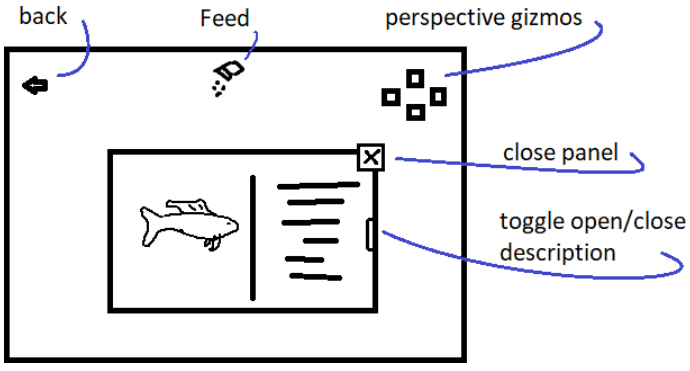

Figure 26. Main screen

4.1.1. Colors

Great color palettes can set the right mood and motivate visitors to take action. However, designing something new is often challenging to decide on the color scheme because there are many possible color combinations. In its article [11], Munro covers what to consider when choosing a color palette.

For the virtual aquarium, we thought of a flat design with a color palette that denotes cleanliness and is friendly. A palette based on the colors yellow, black, and white was chosen.

Figure 27. Color palette chosen

4.1.2. Layouts

The result obtained as a User Interface can be seen in the following figures, where the layout of the screens described above is shown. Figure 28 shows the standby screen layout, and Figure 29 and Figure 30 show the layout of the main screen. The layout of the selected fish Information Panel is shown in Figure 30.

Figure 28. Digital Fishes Interactive Virtual Aquarium, Waiting Screen layout

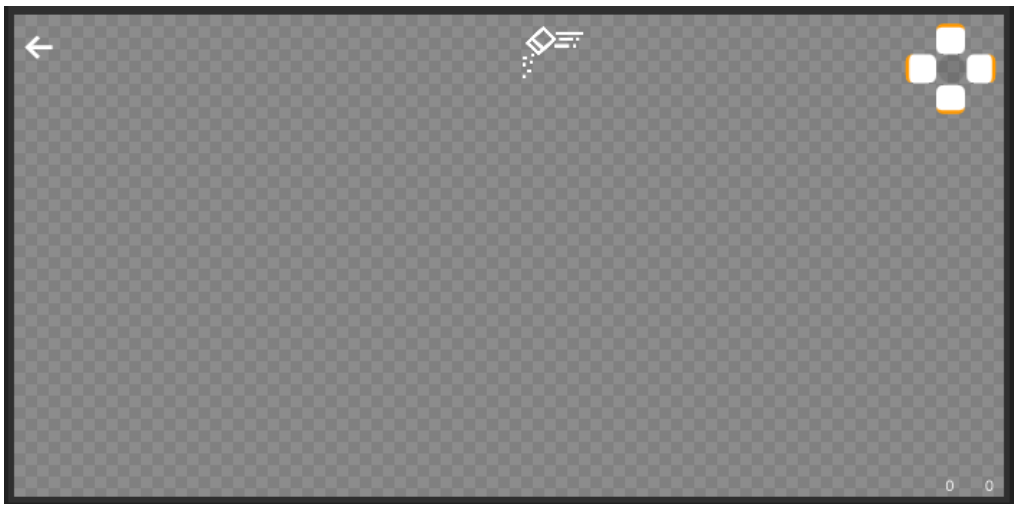

Figure 29. Digital Fishes Interactive Virtual Aquarium, Main Screen layout

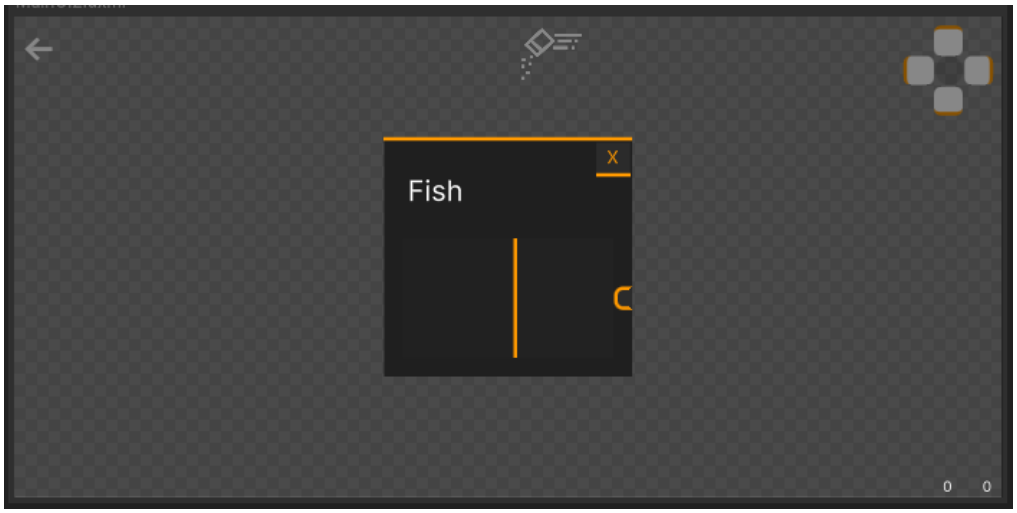

Figure 30. Digital Fishes Interactive Virtual Aquarium, Main Screen layout with fish details

4.2. 3D Models Design

Fish have bilateral symmetry that is relatively simple to model in 3D. For the modeling, lateral images of fish were taken as references. The generated models are low poly to make them lighter for rendering. In the final result, the fishes are scaled, using the largest to establish a proportion as a reference.

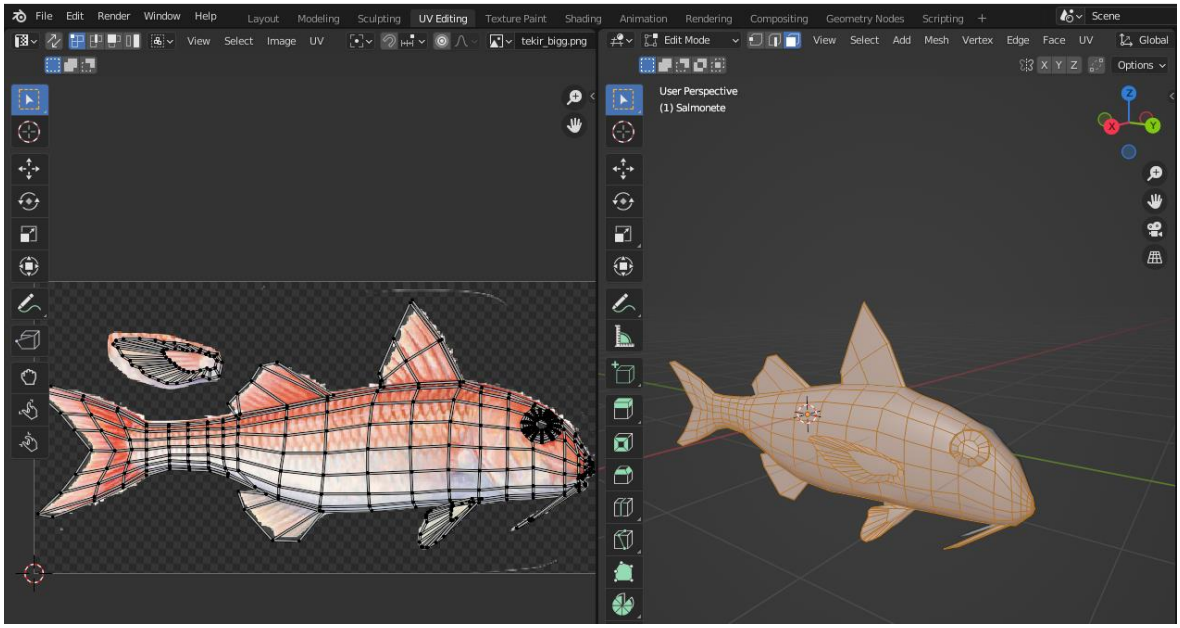

Figure 31. The model and UV mapping process

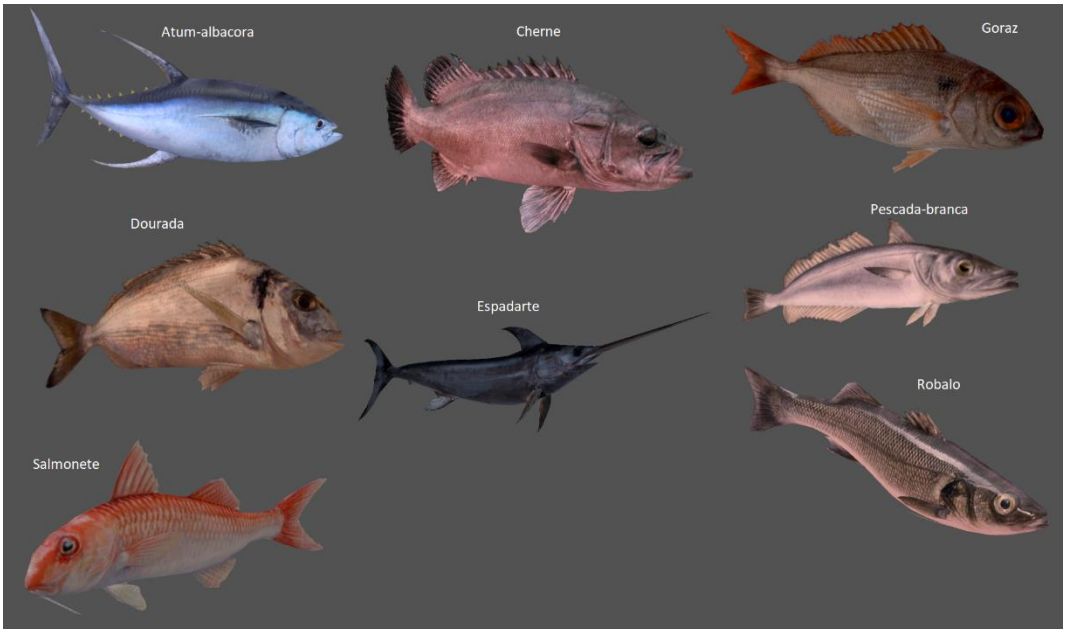

Figure 32. Fish models from Portuguese waters created

Interlocking planes were used to model the algae with a texture with a transparency layer (Figure 34). The textures for the algae were generated from base images and divided into three fundamental images for their use in the algae materials to achieve a better visual effect.

- The first image contains the base color and the transparency channel (Figure 33, A).
- The second image is edited so that each color channel contains different information about the material, the red channel, the metallic property, the green channel, the smoothness property, and the alpha channel, the ambient occlusion (Figure 33, B).
- The third image is a Normal Map texture (Figure 33, C).

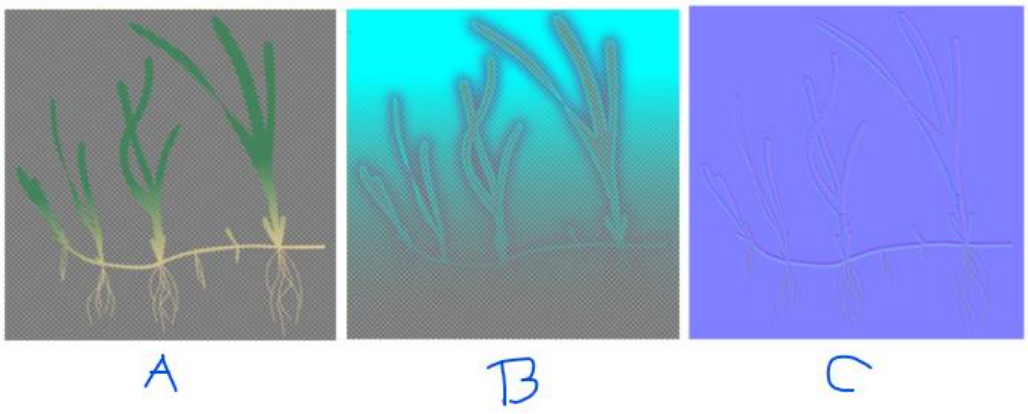

Figure 33. Base textures for algae

A: Albedo texture; B: MAODS texture; C: Normal texture

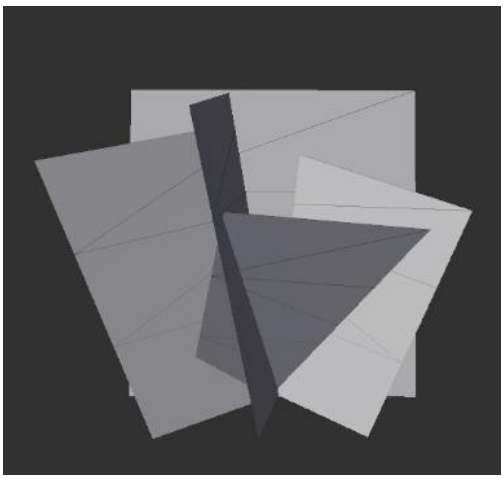

Figure 34. The base model of a seaweed

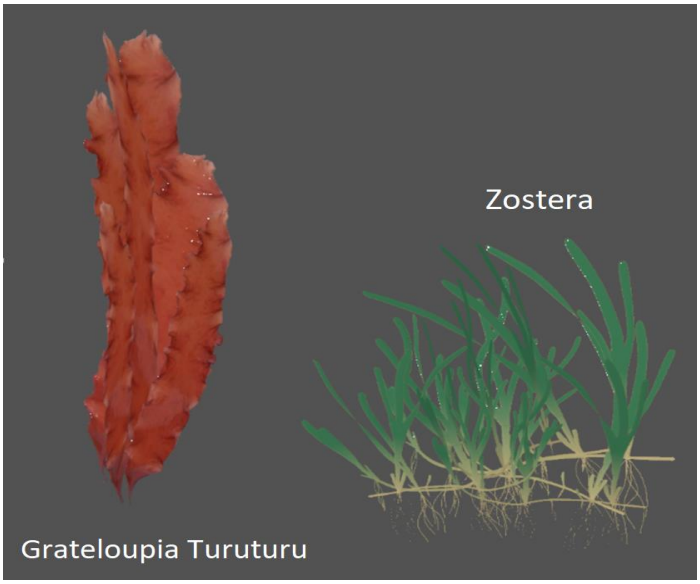

Figure 35. Models of seaweed found in Portuguese waters created

4.3.Fish behavior

For the implementation of fish behavior, we use the Boid algorithm created by Craig W. Reynold [24], which allows us to use simple rules to define the movement in groups of fish.

4.3.1. Boids

The abbreviation ”Boid” is an abbreviated version of the term ”bird-oid object” (”bird-like object”). In the natural world, schools of fish or flocks of small birds move in this way.

A Boid group is a group of objects that move smoothly. According to the research in [24], group movement is the result of the actions of individual objects, acting solely based on their local perception of the world. A flock is thus a formation formed as a result of the interaction between the behavior of individual birds. Therefore, a flock movement simulation can be developed by creating models of several birds and introducing additional interactions between them [25].

In his paper [24], Reynolds describes three levels of control used to create "autonomous characters" and simulate their movements within a flock of birds or a school of fish. These techniques have been used in countless movies, animations, and video games. The levels of control can be split like this:

- Action selection (highest): involve strategies, goal selection, and planning.
- Steering (middle): path determination.
- Locomotion (lowest): involves the underlying animation mechanics.

By its weight in the solution, we start describing the middle (Steering) layer.

4.3.2. Steering

The basic model of a herd of swarming objects consists of three simple behaviors that govern their movement. Here [26], Reynolds describes how individual Boid maneuvers are carried out based on the location and speed of other objects:

- Separation (Figure 36) - it is responsible for avoiding crowds of local members of the herd, thanks to which there is no collision with nearby objects;
- Alignment (Figure 36) - requires movement to take place about the average position of the local herd members;
- Coherence (Figure 36) - requires objects to move toward the average position of local herd members, leading to an attempt to stay close to neighboring herd members.

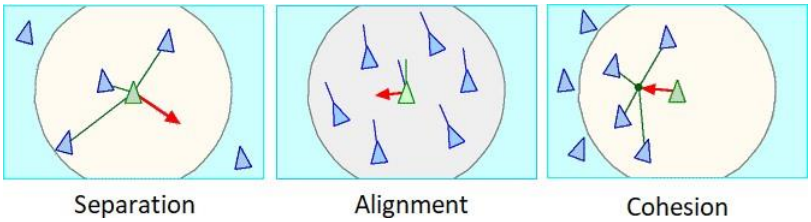

Figure 36. Steering [26]

Each Boid has direct access to the full geometric description of the scene. However, participation in the herd causes the object to respond only to members from a limited area around it (Figure 37). The neighborhood is described by the distance (measured from the

center of a single Boid) and the angle of the direction of movement of the Boid. Herd members outside this neighborhood are ignored [26].

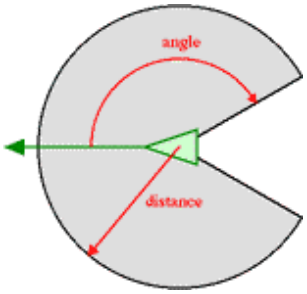

Figure 37. Boid's neighborhood [26]

Our implementation of the steering behavior is based on the pseudo-code proposed by [25] with the necessary modifications to improve performance in Unity with hundreds of Boids.

We can say that the implementation of the forces that affect the motion of an object can be achieved using vector notation. So, our Boid is composed mainly of a Velocity vector and its Position in space (Figure 38).

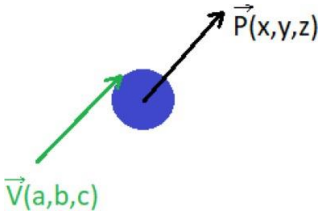

Figure 38. Boid vectors

At the most basic level, a Steering Behavior is a function that returns a Vector (could be 2D or 3D) with the direction the Boid should move to on the next frame. Every update, all Boids have their next Velocity and Position, calculated based on whatever list of behaviors we have decided to use.

We can add as many behaviors as needed to get the movement we want since there is some simple vector math (v1 + v2 + v3 + vN). The key to making the school of fish work is combining Cohesion, Separation, and Alignment. The more steering behaviors are defined and used, the more complex and ”realistic” the overall effect.

Starting by describing the SteerTowards action (Figure 39), according to Reynolds [26], the steering vector is the difference between the desired velocity and the Boid's current velocity. The desired velocity is a vector in the direction from the Boid to the target. The length of the desired velocity could be max_peed, or it could be the character's current speed, depending on the application.

```
1
2   desired_velocity = normalize (targetPosition - position) * max_speed
3   steering = desired_velocity - velocity
```

Figure 39. Steer Towards action

To calculate the behavior, we must iterate between all the Boids and calculate for each one Cohesion, Separation, and Alignment.

Steering for cohesion can be computed by finding all Boids in the local neighborhood and computing the average position of the nearby Boids. Then, the steering force can be applied in the direction of that average position (subtracting our Boid's position from the average position).

Steering for alignment like cohesion, but instead of averaging the positions of the other Boids, we average their velocities. We calculate a 'perceived velocity,' then add a small portion to the Boid's current velocity to steer it in the same direction as its neighbors.

To compute steering for separation, a repulsive force is calculated by subtracting the positions of our Boid and its neighbors, normalizing, and then applying a 1/r2 weighting. (the position offset vector is scaled by 1/r2). Finally, these repulsive forces for each nearby Boid are summed together to produce the overall steering force.

As we can see from the method for the calculation described by Reynolds [24] and the implementation proposed by [25], we go through the Boids, filtering those that are in the neighborhood in the perception angle. Then, make an average of the position and velocity for the cohesion and alignment cases and an algebraic sum to calculate the steering separation repulsive force. Therefore, we split the implementation into two essential parts: The first contains only the calculation of the number of visible neighbors, the average

position, and the average velocity, in addition to the calculation of the alignment, all in a single iteration.

In the second part, we carry out the final steps of the proposed algorithm: calculating the three force vectors, just applying the SteerTowards action to the average of the alignment, the difference between the average of the cohesion and the current position, and the separation. Finally, the ”desired velocity” vector is obtained with the sum of these new force vectors.

The decision to separate it this way is because the first part is the most expensive. By separating it, we can experiment with different solutions discussed later, leaving only the second part responsible for updating the position and direction, making only the most straightforward calculations with the data found in the first part.

Now to add the obstacle avoidance behavior. The idea is to project rays with increasing angles until finding a free path or the farthest free path (Figure 40). Since we are moving in 3 dimensions for it to work, we must project the rays to points on a sphere.

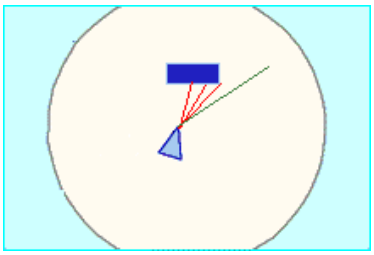

Figure 40. Avoiding obstacles

To generate the points, we use the golden spiral method on a sphere described by CR Drost [27], which uses the golden ratio to distribute points evenly on the sphere (Figure 41).

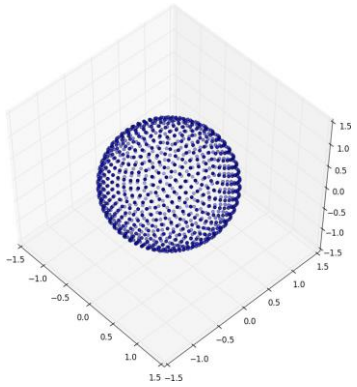

Figure 41. Evenly distributing n points on a sphere

To add the obstacle avoidance behavior after obtaining the direction vector, we must verify if, with this new direction vector, the Boid is directed to collide with an object that is in its perception line. If so, we need to add a repulsion force vector obtained by applying the SteerTowards action with the new free path direction calculated using CR Drost [27] method.

This method is implemented simply by iterating through the points (directions) on the edge of the generated sphere. Then projecting, a ray to detect a possible collision in that direction. Finally, returning to the first free direction or the direction with the most distant free path (Figure 42).

The computation to detect collisions is done with the *SphereCast* function from the Unity Physics package. Note that the distance at which the ray (like a sphere) is projected is the Boid's radius of perception.

Now adding the behavior of the Boids to our fishes, we already have the movement in schools. It only remains to add other behaviors to get closer to reality and better immersion in our virtual aquarium.

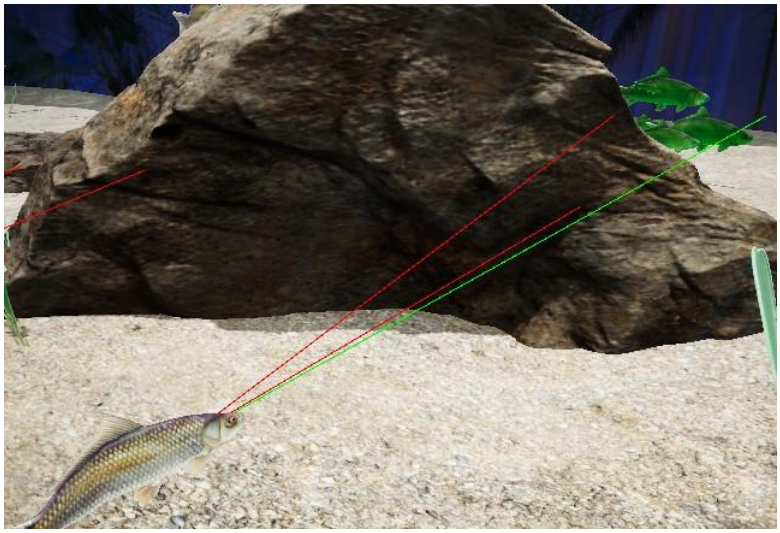

Figure 42. Obstacle avoidance result

4.3.3. Food chain behavior

To add food chain behavior, we separate the fishes by species and assign them an ascending level, with the lowest level being the weakest.
To implement the food chain behavior, we introduce the changes in the Steering Behavior to calculate the target of the predators and the direction of the prey's flight.

Now we have different species, so we apply the cohesion and alignment vectors calculation only to the same species and the separation vector for all of them. Thus, we avoid weighing them up (Figure 43).

Figure 43. The separation between different species result

A simple solution for the target is to assign it the position of the last fish one level lower than the perception radius. Later we can chase the fish, calculating the direction from the fish to the target with the difference between the target and the position of the fish (Figure 44).

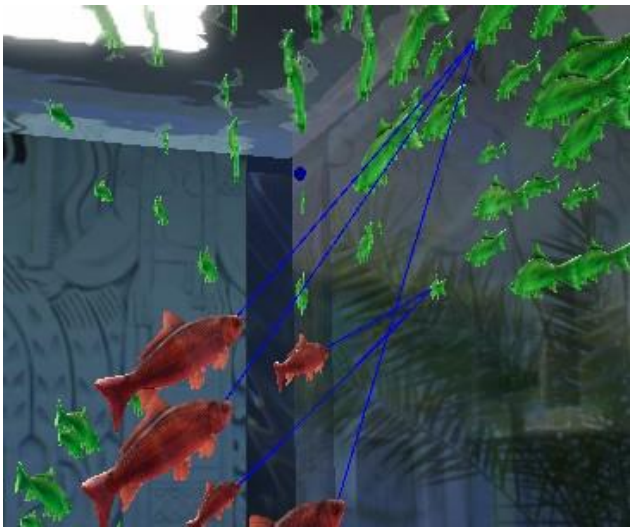

Figure 44. Predators' behavior results

In the case of prey, we calculated a dispersion force decreasing the difference between the predator's and fish's positions. The predator would be all the fishes in the radius of perception of the prey with the highest level. It will result in a dispersion force vector opposite to the predators. In this case, we only calculate the cohesion and alignment when the fish does not have to flee to achieve the effect of dispersion of a school (Figure 45).

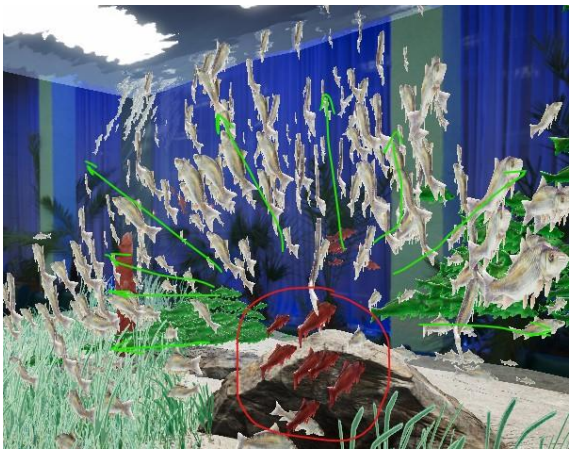

Figure 45. Prey dispersion behavior results

4.3.4. Lonely behavior

For the case where the fish is alone, we calculate a lonely behavior by reusing the same idea of choosing the list of possible addresses in the sphere [27] and selecting one randomly. Another solution that could be used is a Perlin noise to generate the possible addresses, as mentioned here [28]. This calculation is executed in intervals of several seconds to make smooth movements and keep the fish in the same direction for a while.

4.3.5. Action selection

To interact with our aquarium was implemented goals tracking strategies, which are:

- The fish follow the movement of the pointer or finger (Figure 46).
- The fish move toward the food when fed (Figure 47).

For this implementation, we steer towards a target object to obtain the vectors in the direction of the objective, then add that vector to the rest of the behavior to guarantee separation, cohesion, and alignment.

To **follow the finger or the pointer**, they are set as a target object while the user moves them through the screen.

Figure 46. Pointer target results

In the case of **following the food**, as we use a particle system for the render, we calculate the midpoint of the particles and use it as a target. Thus, the midpoint is recalculated while the particles disappear until none are left.

Figure 47. Food target results

4.3.6. Locomotion

The locomotion is the lowest level and only involves the underlying animation mechanics. In this case, the fish's swimming is carried out using the vertex displacement techniques in the shaders, which can be seen in the next section.

4.3.7. Performance optimization

Note that the straightforward implementation of the Boids algorithm has an asymptotic complexity of $O(n^2)$. This is because the Boid needs to consider all the others to determine if it is not a nearby flock mate [26].

As Reynolds [26] suggests, it is possible to reduce this cost to almost O(n) by using a proper spatial data structure that allows us to keep the Boids ordered by their location. There is no need to iterate through all the Boids to find the neighbors.

A data structure that can be used for this purpose is a Multi-HashMap, with which we can apply the spatial hashing technique. It consists of using as an index a hash obtained from the object's location in the space subdivided into cells, see about in [29]. The objects are stored

in such a way that when we query for the hash of the location, we get the list of objects in the cell in which it is located.

Unity has a collection type called *NativeParallelMultiHashMap* which we use to store the positions of our Boids and thus quickly get the list of neighbors.

The key here is to generate the hash. For this, we use the function proposed in [30], which can be translated to the code represented in (Figure 48).

```
1
2   hash = (Floor(pos.x / cellSize) * 73856093) ^ (Floor(pos.y / cellSize) * 19349663) ^ (Floor(pos.z / cellSize) * 83492791);
```

Figure 48. Spatial hash derived from [11]

For the Floor function, we apply the one suggested here [31], to gain calculation speed, see the result (Figure 49).

Figure 49. The spatial hash result with active cells highlighted

To further optimize the solution, we can implement the calculation of the Boids using multi-threading.

Here we have investigated two variants, one of which has been finally adopted.

The first variant is to use a Compute shader[18]. Compute shaders are programs that run on the graphics card outside the rendering pipeline. They can be used for massively parallel GPGPU[19] algorithms or to accelerate parts of game rendering.

This method proves to be faster than using multi hash calculation. However, in this case, as the shader uses the GPU, it is inefficient in our solution, which needs to do the calculation on the server in a build optimized for a dedicated server that does not need asset as audios, textures, and shaders. In addition, this would limit us to using more resources in our deployment.

The second variant is using Unity Data-Oriented Technology Stack (DOTS), which will make it possible to take full advantage of today's multi-core processors. This new Unity architecture will lead the future developments on this platform. They say they are rewriting the Unity core based on DOTS. This work uses Unity C# Job System and Unity Burst Compiler packages from the DOTS stack.

The C# Job System takes advantage of the multiple cores. In addition, C# Job System exposes Unity's internal C++ Job System, allowing C# scripts to run as jobs alongside Unity's internal components. It also protects against some of the multi-threading pitfalls, such as race conditions.

Burst Compiler is a new LLVM-based[20] backend compiler technology that takes C# jobs and produces highly optimized machine code. In addition, the Burst Compiler optimizes the output for the compiling platform.

The characteristics of these latest technologies are ideally suited to the demands of our solutions:

- A solution to the calculation problem using multi-threads, in addition to integration with the spatial hashing method, exploiting the data structures it offers us, increasing, even more, the performance.

[18] https://docs.unity3d.com/Manual/class-ComputeShader.html
[19] https://pt.wikipedia.org/wiki/GPGPU
[20] https://llvm.org

- Build optimization for a dedicated server using Burst.

To measure the proposed solutions' performance, we tested the different implementations for an extreme scenario where the Boids are all together chasing a goal and meeting in the corner of the tank.

Let us note that in the methods tested during this first iteration of improvements, the weight of the computation falls on the detection of collisions. If we see the last case (Figure 50, yellow line) in which we use the Unity Physics[21] package, which allows us to perform the calculation in multi-threads, the time is significantly reduced.

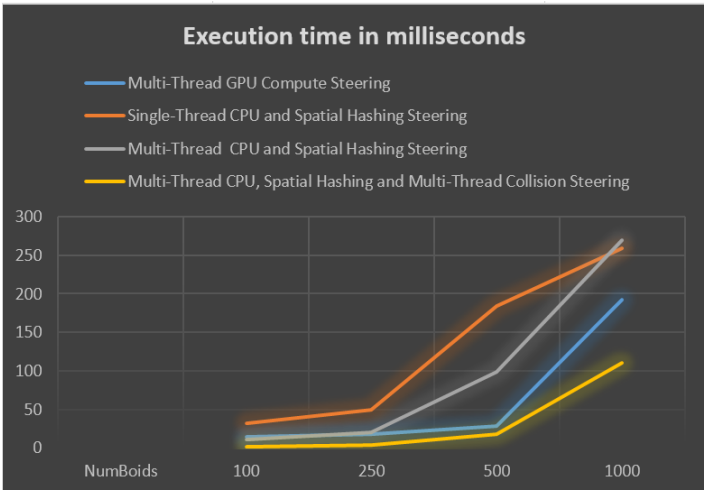

Figure 50. Execution time in milliseconds of the movement of the Boids stressed with collisions

At this point, the calculation has been significantly optimized. However, it is still not ideal because if we add the synchronization cost to an extreme scenario like the ones tested previously, a case in which the performance is drastically reduced.

We have realized that the physics calculation to detect collisions is where the weight lies at the moment in calculating the behavior of the Boids. To mitigate this, we use a technique similar to that suggested by Müller and colleagues [30]. In that paper, they describe a new algorithm for detecting collisions and self–collisions of deformable objects based on spatial

[21] https://unity.com/dots/packages#unity-physics-preview

hashing. The algorithm classifies all object primitives, i. e., vertices, and tetrahedrons, with respect to small axis–aligned bounding boxes AABB. Therefore, a hash function maps the 3D boxes (cells) to a 1D hash table index. As a result, each hash table index contains a small number of object primitives that must be checked against each other for the intersection. The actual collision detection test computes barycentric coordinates of a vertex with respect to a penetrated tetrahedron.

Considering that the objects with which the Boids interact are static (the pond boundaries and the obstacles), the physical space for the static objects was partitioned using the same spatial partitioning technique mentioned above.

The objects stored in the partitions correspond to the triangles that make up the mesh of a 3D object. These are previously extracted and stored as an asset of the application. In addition, an extension has been made for the editor with which this task is carried out with just one button (Figure 51).

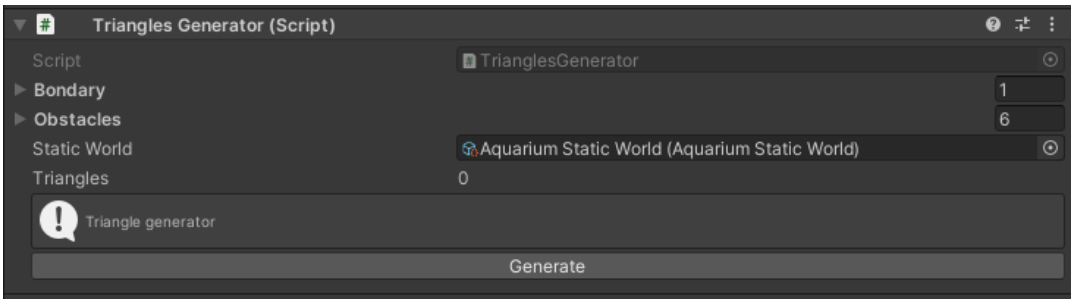

Figure 51. Triangles generator

Having already the triangles with their normals positioned in a partitioned way, what remains is the calculation to detect the collisions. Applying geometry, we can calculate the intersection of a line with a triangle[22]. If these intersect, there is a collision in the direction described by that line. Otherwise, we can calculate the distance between the triangle and the line's endpoint to avoid passing close and sticking to the object. The space cell where the point at the end of the guideline must be consulted to apply this algorithm (Figure 52).

[22] https://mathworld.wolfram.com/Line-PlaneIntersection.html

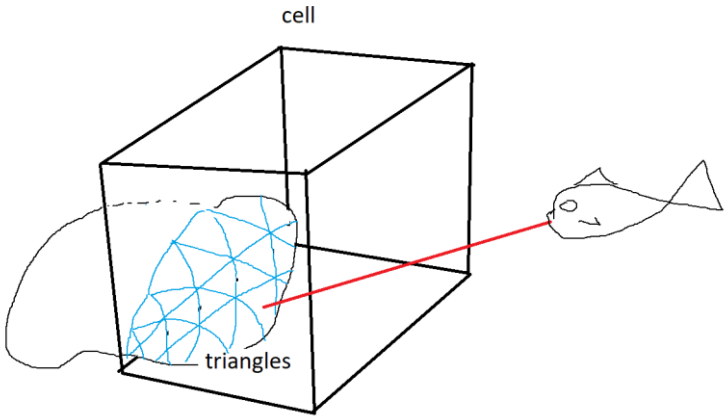

Figure 52. Application of collision detection with spatial partitioning

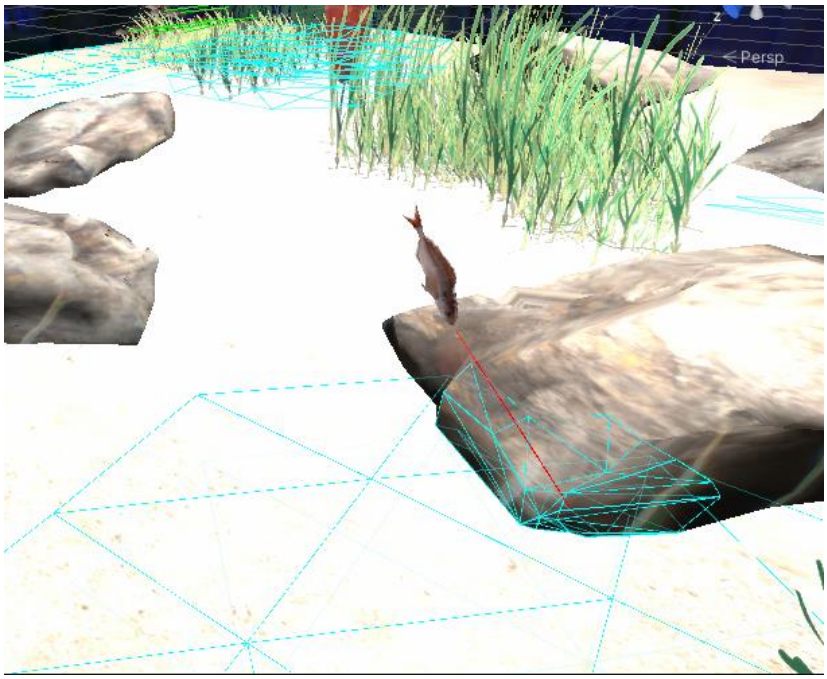

Figure 53. Collision detection with spatial partitioning

When testing the previous scenarios again, we can appreciate the improvement in the speed of calculating the behavior of the Boid (dark blue line in Figure 54).

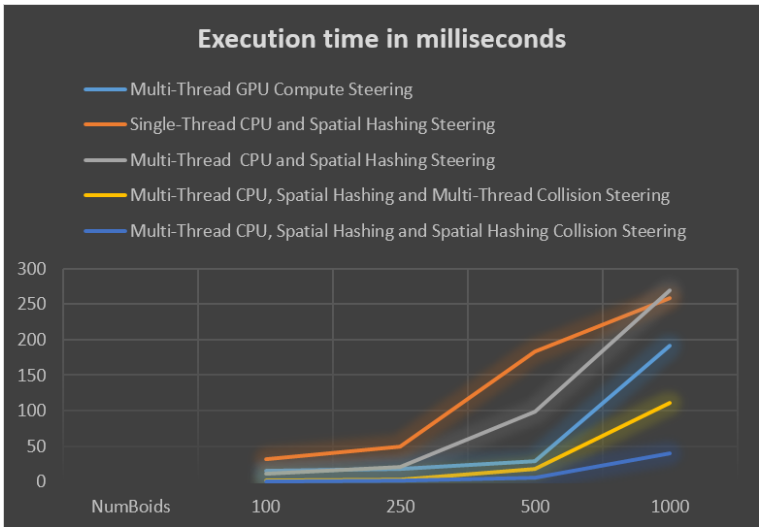

Figure 54. Adding spatial partitioning and collision detection test result

4.4.Shaders

A shader is a small piece of program, written in a low-level language that looks like C, compiled into a native binary executed by the GPU. Two languages are used today GLSL (for OpenGL Shader Language) and HLSL (for High-Level Shader Language used by DirectX). As mentioned above, the shaders for this solution are built using the Unity Shader Graph tool [32].

The shaders created during the development process mentioned below inherit the behavior of the base shaders used in URP Lit. Therefore, the correct operation of the lights and shadows that the Unity graphics engine provides is guaranteed.

4.4.1. Fish Animation

As we mentioned before, the locomotion layer proposed by Reynolds [24], was implemented only using shaders.

The proposed solution consists of simulating the simple swimming movement of a fish by moving the vertices of the model using a sine function through its axes depending on how we have the model oriented (Figure 55).

The solution was inspired by the article [33] by Bitshift Programmer, in which he mentions something that we were able to verify regarding the performance of this type of solution, using an animation if we consider that the GPU calculates the shaders.

In our case, the orientation of our models is thus the Z axis (Figure 55), for which we will displace the vertex in X along the Z axis of the model using a sine function to simulate the undulating movement:

$$x = \sin(z)$$

To control the animation in terms of the amplitude, frequency, and speed of the swimming movement, we do the following:

vertex.x += sin(vertex.z * FrequencyZ + Time * SpeedZ + MoveOffset) * AmplitudeZ

Note that we introduced the time variable, this is necessary to guarantee continuous movement and an offset to later be able to generate a different movement for each fish.

We just need to limit the amount of movement through the model. To map the movement, we use a gradient from the limit of the head to the tail (Figure 55). And to calculate this displacement through a gradient, we apply a linear interpolation[23] (lerp) between the vertices by a Sigmoid function[24] (smoothstep) of the limit of the head:

Sinx = vertex.x + sin(vertex.z * FrequencyZ + Time * SpeedZ + MoveOffset) * AmplitudeZ

vertex.x = lerp(vertex.x, Sinx, smoothstep(HeadLimit, HeadThreshold, vertex.z))

[23] https://en.wikipedia.org/wiki/Linear interpolation
[24] https://en.wikipedia.org/wiki/Sigmoid function

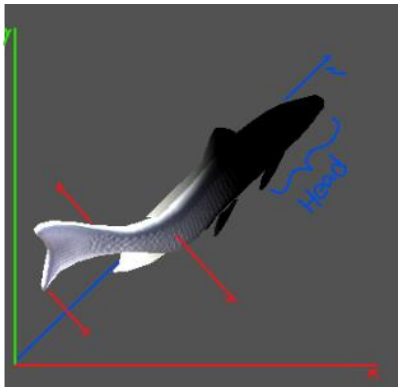

Figure 55. Fish vertex displacement idea

4.4.2. Water surface

We are based on the idea used by Ilett [21] to create a water surface, which is divided into components to achieve a simple water surface shader. We need displacement vertically for low-frequency wave motion with high amplitude, details of high-frequency waves, and transparency with a distortion that gives it a touch of depth.

1. Displacement vertically for low-frequency wave motion with high amplitude: For the movement of low-frequency waves, we also use the vertex displacement technique, but in this case, we do it based on a height map generated by a Gradient Noise (Figure 56, A).
2. Details of high-frequency waves: For distortion and ripples, we also use a noise gradient transformed into a normal map (Figure 56, C), but the gradient with another scale achieves high-frequency wave details. We can combine several layers by shifting in different directions to achieve a slightly more chaotic effect.
3. Gradient Noise maps Finally the transparency we use a **Shader Graph - Scene Color** node to obtain transparency [34].

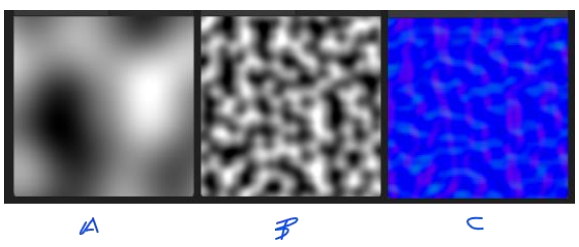

Figure 56. Gradient Noise maps

The movement is achieved by displacing the maps (height, normal) as a function of time in such a way that when a point of the map passes through a vertex, it is displaced by the color scale input (black and white) (Figure 57).

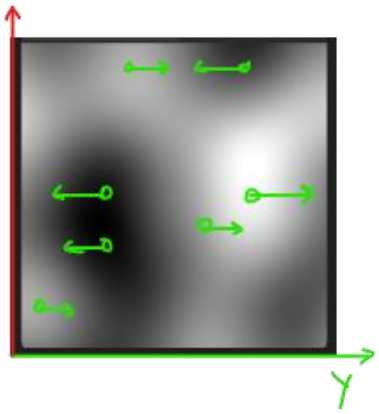

Figure 57. Vertex displacement map

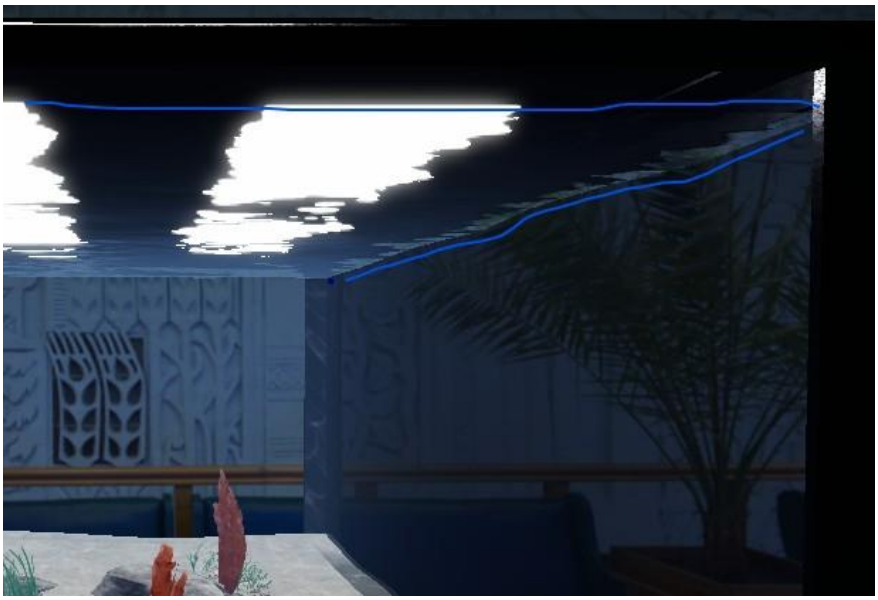

Figure 58. Water surface result

4.4.3. Seaweed

The seaweed shader is implemented with the same idea of vertex displacement based on a gradient noise map (Figure 57). In this case, the displacement is horizontal, based on the

vertex's position in world space. In this way, an organic animation effect of all the seaweed is achieved (Figure 59).

The same idea to limit the movement of the fish from the head is used to anchor the movement to the root of the plant.

Figure 59. Seaweed

4.4.4. Caustics

Unity Decal Projector was used for the caustics to project a texture onto all intersecting models. Note that this projector only correctly projects the texture on one axis. On the others, it stretches it (Figure 61: A). To correct this, we decompose the absolute position in the world space and add each of its components (Figure 60). It can also be added distortion to the texture and displacement, combining several layers in the same way as the one used in the shader for the water surface (Figure 61: B).

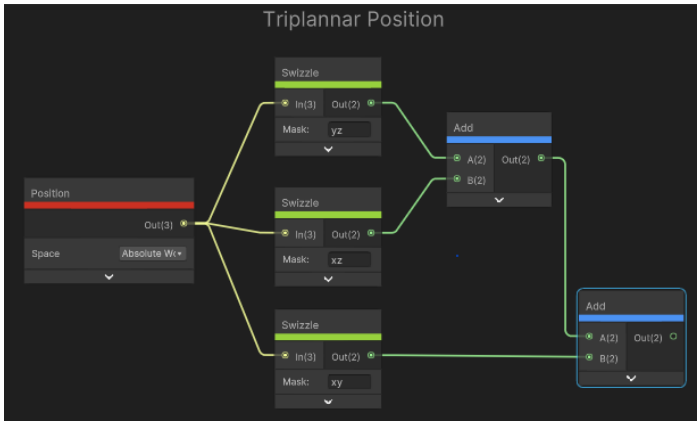

Figure 60. Triplanar decal projection

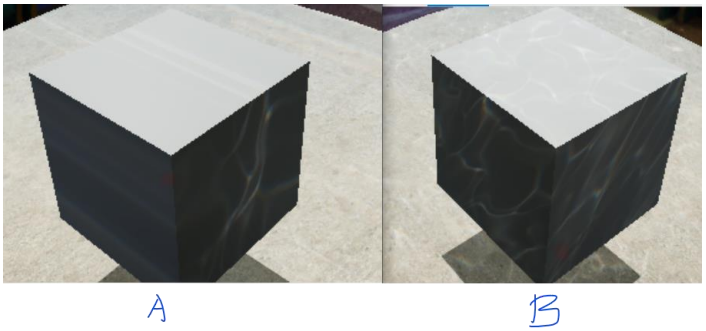

Figure 61. Caustics projection

A: projection on one axis, B: projection on three axis

4.5.Remote Configurations Management

As briefly described in the technologies selection section, we use the services offered by Unity to manage the configurations of the Virtual Interactive Aquarium.

The solution can update the configuration without stopping the server using a control variable where the version of the current configuration is established. However, for the change to take effect, the version of the new configuration must be updated. Figure 62 shows a snapshot of the fish setup parameterization.

```
[{
    "id": "1",
    "assetId": "Salmonete",
    "active": true,
    "amount": 100,
    "fishName": "Salmonete",
    "information": "Mullus surmuletus O SALMONETE é um peixe ...",
    "boidSettings": {
        "minSpeed": 2,
        "maxSpeed": 5,
        "perceptionRadius": 5,
        "avoidanceRadius": 1,
        "maxSteerForce": 3,
        "alignWeight": 3,
        "cohesionWeight": 1,
        "seperateWeight": 1.5,
        "disperseWeight": 3,
        "species": 1,
        "level": 0
    }
}, {
    "id": "2",
    "assetId": "Merluccius",
    "active": true,
    "amount": 10,
    "fishName": "Pescada-branca",
    "information": "Merluccius merluccius A PESCADA-BRANCA tem um corpo ...",
    "boidSettings": {
        "minSpeed": 2,
        "maxSpeed": 5,
        "perceptionRadius": 5,
        "avoidanceRadius": 1,
        "maxSteerForce": 3,
        "alignWeight": 3,
        "cohesionWeight": 1,
        "seperateWeight": 1.5,
        "disperseWeight": 3,
        "species": 2,
        "level": 0
    }
}]
```


Figure 62. Snapshot of the parameterization for the configuration of the fish

Below is a screenshot of the service to manage remote configurations in Unity Gaming Services (Figure 63). It shows the list of configurations by environments that are available for the Digital Fishes Interactive Virtual Aquarium.

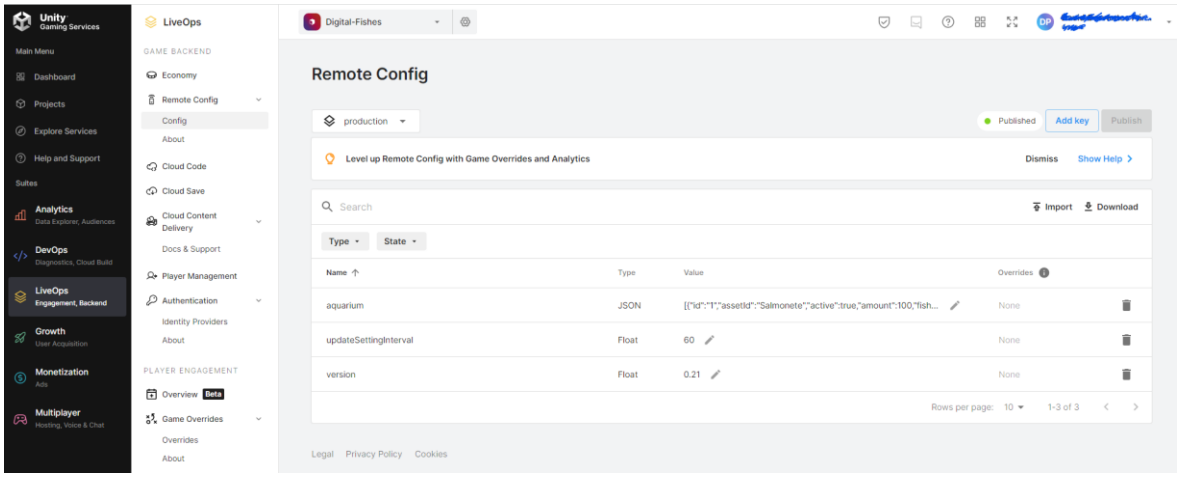

Figure 63. Unity Remote Config Portal

4.6.Fish Asset Management

As previously described during the technologies selection section, the remote loading of the fish models has been developed using the "Unity Addressable" service.

Client applications only download these templates for the first time at run-time only once. In this way, the size of the compiled application is also reduced, in addition to meeting the requirement needed to make the process of incorporating new fish species more flexible.

Note that the Unity editor must package these models as a **Prefab**. A Prefab allows an object's creation, configuration, and storage with all its components, property values, and child objects as a reusable asset. In addition, the Prefab acts as a template from which it can create new Prefab instances.

In the fish Prefab, a collider must be added to receive the touches on it and scale it properly to be proportional to the other fish, always taking the largest one as the base scale. Finally, once the packages are generated, they must be copied to the configured remote storage.

The setback is that the solution chosen to store these files must be public; otherwise, they could not be accessed. Also, every time the assets are moved, the new address must be updated, and the entire application must be recompiled.

4.7.Deployment platform

To deploy our solution, we can choose any available cloud service provider. In this case, we used Oracle Cloud[25] because of their free VM Instances and Buckets. However, thanks to the flexibility of Docker, we can also use Microsoft Azure[26], AWS[27], Google Cloud[28], DigitalOcean, and others. Of course, it all depends on how expensive the resource properties are to maintain. In this case, our solution is relatively light. It does not need many features. Note that a self-hosted solution is a valid option too.

For this solution, a virtual machine is used, which contains a Docker container running in daemon mode with our server and another Docker container with a static site as a WebGL application. In addition, an Oracle Cloud Bucket for the hosting of the assets loaded remotely by the addressable feature (Figure 64) shows the infrastructure diagram.

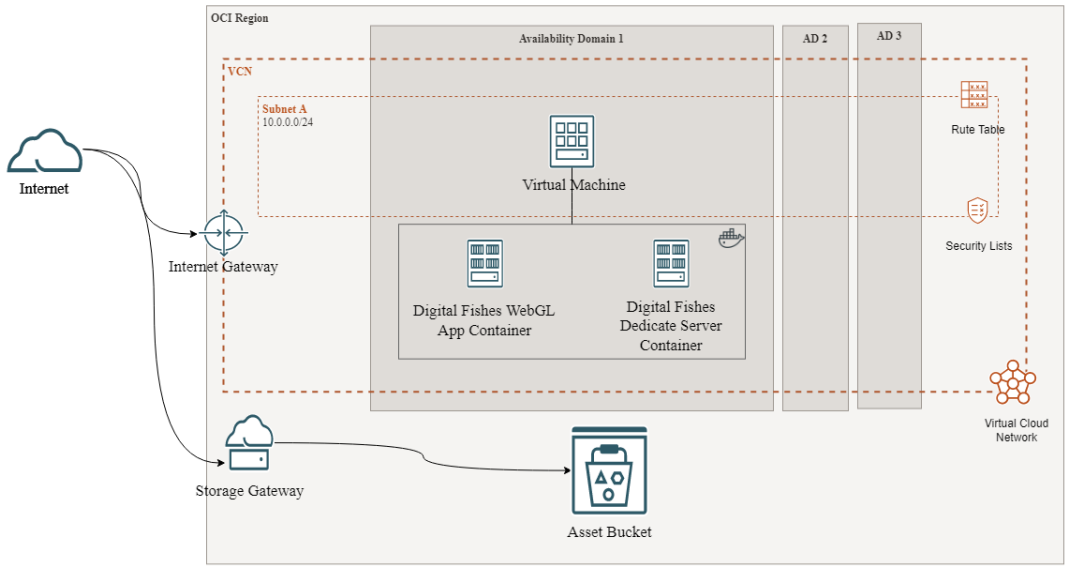

Figure 64. OCI Digital Fishes Infrastructure

[25] https://www.oracle.com/cloud
[26] https://portal.azure.com
[27] https://aws.amazon.com
[28] https://cloud.google.com

4.8. Chapter Summary

As a summary of the development phase, we can summarize the following:

- A simple user interface has been designed that can be used by children, which offers the necessary interactions with the virtual aquarium, such as feeding the fishes and interacting with them by tapping on the screen of the terminals to which the fish direct their attention. Also, a form of physical presentation using a frame to hold three Tablets in the form of a fish pond is proposed.
- The resulting interface also has a details panel in which an enlarged interactive view of the selected fish and its description is shown to give it a more educational feature.
- Models of a selection of fish and algae from Portuguese waters were designed and developed for the aquarium.
- For the aquarium, models of a selection of fish and algae from the Portuguese seas were designed and developed.
- The algorithm Boid was adapted for the development of the behavior of fish in schools. In addition, individual behaviors and interactions between different species as prey/predator were incorporated.
- Optimizations were introduced to improve performance and agility in calculating fish movement using spatial partitioning techniques. This same technique was applied to partition the static physical environment to optimize the calculation of collision detection.
- Shaders were developed to animate the locomotion of fish, the movement of algae, the water's surface, caustics, and others.
- A solution was obtained to remotely configure the virtual aquarium through integration with Unity Gaming Services.
- For the deployment of the solution, images have been created using Docker technologies to facilitate their deployment as containers in any cloud hosting service.

As a result, an Interactive Virtual Aquarium was obtained, which has real-time synchronization between all the terminals for which the client has been released, mainly Web and Android.

5. Evaluation

This chapter describes and exposes the product assessment methods and results of evaluating the Digital Fishes product.

The usability test was conducted using the System Usability Scale (SUS) questionnaire, specifically the European Portuguese Validation of the System Usability Scale [35]. The participants were nine children between 10 and 11 years old, and two of them were female, under the supervision of adults, during the free time activities at "Miúdos e Companhia, Maceira." The survey used the template shown in Appendix A. In addition, the application was presented with the physical prototype made by joining three tablets with which the children could interact.

Question	U -1	U - 2	U - 3	U - 4	U - 5	U - 6	U - 7	U - 8	U - 9
1	1	1	2	3	3	3	1	1	1
2	3	2	5	5	4	4	5	2	4
3	1	4	1	1	2	1	1	1	1
4	5	3	5	5	1	1	5	4	3
5	1	2	2	1	2	2	1	2	1
6	3	3	5	5	5	5	5	5	3
7	1	4	1	2	2	1	1	2	3
8	5	2	5	5	5	5	5	2	1
9	1	1	1	1	2	2	3	3	3
10	1	1	5	5	5	5	5	5	5
Odd	20	13	18	17	14	16	18	16	16
Even	12	6	20	20	15	15	20	13	11
SUS Score	80	47.5	95	92.5	72.5	77.5	95	72.5	67.5
							Avg SUS Score		77.8

Table 3. SUS survey analysis

When analyzing the results of the SUS surveys (Table 3), we can see an average acceptance of 77.8%, which indicates that it is good in terms of SUS usability. In general, the application raised the interest of children and educators, from whom constructive and positive comments were received.

6. Publications

As part of this work, an article entitled "Digital Fishes" has been accepted to the International Conference on Computer Graphics and Interaction (ICGI 2022) as a conference short paper, in which the development process of the Interactive Virtual Aquarium is described, focused mainly on the solution that calculates the behavior of the fish and its optimization.

7. Conclusion

The conclusions derived from the present work are:

- Digital Fishes is an Interactive Virtual Aquarium with a simple user interface that has been designed for all kinds of public. It offers the necessary interactions, such as feeding the fishes and interacting with them by tapping on the screen of the terminals to which the fish direct their attention, in addition to proposing a form of physical presentation using a frame to hold three Tablets in the form of a fish pond.
- The resulting solution interface also has a details panel in which an enlarged interactive view of the selected fish and its description is shown to give it a more educational feature.
- The algorithm Boid was adapted for the development of the behavior of fish in schools. In addition, individual behaviors and interactions between different species as prey/predator were incorporated.
- Optimizations were introduced to improve performance and agility in calculating fish movement using spatial partitioning techniques. This same technique was applied to partition the static physical environment to optimize the calculation of collision detection.
- Shaders were developed to animate the locomotion of fish, the movement of algae, the water's surface, caustics, and others.
- A solution was obtained to remotely configure the virtual aquarium through integration with Unity Gaming Services.
- Supported by an architecture based on multiplayer dedicated server architecture, the obtained solution allows control of the aquarium and makes it possible to have clients released for various platforms.

In summary, an Interactive Virtual Aquarium was obtained named “Digital Fishes,” which has real-time synchronization between all the terminals for which the client has been released, mainly Web and Android. Digital Fishes Interactive Virtual Aquarium currently

recreates a Portuguese marine environment setting, with algae and fish selected from the zone. The fish were created to appear as natural as possible, swimming in groups or alone and reacting to user interactions through contact with the device's screen. The reactions can vary between fleeing or approaching the contact's source and reacting to the user's feeding action. Each species of fish shown has its respective description and a simple way to access it, thus allowing us to learn about the diversity on the Portuguese coasts.

The Digital Fishes Interactive Virtual Aquarium was well received in the tests carried out with a group of children who were very interested in the solution.

We can conclude that with the development of Digital Fishes Interactive Virtual Aquarium, we have managed to obtain an interactive virtual experience of an aquarium which has the potential to become a didactic and decorative tool to encourage schools and teaching centers.

For future work, a few features are recommended:

- It is suggested to fully adopt the DOTS architecture proposed by Unity, which we currently partially do to improve the networking and the performance overall.
- Develop a portal for managing the virtual aquarium, which can directly consume the remote configuration service through a REST API.
- Add more variety of fish species.
- Add different habitats which can be configured.
- Improve the models and quality of the textures and the Art in general.
- The feature of adopting a fish and changing its name can also be added.

Bibliography References

[1] J. R. Li, L. P. Khoo and S. B. Tor, "Desktop virtual reality for maintenance training: an object oriented prototype system (V-REALISM)," *Computers in Industry,* vol. 52, pp. 109-125, 2003.

[2] G. Robertson, M. Czerwinski and M. Van Dantzich, "Immersion in desktop virtual reality," in *Proceedings of the 10th annual ACM symposium on User interface software and technology,* 1997.

[3] H.-C. Lee, E.-S. Kim, N.-K. Joo and G.-T. Hur, "Development of real time virtual aquarium system," *International Journal of Computer Science and Network Security,* vol. 6, p. 58–63, 2006.

[4] T. Lypovyi and J. Montusiewicz, "Simulation of BOID type behaviours in Unity environment," *Journal of Computer Sciences Institute,* vol. 3, p. 23–27, 2017.

[5] "Dream Aquarium," [Online]. Available: https://www.dreamaquarium.com. [Accessed 25 Sept. 2022].

[6] E. M. Studios, "Aqua-tv," 2016. [Online]. Available: https://aqua-tv.co.uk. [Accessed 25 Sept. 2022].

[7] J. Sachs, "Prolific Publishing Inc," [Online]. Available: https://www.prolificpublishinginc.com/store/product.php?productid=64. [Accessed 25 Sept. 2022].

[8] "Softonic," [Online]. Available: https://amazing-3d-aquarium.en.softonic.com. [Accessed 25 Sept. 2022].

[9] Simaquarium, "Simaquarium," [Online]. Available: http://simaquarium.com/en. [Accessed 25 Sept. 2022].

[10] "Baixaki," [Online]. Available: https://www.baixaki.com.br/download/digifish-aqua-real-2-steam.htm. [Accessed 25 Sept. 2022].

[11] “Visit Sealife,” [Online]. Available: https://www.visitsealife.com/melbourne/whats-inside/virtual-aquarium. [Accessed 25 Sept. 2022].

[12] “New England Aquarium,” [Online]. Available: https://www.neaq.org/visit/at-home-events-and-activities. [Accessed 25 Sept. 2022].

[13] “Vertigo Systems,” [Online]. Available: https://www.vertigo-systems.de/en/products/interactive-aquarium/living-aquarium. [Accessed 25 Sept. 2022].

[14] “Future Park Teamlab,” [Online]. Available: https://futurepark.teamlab.art/en/playinstallations/sketch_aquarium_connected_world. [Accessed 25 Sept. 2022].

[15] M. Freitas, L. Pereira, C. Afonso and T. Mouga, *Sabores do mar: algas edíveis do centro de Portugal*, Politécnico de Leiria, 2021.

[16] Gabinete Ministério do Mar, “ÁREAS MARINHAS,” 2018.

[17] C. viva, AS ESPÉCIES MAIS POPULARES DO MAR DE PORTUGAL: NUM RESTAURANTE PERTO DE SI, Viana do Castelo: Ciência viva, 2015.

[18] scrum.org, “Scrum,” *Scrum.org. (Accessed 2022 sept). Retrive from: https://www.scrum.org*, 2022.

[19] Digite, “What Is Agile Methodology? - Overview Of Agile,” *Digite (Accessed 2022, Sept). Retrive from: https://www.digite.com/agile/agile-methodology/#scrum* , 2022.

[20] Wikipedia, “ Unity, Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine). [Accessed 25 Sept. 2022].

[21] K. Group, “Khronos Group,” [Online]. Available: https://www.khronos.org/webgl. [Accessed 25 Sept. 2022].

[22] “Blender,” [Online]. Available: https://www.blender.org/about. [Accessed 25 Sept. 2022].

[23] Docker.org, “Docker overview,” *Docker.org (Consulted 2022 Sept) Reference from: https://docs.docker.com/get-started/overview*, 2022.

[24] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987.

[25] C. Parker, “Boids pseudocode,” *Website: http://www.kfish.org/boids/pseudocode.html,* 2007.

[26] C. Reynolds, “Boids background and update,” *http://www.red3d.com/cwr/boids/,* 2001.

[27] C. R. Drost, “The golden spiral method,” *Stack Overflow.(2017, May 24). Retrieved from: https://stackoverflow.com/questions/9600801/evenly-distributing-n-points-on-a-sphere,* 2017.

[28] A. Beck, Z. Yumak and N. Magnenat-Thalmann, “20 Body Movements Generation for Virtual Characters and Social Robots,” *Social signal processing,* p. 273, 2017.

[29] S. Lefebvre and H. Hoppe, “Perfect spatial hashing,” *ACM Transactions on Graphics (TOG),* vol. 25, p. 579–588, 2006.

[30] M. T. B. H. M. Müller, D. Pomeranets and M. Gross, “Optimized spatial hashing for collision detection of deformable objects,” 2003.

[31] YvesDaoust, “Fast floor/ceiling functions,” *Code Project.(2013, 24 Dec). Retrieved from: https://www.codeproject.com/Tips/700780/Fast-floor-ceiling-functions,* 2013.

[32] D. Rousset, “Understanding Shaders, the secret sauce of 3D engines,” *David Rousset (Consulted 2022 Sept) Reference from: https://www.davrous.com/2020/03/22/understanding-shaders-the-secret-sauce-of-3d-engines,* 2020.

[33] B. Programmer, “How To Animate A Fish Swimming With Shaders,” *Bitshift Programmer.(2018, 12 Jan). Retrieved from: https://www.bitshiftprogrammer.com/2018/01/how-to-animate-fish-swimming-with.html,* 2018.

[34] D. Ilett, “How To Use Every Node in Unity Shader Graph,” *Daniel Ilett.(2021, 20 May). Retrieved from: https://danielilett.com/2021-05-20-every-shader-graph-node/,* 2021.

[35] A. I. Martins, A. F. Rosa, A. Queirós, A. Silva and N. P. Rocha, "European Portuguese Validation of the System Usability Scale (SUS)," *Procedia Computer Science,* vol. 67, pp. 293-300, 2015.

[36] D. Shiffman, "The Nature of Code, Chapter 1: Vectors," *Particle Systems http://natureofcode. com/book,* 2016.

[37] Y. Kryachko, "Using vertex texture displacement for realistic water rendering," *GPU gems,* vol. 2, p. 283–294, 2005.

[38] C. W. Reynolds and others, "Steering behaviors for autonomous characters," in *Game developers conference,* 1999.

[39] L. Munro, "The Role of Color in Product Design: UX of Color Palettes," *Xd.Ideas Adobe (2019, 11, Oct) from: https://xd.adobe.com/ideas/principles/web-design/ux-of-color-palettes,* 2019.

Appendix A: SUS Survey Template

The following template for SUS surveys contains the questions in English and their equivalents in European Portuguese.

	Scale: 1-5	**Escala: 1-5**	
	1 = Agree, 5 = Not Agree	**1 = Concordo, 5 = Nao concordo**	
No.	Original questions	Portuguese translation	Answer
1	I think that I would like to use this system frequently	Acho que gostaria de utilizar este produto com frequência	
2	I found the system unnecessarily complex	Considerei o produto mais complexo do que necessário	
3	I thought the system was easy to use	Achei o produto fácil de utilizar	
4	I think that I would need the support of a technical person to be able to use this system	Acho que necessitaria do ajuda de um técnico para conseguir utilizar este produto	
5	I found the various function in this system were well integrated	Considerei que as várias funcionalidades deste produto estavam bem integradas	
6	I thought there was too much inconsistency in this system	Achei que este produto tenhia muitas inconsistências	
7	I would imagine that most people would learn to use this system very quickly	Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto	
8	I found the system very cumbersome to use	Considerei o produto muito complicado de utilizar	
9	I felt very confident using the system	Senti-me muito confiante a utilizar este produto	
10	I need to learn a lot of things before I could get going with this system	Tive que aprender muito antes de conseguir lidiar com este produto	

ASQ survey:

		Sim/Não
	No geral, estou satisfeito(a) com a facilidade de completar a tarefa neste cenário.	
	No geral, estou satisfeito(a) com o tempo necessário para completar a tarefa neste cenário.	
	No geral, estou satisfeito(a) com a informação de apoio ao completar a tarefa.	

Appendix B: Digital Fishes Interactive Virtual Aquarium screenshots

Figure 65. Digital Fishes Interactive Virtual Aquarium, Waiting Screen

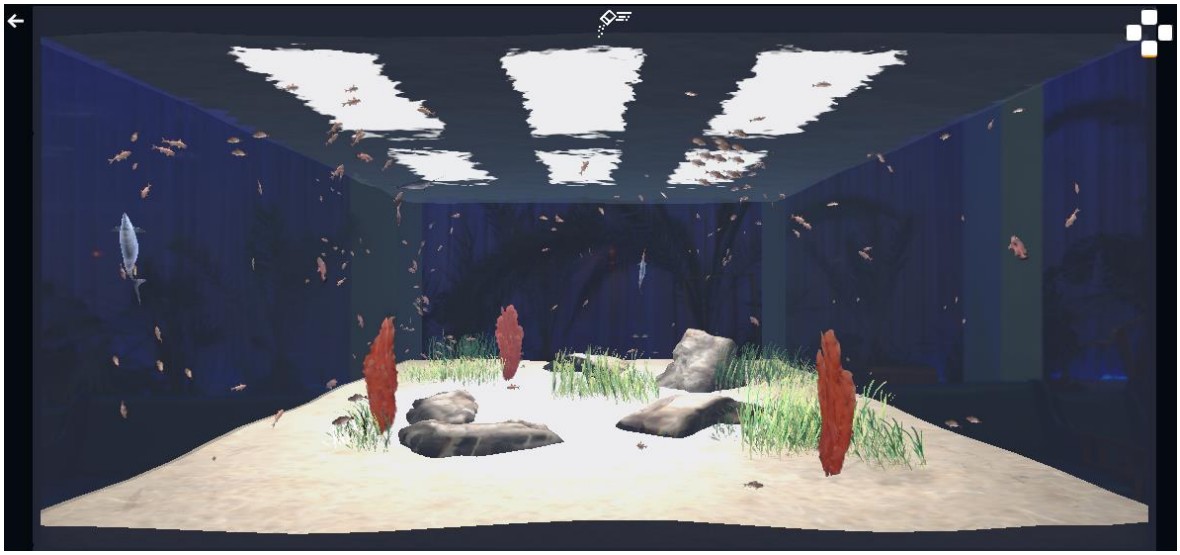

Figure 66. Digital Fishes Interactive Virtual Aquarium, Main Screen

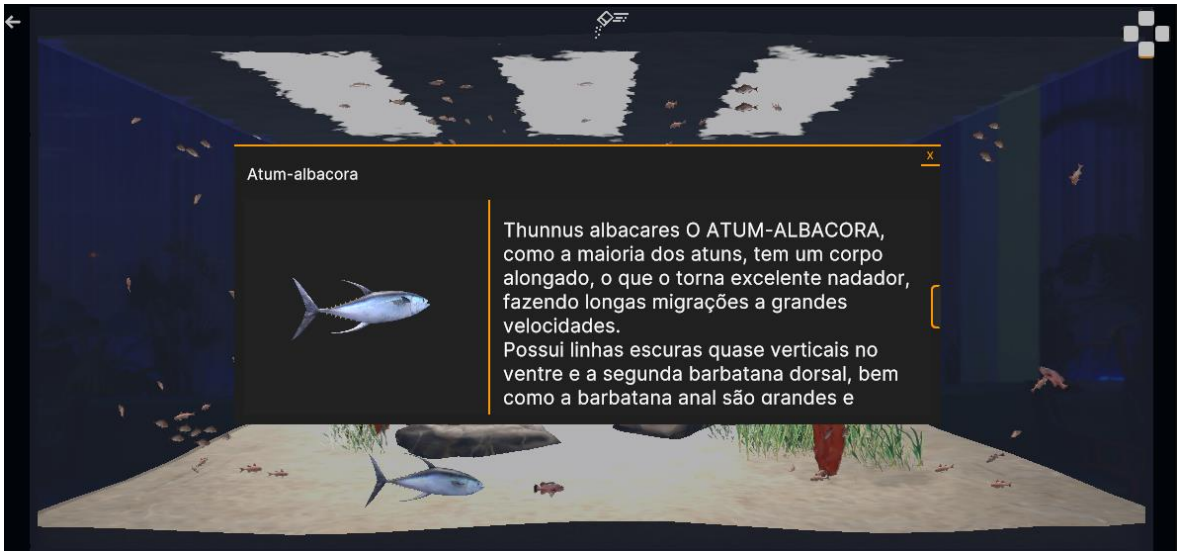

Figure 67. Digital Fishes Interactive Virtual Aquarium, Fish Details