UNIVERSITAT JAUME·I

Grado en Diseño y Desarrollo de Videojuegos

In Search of Adventure
Development and implementation of a naval combat video game

Final Degree Project's Technical Report

César Díaz Delgado

Bachelor's Degree in
Video Game Design and Development
Universitat Jaume I
June 2021

Supervised by : **Carlos Granell Canut**

ACKNOWLEDGMENTS

To begin with, I would like to thank Carlos for his guidance, my parents for their patience, and my sister, my partner and my best friend for having shared their passion for video games with me.

ABSTRACT

In Search of Adventure is a video game of Naval Combat and exploration in Third Person, which could be classified as Third-person shooter game. In it you can immerse yourself in the skin of the helmsman of the great ship of the Straw Hat Band, the Thousand Sunny, which you must pilot through wind, tide and various dangers in order to continue your adventure to the great treasure of the King of the Pirates, the One Piece. This document consists of the memory of the final project of Bachelor's Degree in Video Game Design and Development at the Jaume I University.

CONTENTS

1 INTRODUCTION

1.1 Motivation

As a regular video games player of the One Piece series, no matter what platform, I have always missed giving more importance to a character as indispensable as the ship itself (Thousand Sunny) of the main manga / anime band (the Straw Hat Band), which is more of a companion than an object or vehicle, being relegated in them to a mere transition in cinematics, a battle scene or a base of operations that we can merely explore. And it is because naval combat is so important in piracy, it is unthinkable that this aspect is not dealt with in video games, especially with the great possibilities that a ship like the Thousand Sunny provides, with its great variety of gadgets and mechanisms, capable of destroy a large fleet of ships with a "great blast of compressed air" or even fly through the sky.

That is why when I had to decide how to approach my TFG, based on a complex (and incomplete) modeling of the ship that I made for the subject 3D Models of the third year of college career, a large number of mechanics and attractive ideas began to occur to me that would bring me closer to that desire, since "If something does not exist, create it yourself."

1.2 Objectives

The main objective was simple, taking into account the limitation of time and resources, to develop a game as polished and complete as possible. This project can be divided into the following objectives:

- **Player mechanics:** Since the project was created around him, it had to be as complete as possible, integrating all the mechanics thought with their respective animations, being as attractive and faithful to the original work as possible.

• **Environment:** Undoubtedly, in a naval-themed video game and in a world of pirates, few aspects matter more than the sea, that is why it had to be prioritized, achieving a visually pleasing sea with a certain cartoon aspect. This cannot be empty so it is also important to populate it with islands and enemies representative of the work that limit the player's progress. In this last aspect, the objective is a simple AI that is not a complex challenge for the player.

• **Art**: Both the 2D and 3D art sections have to have a cartoon appearance, visually attractive and with flat colors.

2 PLANNING

In this section, it is explained how time has been invested in carrying out the different tasks that have made up this project (see Figure 2.1). A certain percentage of the total time has been spent writing mandatory documents, including this one. The time prior to which it was devoted to creating and preparing the 3D model of the ship is not taken into account.

Task	Type	Hours
Prepare the 3D model of the ship for the project	Modeling/Art	-
Write GDD	Design/Documentation	5h
Camera 3D	Programing	2h
Pointed system	Programing	10h
Thousand Sunny animations	Design/Art	30h
Thousand Sunny movement	Programing	30h
Shooting mechanics	Programing	20h
Sea Shader creation	Design/Art	10h
Thousand Sunny particle system	Design/Art	30h
Thousand Sunny landing system	Programing	10h
Creating colliders for game elements	Programing	30h
Dibujo de los elementos del HUD	Design/Art	20h
Enemies 3D modeling implementation	Art/Modeling	40h
Enemy IA creation	Programing	25h
Game menu system creation	Programing	10h
Technical proposal	Documentation	15h
Analysis, design and writing of the document	Design/Documentation	40h
Totals hours		+320h

Figure 2.1: Tasks

3 GAME DESIGN DOCUMENT

3.1 Introduction

The main concept of the game is to pilot the famous ship Thousand Sunny across the wide sea, avoiding and facing dangers such as islands and enemy ships, in order to get the great treasure sought by all from the King of Pirates, the One Piece.
The game integrates mechanics of the series adapted to a video game, not seen before since this aspect of the series has never been transferred to a video game. We can use a wide variety of defensive and offensive systems available to the ship, such as blades and large compressed air cannon to make a devastating shot or make an impressive jump to cover great distances quickly.

It is a game to pass the time suitable for all audiences, it is not a game that is very childish or easy, nor is it too mature or complicated. It has a standard theme intended for both Eiichirō Oda (One Piece) manga / anime fans, as well as those who like naval-themed games.

Has been designed mainly for PC given the scope of this platform, since almost every home has one. Also for the ease of demonstration in the defense of the TFG and for its short length.

License:

The video game is based on the world of the famous manga/anime One Piece created by Eiichirō Oda. All image rights belong to him.

3.2 Game Mechanics

The objective in the game is to reach a point located on the map, trying to get the maximum score in the process. This depends on how many objects you can collect, how many enemies you defeat, the amount of life with which you arrive and the time necessary for it. You can choose to make a pacifist race trying to reach the goal without engaging in combat or to reach it trying to defeat all enemies.

Scattered on the map you will find “Cola Barrels” floating in the sea, when you pass near them you will collect them and they will add a small amount of points. This will also be achieved by defeating enemies. On the map you will find islands or portions of the mainland in the form of obstacles. As a trick it will be possible to jump over these using one of the ship's techniques.

The game integrates a 3D camera that follows the player and can be rotated around the player.

A small amount of score and a portion of your "Cola Bar" are added to collect the barrels. This Tail Bar is essential to use the ship's systems (which will reduce a quantity of this with its use):

- The “channel 0” of the “Soldier Dock System” (a set of compartments inside the ship), with which you unfold two side blades, with which you will “dash” forward.
- Two techniques for which it is necessary to have at least half of the maximum Cola, however these require a certain waiting time to load, in which the protagonist may be exposed. These are:

  - **"Gaon Cannon"**: Using the full amount of Tail, the Thousand Sunny fires a powerful blast of air forward that destroys any ship in its area of action. Offensive action.

- **"Coup de Burst"**: Using the full amount of Cola, the Thousand Sunny propels itself through the air a fixed distance forward, destroying those ships that are behind at the time of the jump in an action area. If you cannot land in a safe area surrounded by water, it cannot be used. Evasive action.
If you cannot land in a safe area surrounded by water, it cannot be used. Evasive action.

To destroy enemy ships (action that will add points) you can also make use of lateral cannonball hollows. This action will require a certain loading time once used and its range and trajectory can be modified at a certain angle.

The user makes use of the ↑ ← → ↓ or WADS keys to move the ship around the map, with the mouse you can rotate the camera around the ship and fire the cannonballs with the left button. Holding the right mouse button will use the dash, canceling it once it is released.

Special actions will be activated with the Q and E keys, respectively.

The player does not need to be especially skilled to play, he just has to get used to the controls and the gameplay

The game does not have a save (the game will not last more than 10 min), in each game you start the level from the beginning. This consists of a single level in which you must reach a marked point on the map, skirting portions of land and dodging or eliminating enemy ships. These can ambush you if you enter a closed area such as a gulf.

The score in the game depends on the enemies you defeat and the barrels you collect. Once the goal is reached, your will add that initial score to one calculated with the time spent, the damage of the ship and the number of barrels collected and enemy ships defeated.

3.3 Game States

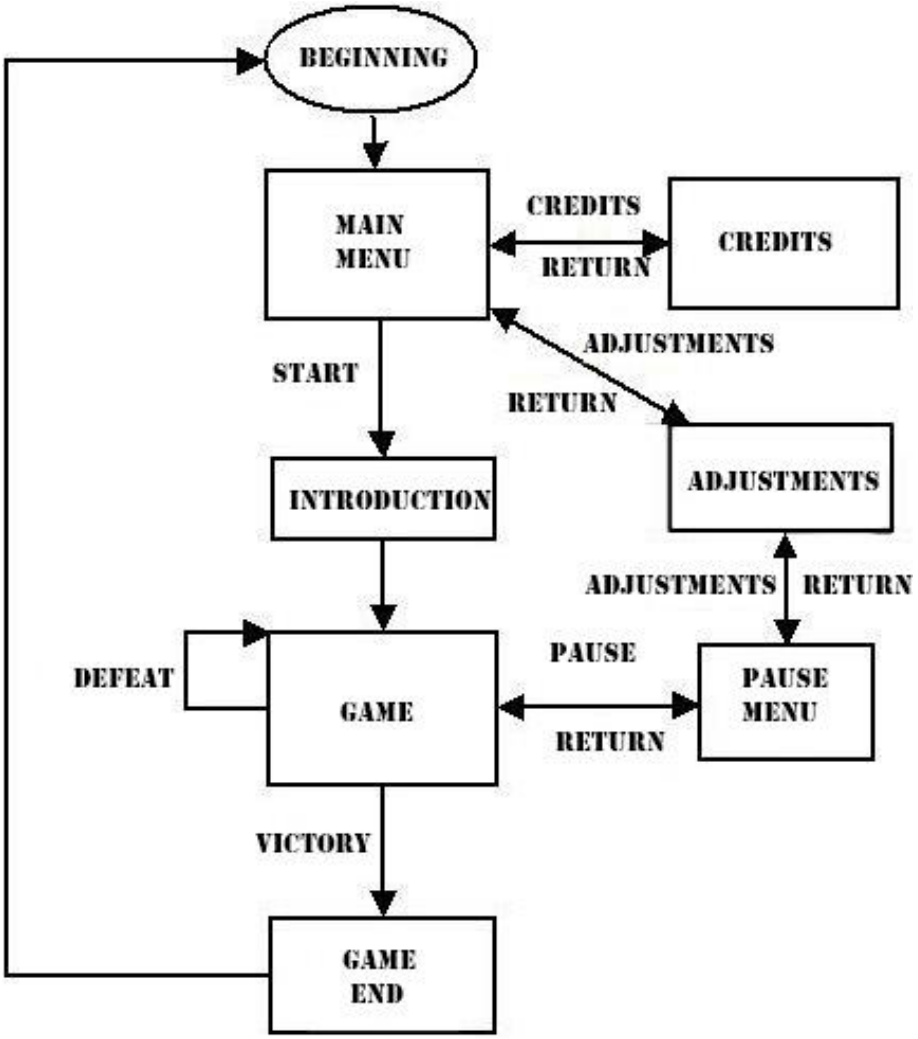


Figure 3.1: Menu flow

3.4 Interface

- **Home Screen**

  This is the first scene we see, in it we see a background image, the name of the game and a text that tells us to press the ENTER key or click on the screen to continue.

Figure 3.2: Home screen picture

- **Main Menu**

  We access this menu from the initial screen, in it we find the same background image and a series of buttons to: Start the game, go to the settings menu, check the game credits and finally close the game.

Figure 3.3: Main menu picture

- **Game HUD**

  In it we can see the life value of the main character, the amount of stored Cola (Cola Bar), a minimap to better locate the elements of the map and some non-interactive panels in which we can check what technique we can spend at that time: The two furthest to the right represent the Gaon Cannon and the Coup de Burst, these light up to indicate that it is possible to use them when the Tail Bar is full; and a third panel that indicates us being illuminated when we can attack with the cannons, once we have attacked, it will lose the brightness and a timer will indicate us when we can use them again.

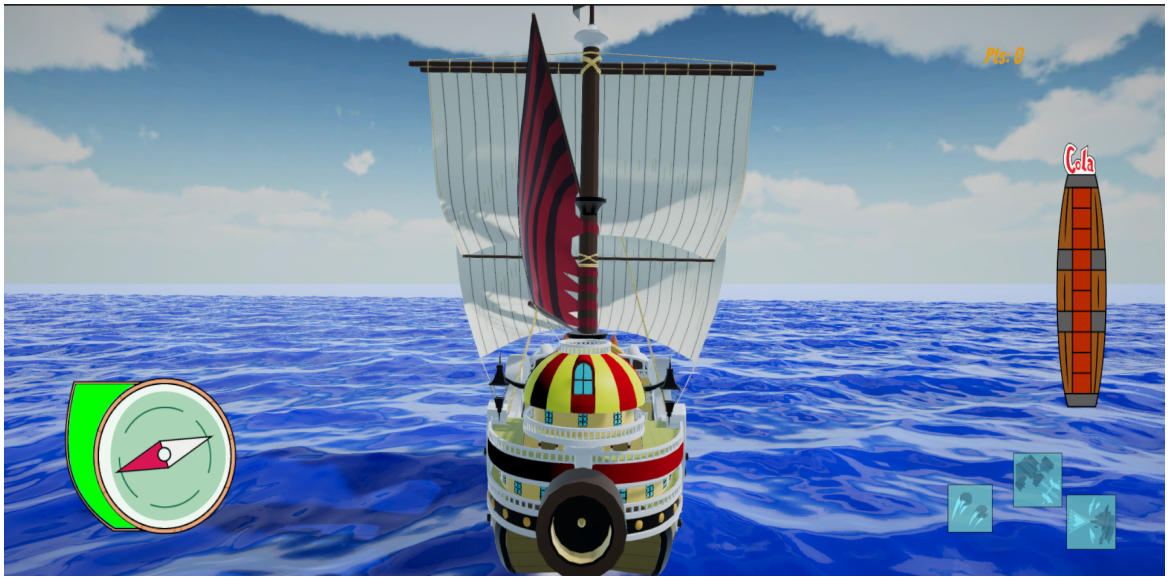


Figure 3.4: HUD picture in gameplay

- **Settings Menu**
  We can access this menu from the initial menu or from the start the pause menu, depending on where we access it, its size or design will vary a little. Here we find a series of buttons to: return to the previous screen and modify the volumes of the game audio or the brightness.

Figure 3.5: Settings menu picture

- **Pause Menu**
  We access from the game with the ESC key, in which we find buttons to: return to the previous screen, go to the settings menu or return to the initial menu.

Figure 3.6: Pause menu picture in gameplay

3.5 Levels

The game consists of a single level, in which we must reach a designated point on the map trying to score the highest possible score. Upon entering, we are presented with the protagonist (Thousand Sunny), a small context and our objective, we are also given the opportunity to teach us a small tutorial to learn the controls of the ship.

On the map we find the glue barrels that we can collect, in addition to the enemy ships that we have to avoid or destroy.

Below we see the score given by each action in the game (see Figure 3.7):

Action	**Score**
Colect Barrel	20
Defeat Enemy	75

Figure 3.7: Game scores

Once the goal is reached, the level ends and the player is shown the score achieved.

The level (located in “Grand Line”) will be limited by two parallel zones where the ship cannot navigate since there is neither current nor wind called “Calm Belt” infested with great beasts called “Sea Kings”, and two zones of very difficult navigation which therefore cannot be accessed either..

3.6 Game Progress

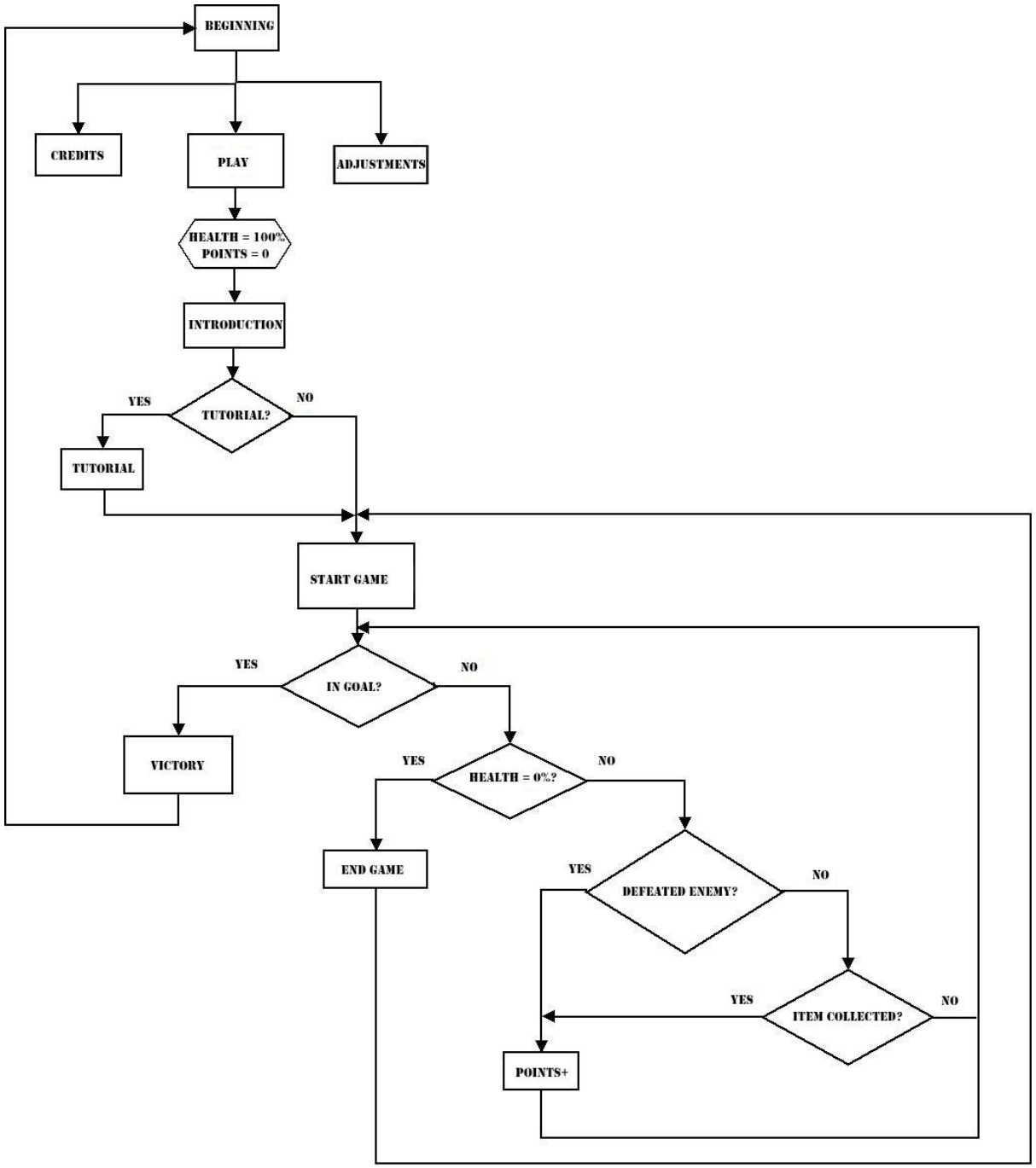


Figure 3.8: Game flow

3.7 Characters

Thousand Sunny : Main character, ship of the Straw Hat pirate gang, whose captain is Monkey D. Luffy. Despite being a lot of wood and metal it has a lot of personality. It contains many mechanisms and compartments that make it an instrument for battle. All of them work based on Cola. It has a lion-shaped mask with two crossbones. It is designed to be the ship of the future King of the pirates and travel the world. We know him as soon as we start the game.

Skills and weapons:

- Two side blades that allow you to increase your speed.
- Seven side cannons with which to launch cannonballs.
- **Coup de Burst**: By means of a propeller in the stern it is able to fly long distances, destroying what is behind it on take off.
- **Gaon Cannon**: Using a cannon in the bow, it is capable of launching a large blast of compressed air forward destroying everything in its path.

Figure 3.9: Picture of Thousand Sunny from One Piece anime

Figure 3.10: Image of Thousand Sunny from the game

ShipCakes: Cake / cupcake-shaped ships, belonging to the fleet of the great pirate Big Mom. They receive their instructions through communication systems called "sea slugs", located in the depths of the sea, so their radius of action is limited to where their signal reaches. We find them scattered across the map.

Weapons:

- Three forward cannons that allow you to attack your enemies.

Figure 3.11: Concept picture of Ship Cake

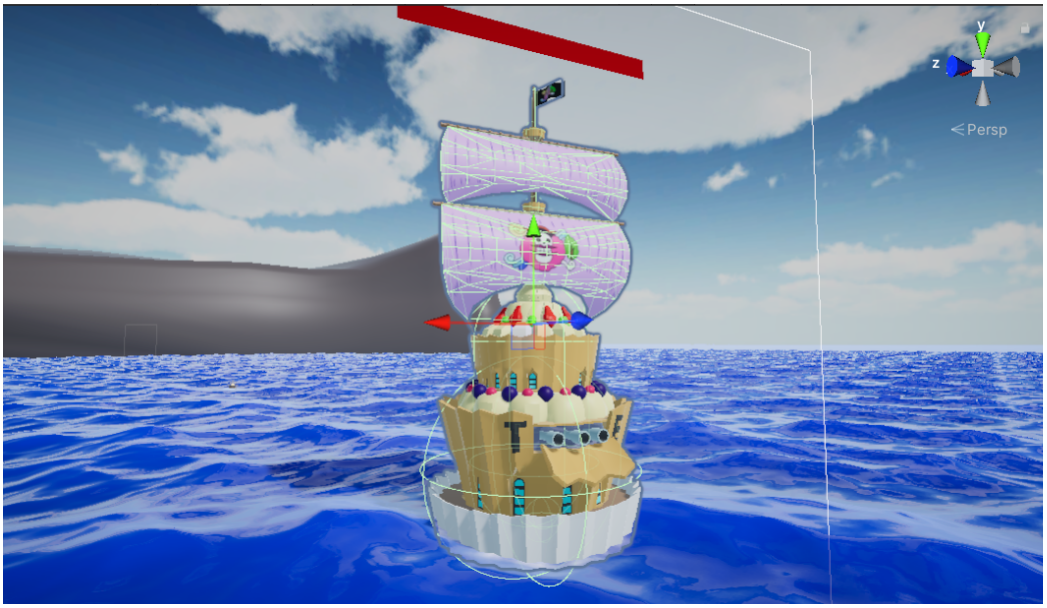


Figure 3.12: Image of Ship Cake from Unity editor

3.8 Items

The only item with which the character can interact are the Cola barrels that we find floating scattered in the sea, these will be automatically collected when passing near them and increase the Cola bar.

3.9 Music & Sounds

Relaxed and catchy music will play at all times, in addition to the sound of the sea.

When the protagonist moves, the sound of the wind will sound, and using the dash you will hear how the blades move the water.

When you perform certain actions like using dash or special techniques, Franky will say a forceful phrase.

In special techniques, the volume of the rest of the sounds will be reduced, it will be heard how the necessary air is charged to use it and finally the explosion.

When shooting (both the protagonist and the enemies) an explosion will be heard first (from the cannons), then the movement of the bullets and finally the impact with the water / target.

3.10 Art & Concepts

All the references of the game have been taken from the anime "One Piece", based on the manga created by Eiichirō Oda.

Renders of the main character:

A soft and cartoon aesthetic has been sought to the elements of the game, with a pleasant appearance for the player.

3D renders of main character:

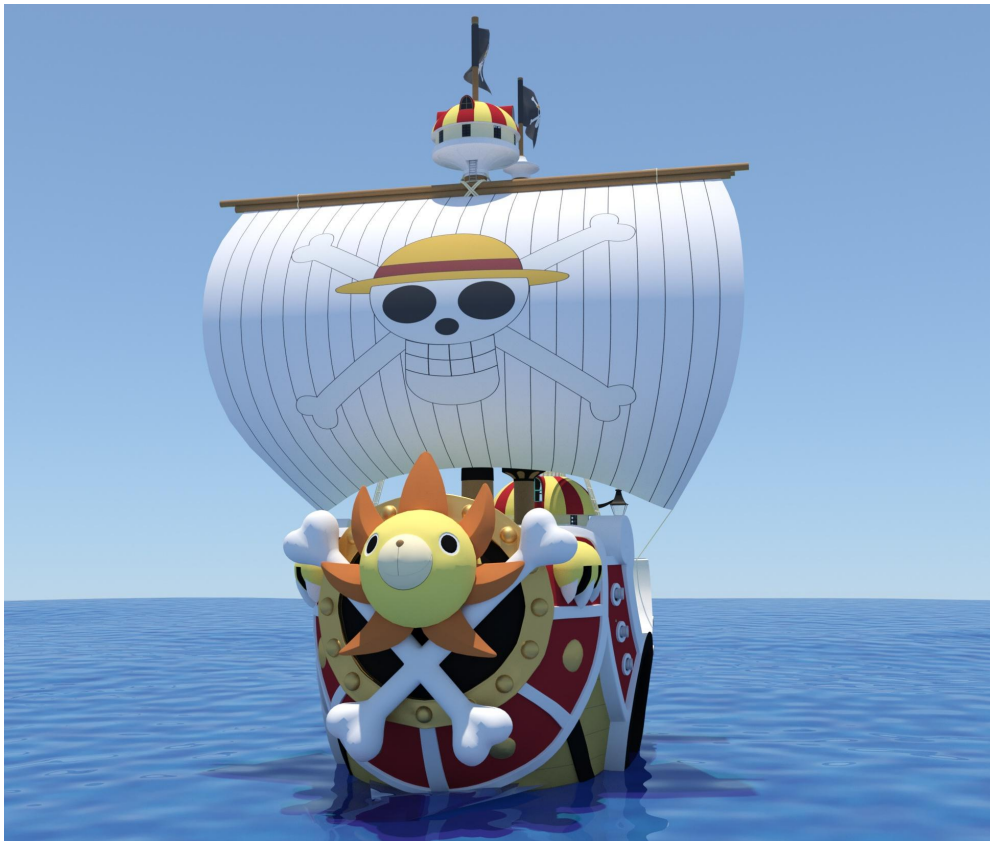


Figure 3.13: Frontal render from Thousand Sunny

Figure 3.14: Side render from Thousand Sunny

3.11 Team Members

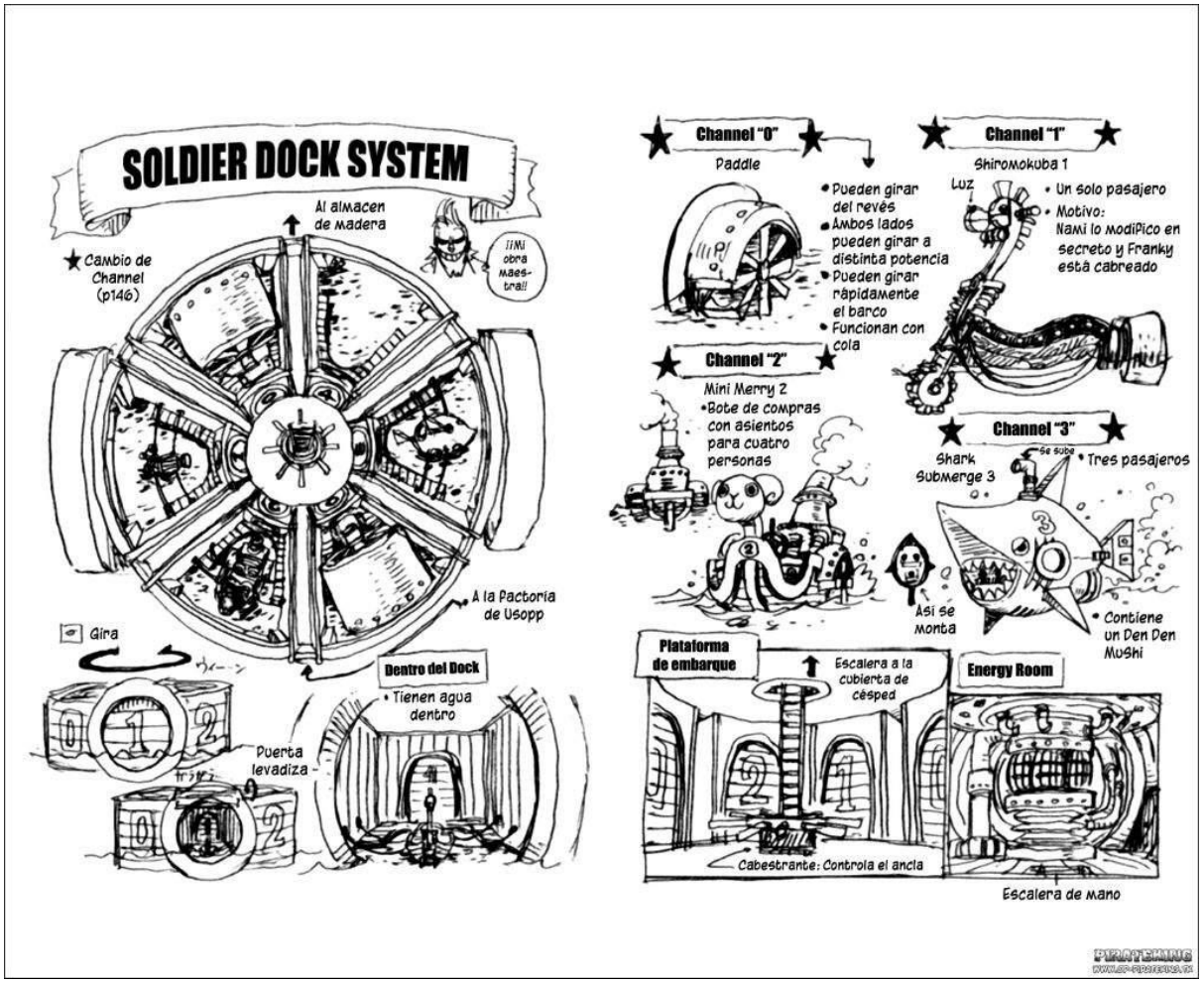
César Díaz Delgado: Programmer and artist of the project

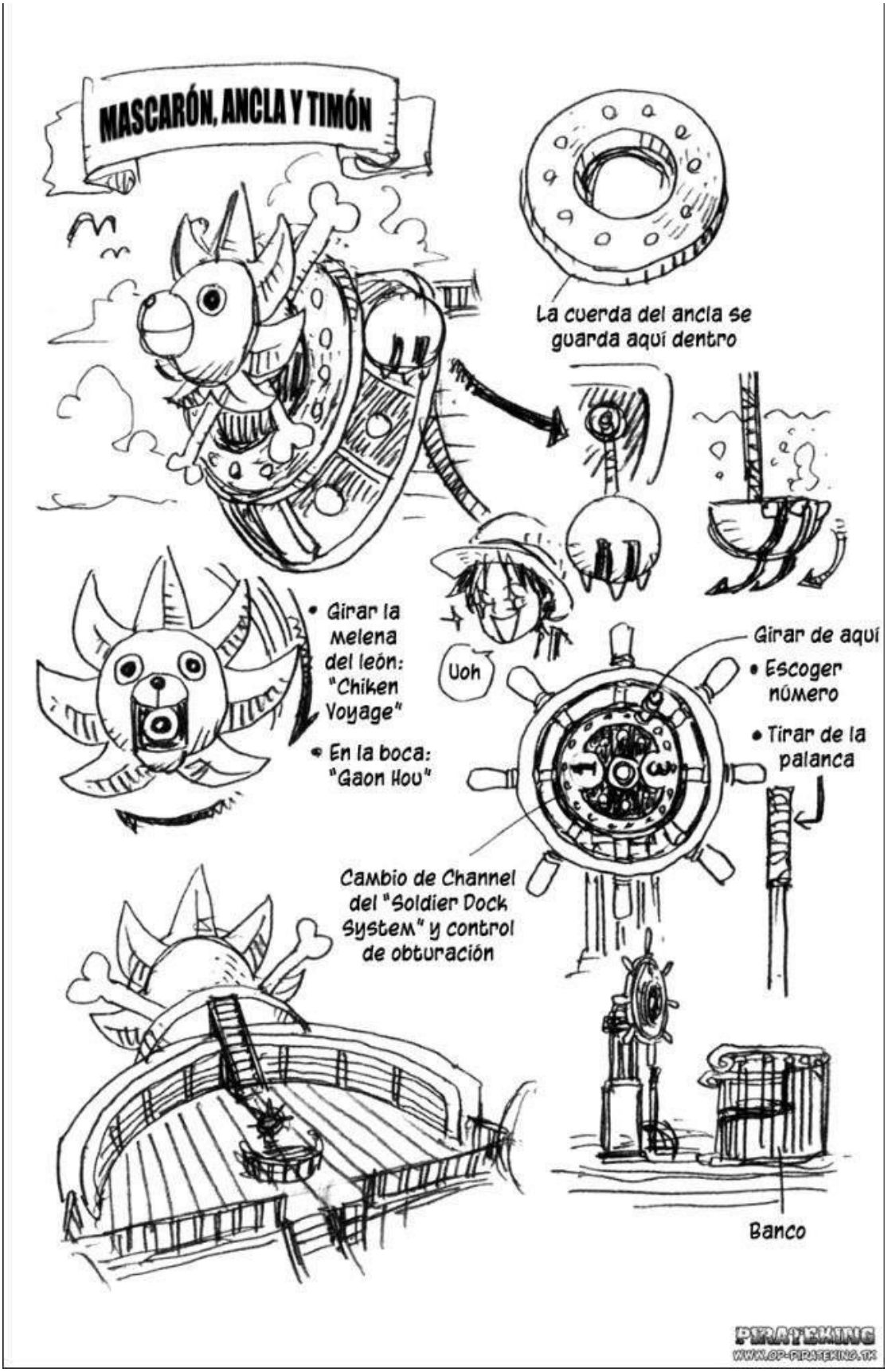
3.12 Work Tools

The project has been created in Unity 3D (version 2020.2.3f1), using as support tools: 3ds Max 2018 for the 3D models, Illustrator and Photoshop for the artistic section and finally Visual Studio 2019 for programming.

3.13 Annexes

Models reference images:




MASCARÓN, ANCLA Y TIMÓN
La cuerda del ancla se guarda aquí dentro
• Girar la melena del león: "Chiken Voyage"
• En la boca: "Gaon Hou"
Uoh
Girar de aquí
• Escoger número
• Tirar de la palanca
Cambio de Channel del "Soldier Dock System" y control de obturación
Banco
PIRATEKING
WWW.OP-PIRATEKING.TK

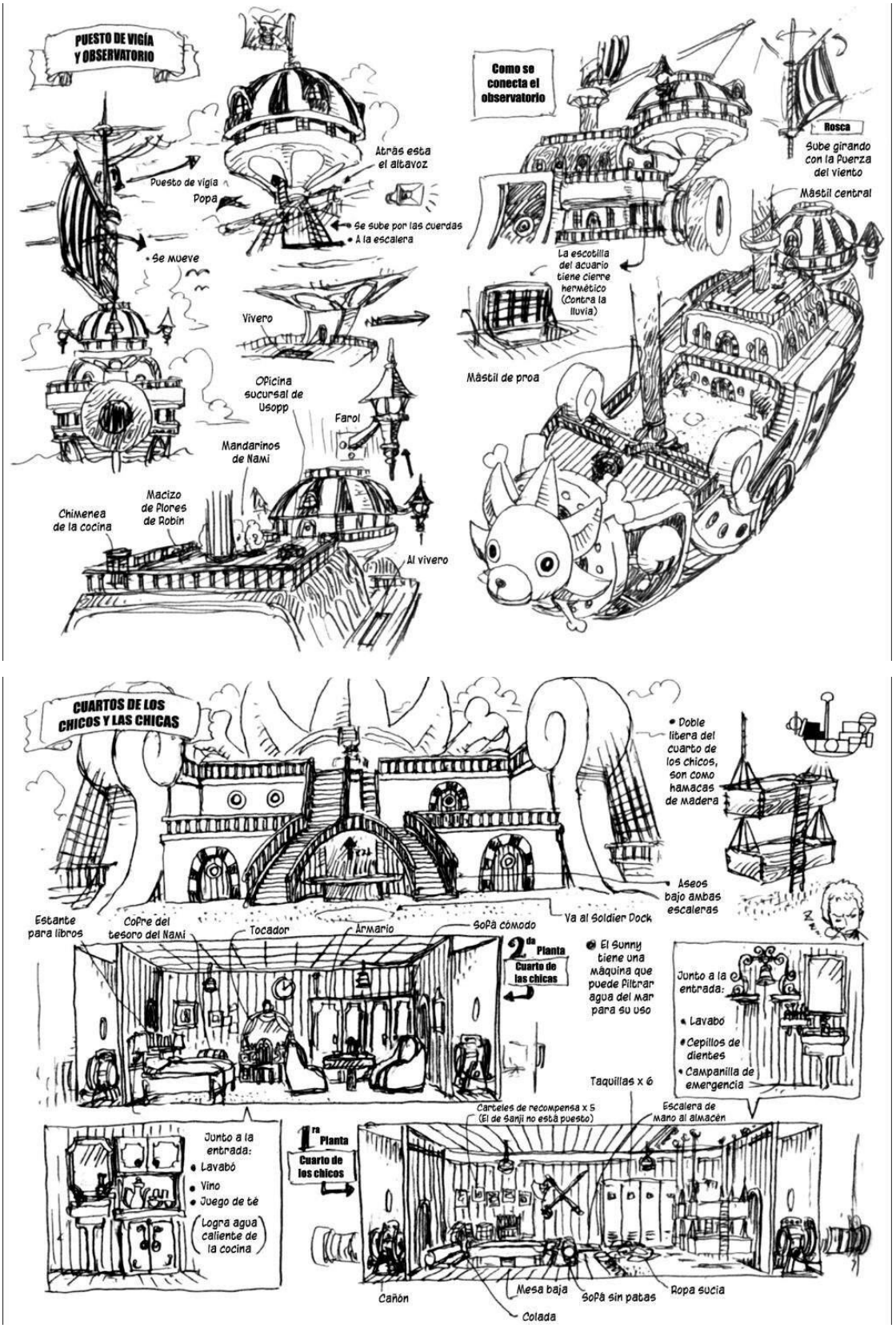


Figure 3.15: Set of explanatory drawings made by the creator

Figure 3.16: Set of images taken form anime One Piece

Figure 3.17: Picture of a Thousand Sunny figure

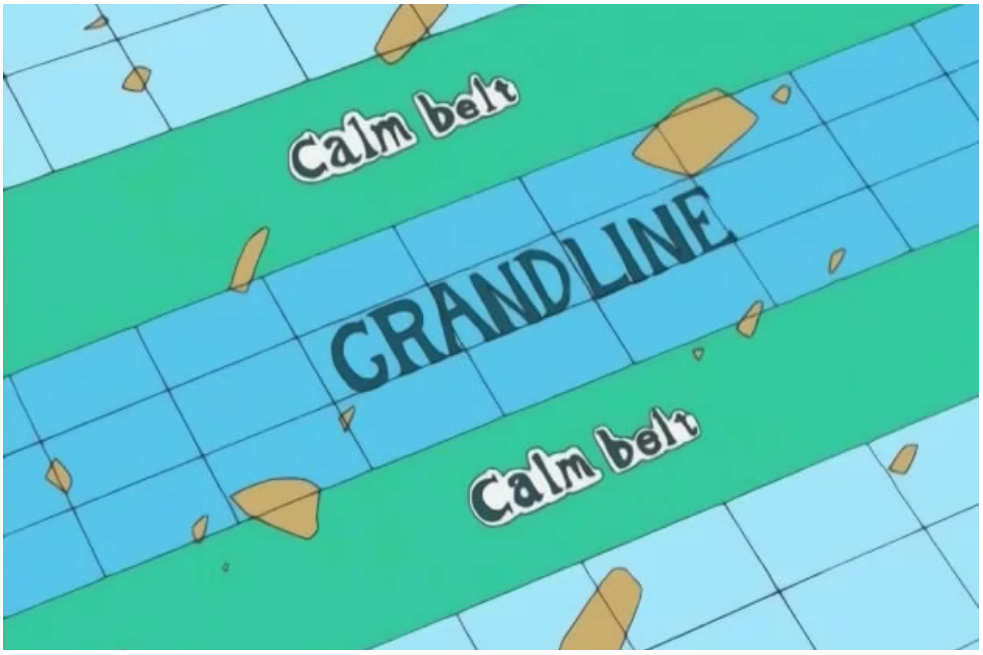


Figure 3.18: Scheme of the sea "Grand Line"

Concepts

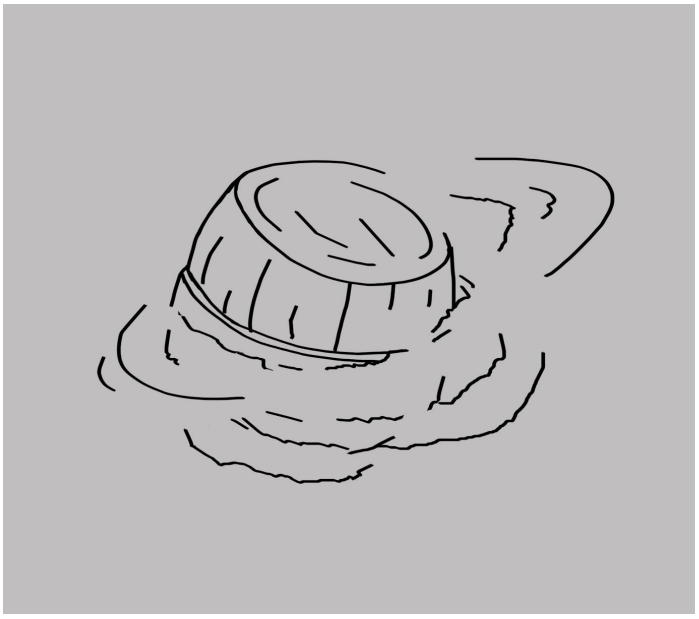


Figure 3.19: Cola Barrel floating sketch

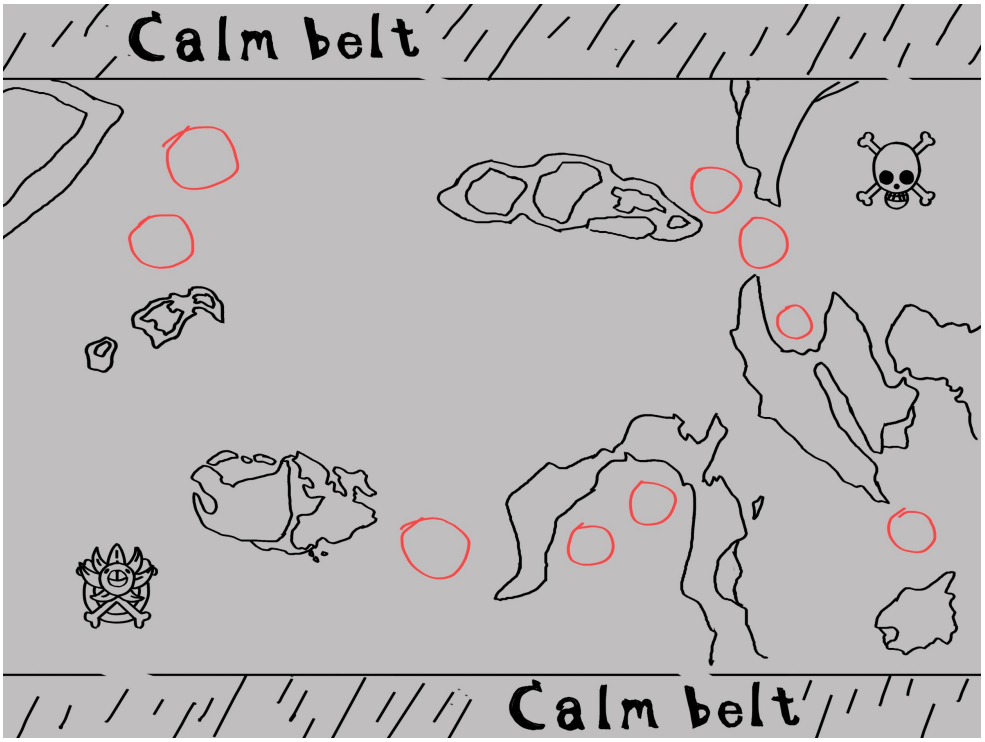
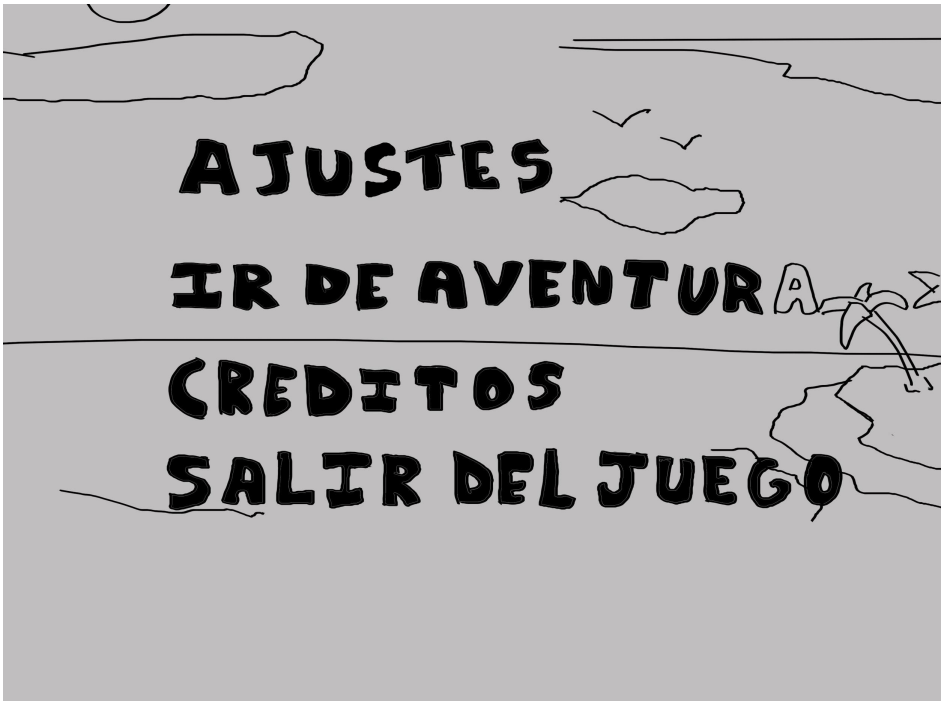
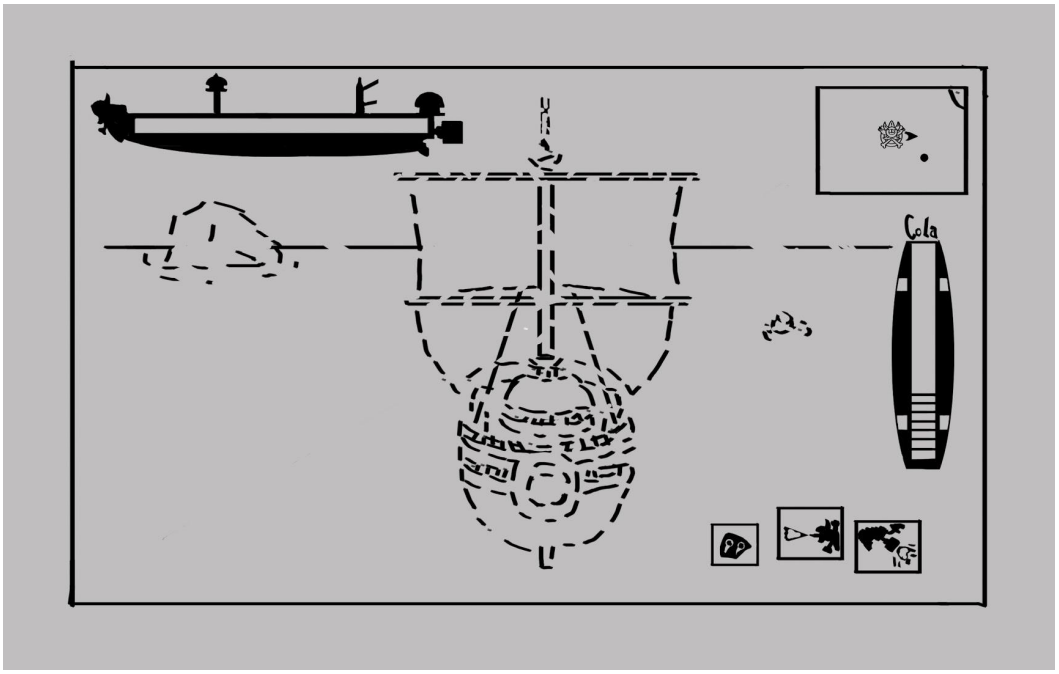


Figure 3.20: Game map sketch




IN
SEARCH
OF
ADVENTURE
PULSA INTRO O CLIKA


Cola

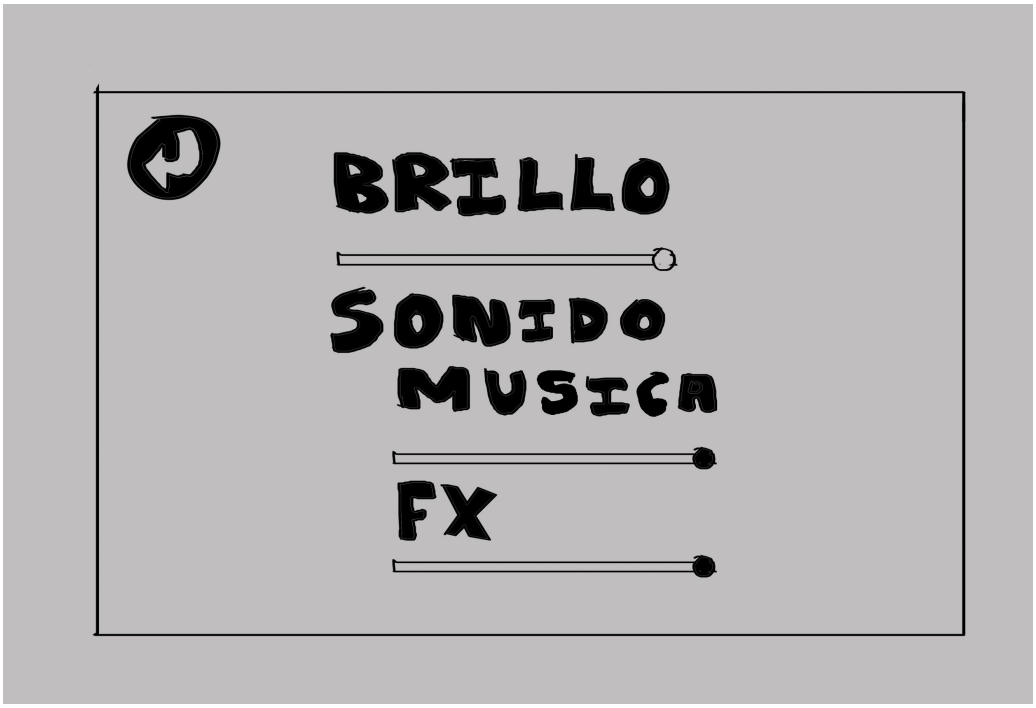


Figure 3.21: Sketches of the various menus

Explanations

Figure 3.22: Crew of the Thousand Sunny (Straw Hat pirate band)

4 DEVELOPMENT & RESULTS

Now it will be explained how the work has been done. To create some of the mechanics and solve problems found in the development process, information has been sought in the manual and official Unity forums, these have been of great importance to solve them.

4.1 Development

4.1.1 Technical Development

The programming language used in the project has been C #, which is the standard in Unity. All the aspects that have been worked on will be explained below. These have been in constant evolution and change, either due to design changes or problems that have arisen in the development process.

1. **Player mechanics**

   Regarding the main character, we find two scripts that handle its main mechanics: Movement and shooting.

   • **Movement:** The first one deals with when to move/rotate the character or if a special action can be performed based on what state he is in: if the character is not in the air (inAir == false) or has an obstacle in front of him, he will be able to move normally (using the movement keys), the movement of the boat is inertial, it does not accelerate to maximum speed or stop instantly, but the speed of movement changes gradually; using the Dash that will increase the speed (holding the shift key, this will be instantaneous) or even using the Burst technique that will make you move a long distance with a jump if you left click with the mouse while pressing the E key (BurstActive == true).

For these two techniques it is key to detect any obstacle in front of the character, for this reason a Trigger (Collider that simply detects the collision, does not prevent bodies from being transferred) has been used in the bow of the ship, preventing it from moving or performing the techniques if it is colliding with another object; An arc is also created that visualizes the trajectory that the ship will follow when making the jump together with another Trigger that is responsible for detecting if there is an obstacle where it is going to land (see Figure 4.1), preventing this technique from being performed and changing both the arc and the "Landing circle" that accompanies the Trigger. It is only possible to land if this Trigger simply detects water.

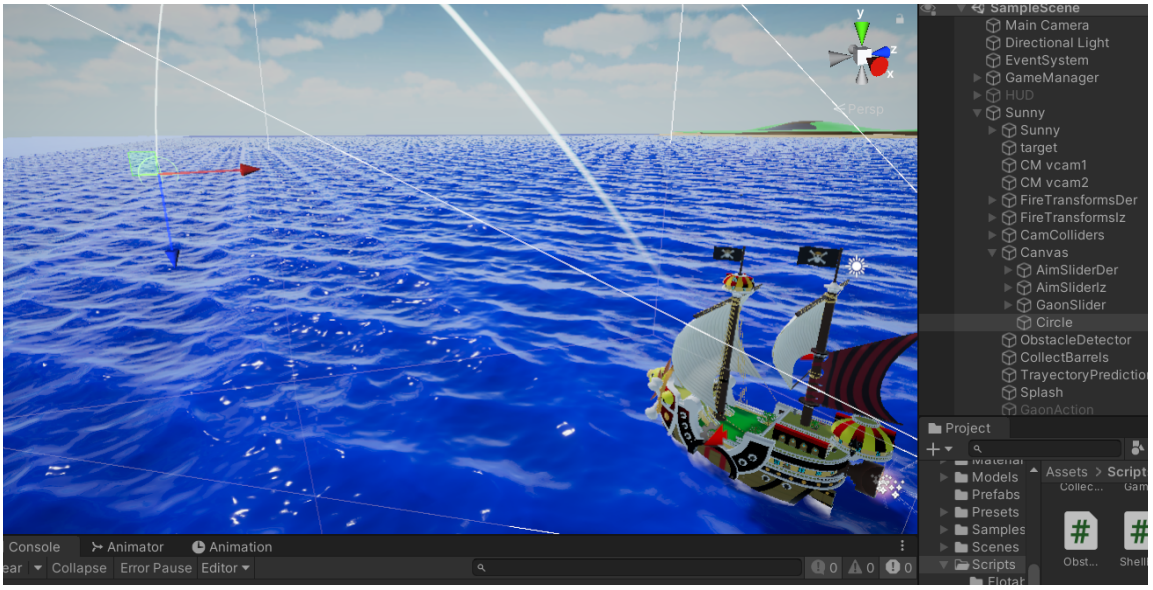


Figure 4.1: Jump path arc and Trigger

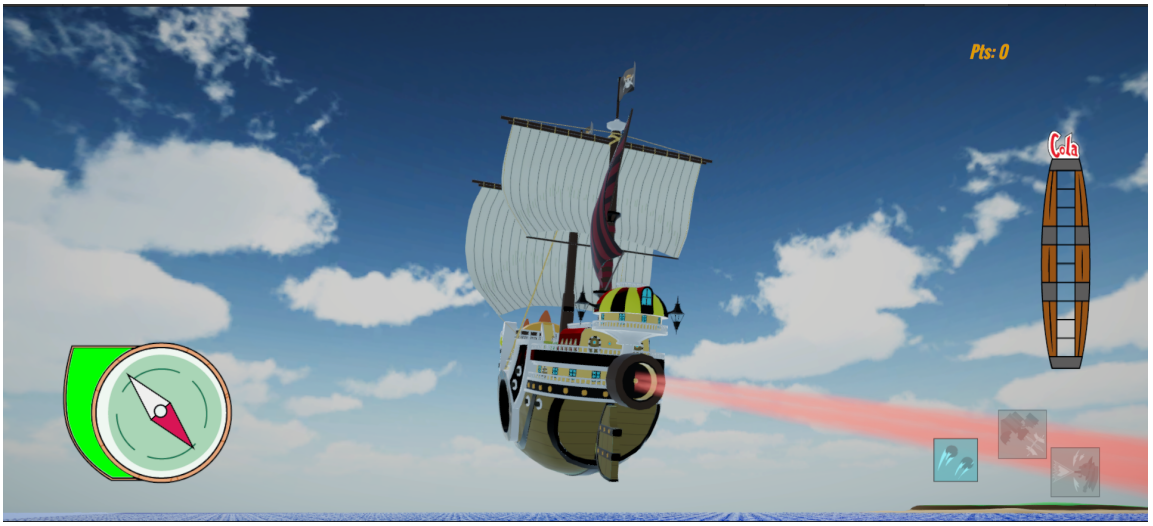


Figure 4.2: Picture of the main character in the air for gameplay

It is worth mentioning that the way to check that it has landed is that it has come into contact with water. For all this, a certain amount of stamina is required, which is achieved by passing near the barrels that are floating on the map (these will also increase the score). It is also responsible for rotating the steering wheel and the boat's own rudder when it turns to achieve more realism.

• **Shooting**: In the second, it is checked when it is possible to point to port or starboard with the right click of the mouse, this depends on what state the character is in and the position of the camera, to later check if it is possible to fire the cannonballs (by a HUD Slider that recharges over time). The second special technique (a wide-range forward attack) that requires pressing the Q key while in the water is also handled. Once a Slider based on the bow of the ship is filled and changes color, this technique can be used by clicking the left mouse button (see Figure 4.3).

Figure 4.3: Gaon Cannon attack sequence

Once it has been verified that it is aiming (it is not possible to do this during the jump), it is possible to modify the trajectory of the cannonballs shot, to later do it by pressing the left click of the mouse. The bullets react with the environment, causing an explosion with smoke if they hit an obstacle or enemy, or splashing if they hit the water.

Figure 4.4: Types of bullet explosions

• **Camera control:** Two types of mouse controllable cameras have been implemented:

- Free camera 3rd person: Camera by default. It allows you to see the space around the ship rotating around it. The camera moves with a slight delay when the boat increases its speed, achieving a certain dynamism to the movement and better appreciating the entire boat.

It has been implemented by the Unity Cinemachine plugin, which defines three circular rails: Top, Middle and Bottom, where the camera will rotate between them (limiting the maximum and minimum vertical rotation) and through them.

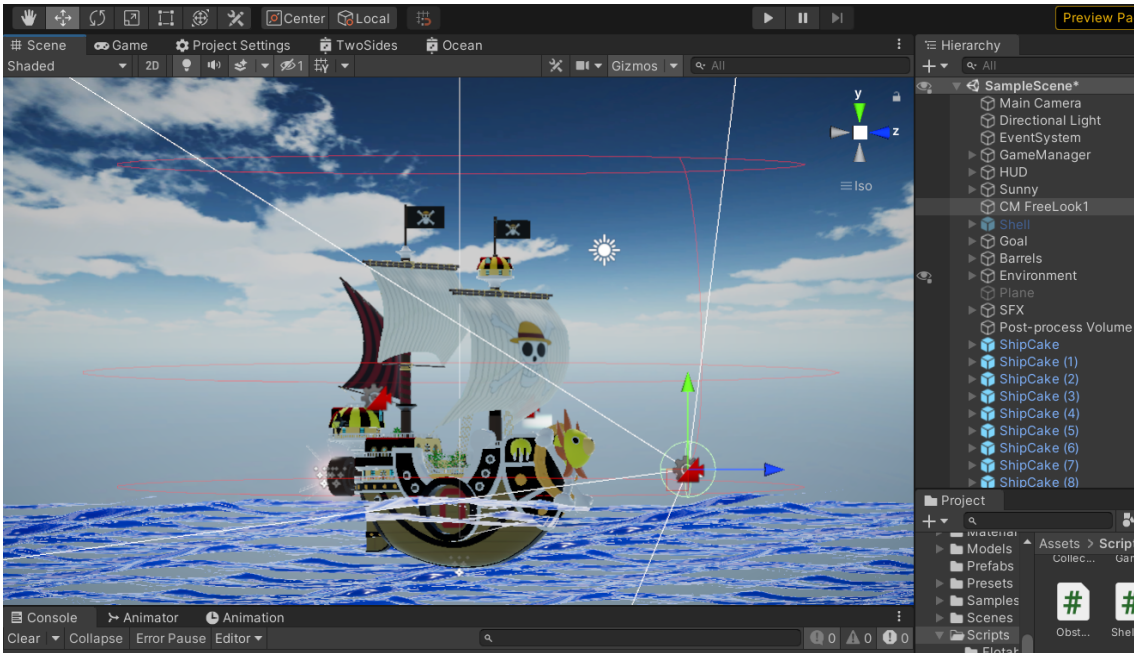


Figure 4.5: 3º camera rails

- Fixed camera 1st person: Used to fire the cannonballs. It allows to aim from the lateral positions of the boat, according to which the 3rd person camera is pointing. This is achieved with the collision of a Trigger that contains the camera and two large Triggers that have the ship assigned to each side, which change the state of a variable (float direc) between 0 and 1. This is characterized by being fixed about meters above the deck and instead of moving it with the mouse, we move a 2D Sprite (see Figure 4.6) that will be rotated using the mouse with the ship as the center of gravity and allows you to appreciate more details of the ship such as the rudder.

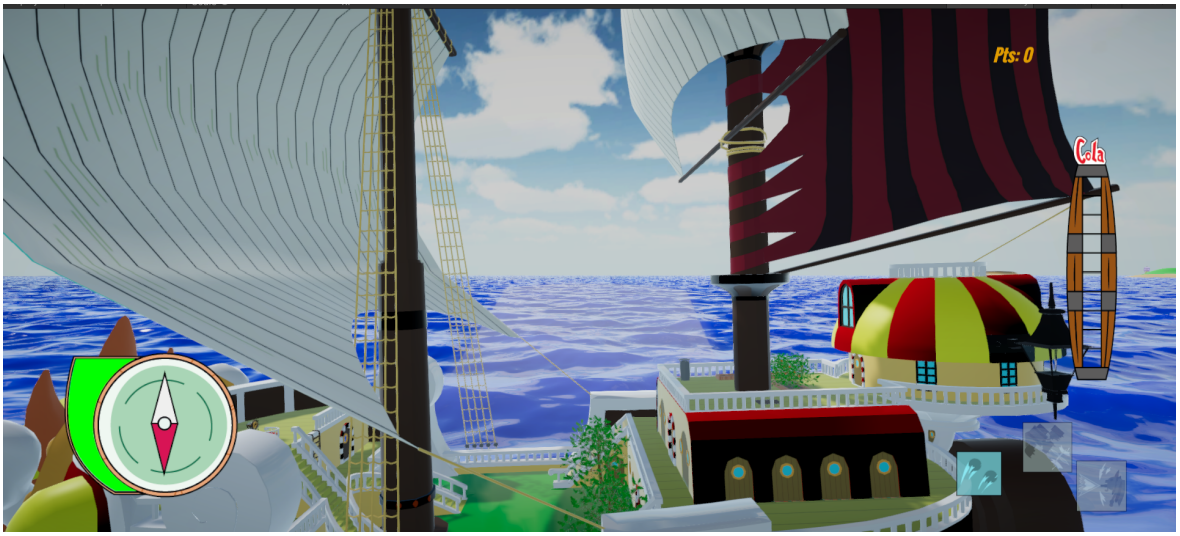


Figure 4.6: Pointed 2D sprite for gameplay

2. **Environment**:

   At first they were going to implement realistic physics regarding the interaction with water, but due to problems with the Unity Shaders and the "unrealistic" shape of the ships in the game, it was discarded for causing strange movements and not being able to implement the foam resulting from that movement due to variables used by the Shader.

   Instead, a Shader has been created using ShaderGraph, which uses the positions and normals of the vertices of the map assigned to it, as well as lighting and stage elements to simulate an ocean with waves, reflections and foam. Two versions of this have been used, the basic one, more abundant and darker, where the player can move, and another lighter, simulating a calm sea that is inaccessible (see Figure 4.7). Afterwards, the movement of all floating objects has been implemented in their animations, achieving a pleasant result.

Figure 4.7: Intersection area of the two types of sea

After this, the ocean was populated with portions of land that would limit the player's passage, assigning them a large number of Colliders that would mold their shape as much as possible, using only the Sfere Colliders, Box Collider Capsule Collider that Unity provides you, limiting a lot its scaling and application. To solve this, a 3D model of a hemisphere was added for use as Mesh Collider, being able to scale optimally and achieve more precise results, however, each collision supposes a greater load for the CPU because this type of Colliders is based on a collision "face to face" of the models, and not calculations of area with center of mass and radius like the rest.

As already mentioned, in this sea we find various barrels floating, which after getting close enough (if the Cola bar is not full), the player will collect and add a small portion of Cola.

• **AI:** In relation to the behavior of the enemies, there are two states: Patrol and pursuit. In the first, each enemy patrols an assigned area, traveling random paths within it (see Figure 4.8). Each enemy is assigned two colliders that serve as vision areas, the first in the form of a long-range cone that detects if the player is in front of them, and the second in the form of a sphere in case the player comes close to the rear. If it detects the player, it will switch to the pursuit state and extract the cannons. If it is some distance from you, it will start to chase you while firing cannonball bursts. In case the player manages to get far enough away from him, the enemy will give up their attack and return to their assigned area to patrol.

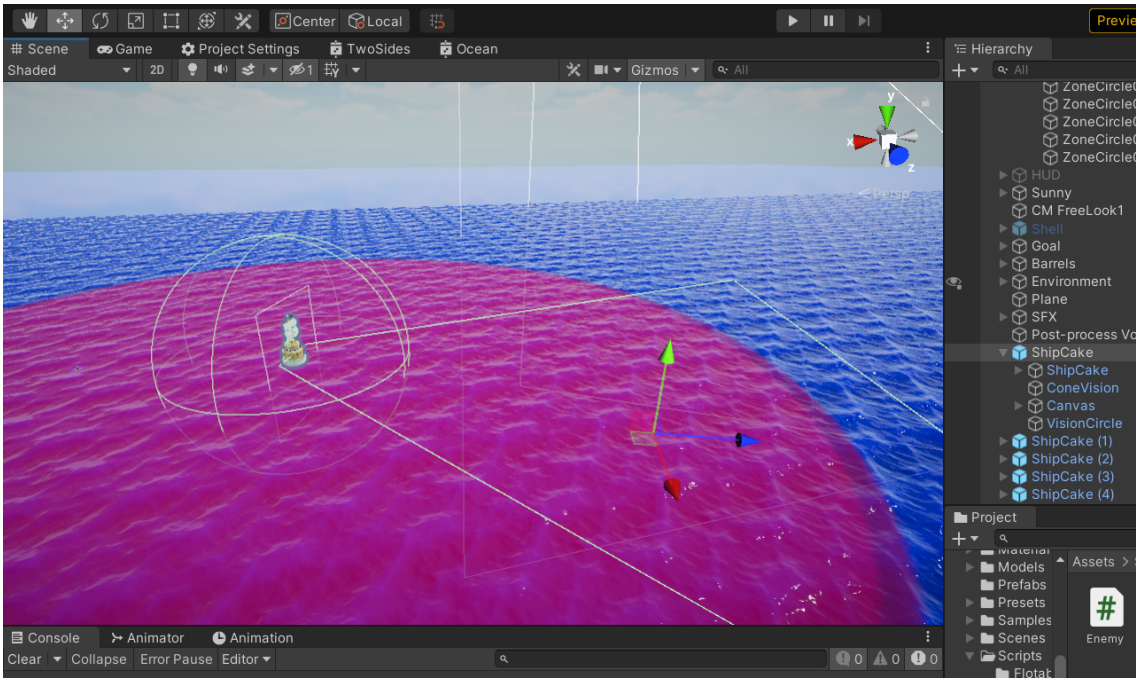


Figure 4.8: Enemies action radius

After all these elements, this is what the game map looks like, with all its interactive elements and its limited areas.

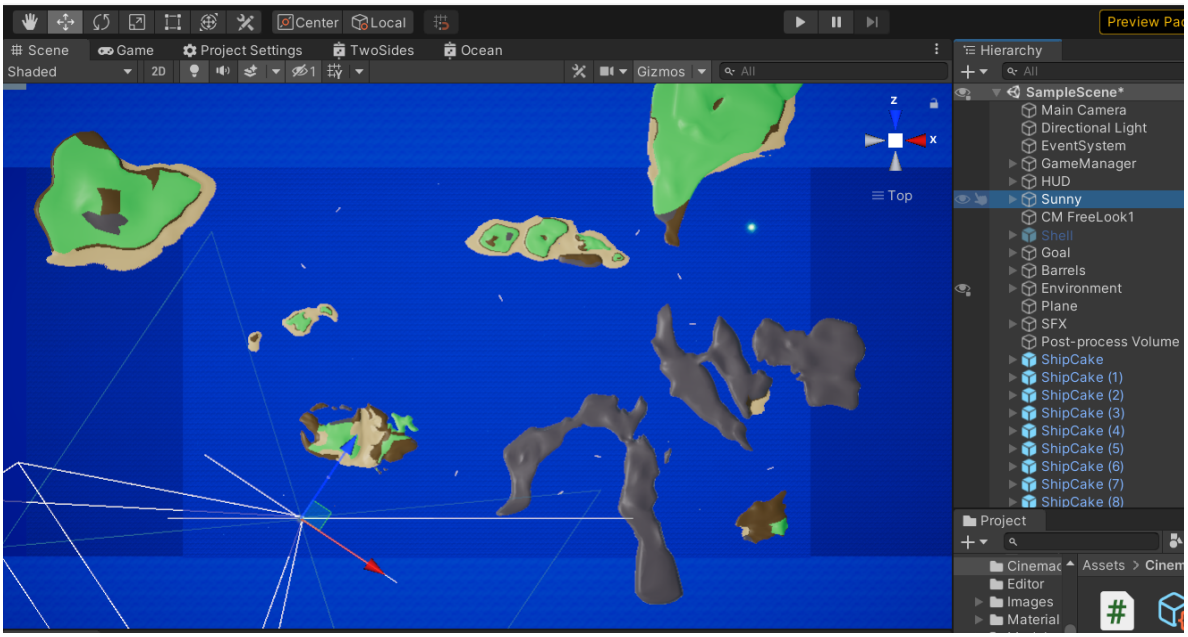


Figure 4.8: Aerial view of the game map

3. **Art**:

Besides from the 3D section already seen, it is worth mentioning the 2D art created for the project:

- **Logo**: An attractive style of bright colors was sought, which would allow the nautical theme of the game to be intuited, that is why a compass (called Log Pose, an important element in the original work) and a steering wheel were introduced.

Figure 4.9: Game logo

- **HUD components:** These sought not to be very intrusive, but attractive according to the main character of the game, we tried to relate the Cola bar with the objects to collect to fill it.

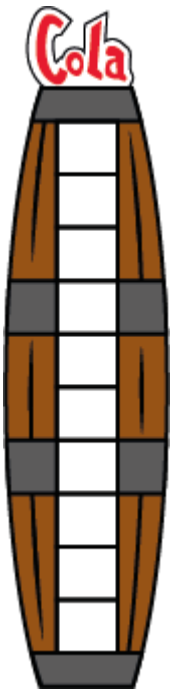


Figure 4.9: Cola bar

At first the health bar was going to be different from the current one, since at first the idea of the compass to mark the way to the goal was not had in mind. After its implementation, the large size of the old bar and the overload of the HUD it was decided to unite both concept.

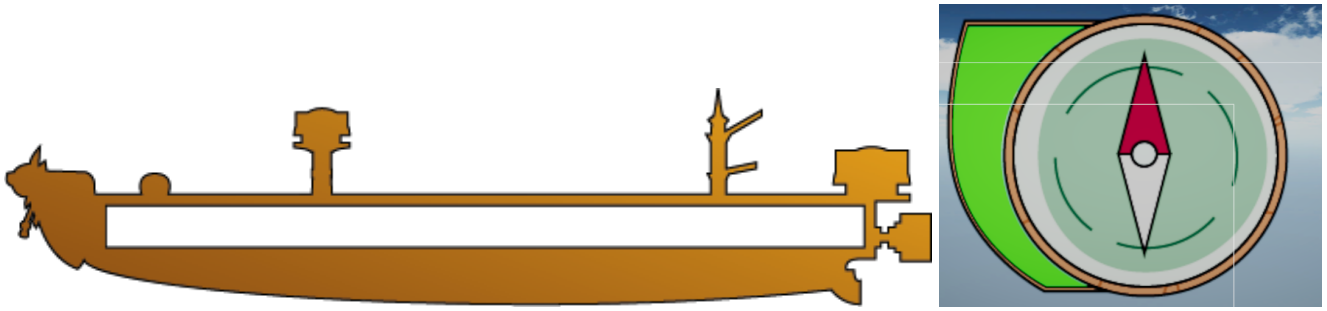


Figure 4.10: Comparative of the two health bars

4.2 Results

In conclusion, I consider that all the proposed objectives have been met, at least on a basic level, however not all the thought and designed aspects of the game have been carried out. Some like AI could have been refined much better, achieving more complex behavior.

It is worth mentioning that as already mentioned, at first the compass mechanics was not conceived, so its design and implementation have been able to subtract a little time from the rest of the tasks, however the final result of the game has improved a lot.

5 CONCLUSIONS & FUTURE WORK

5.1 Conclusions

Being a single project, with the advantages of coordinating each aspect of the game and the clear line of ideas, compared to the disadvantage of having to carry out all the tasks a single person, I think it has been a fairly complete and enjoyable game. However, the limitation of time has been noticed and having to combine this project with other subjects of the degree. I also would have liked to get to implement many more mechanics and aspects to the game.

5.2 Future Work

I would like to continue working on this project, as already mentioned there would be many improvements to add: a more extensive map, areas with change in weather, more types of enemies with different behaviors, interactions with the crew and the islands, etc...

6 PROBLEMS FOUND

Despite being satisfied with the final result, there have been certain situations during the development of the project that have caused me great headaches.

The first of these is the Unity Rigidbody function, which provides physical properties to objects, essential for moving them and creating collisions between them. This has a function to block the movement or rotation of an object in a certain spatial axis, and it is highly recommended to limit, for example, the rotation in the X and Z axes to prevent it from making unwanted movements when forces are used, however I was forced to limit the total rotation (X, Y and Z axes) of the main character of the game since he bypassed these restrictions, and still with that the ship is able to rotate in the Y axis without any explanation.

A "bug" that has also appeared with the rotation of objects has been with the rotation of the rear rudder, this is programmed to turn up to ± 15º on the Y axis depending on where the boat turns, the same code is used to turn the steering wheel, however again without any explanation when the turning of the wheel was implemented after the steering wheel, it stopped working for no apparent reason.

There is also a problem with the rotation of the camera and the aiming, there are times that it rotates abruptly for no reason, making it difficult to use (I think I have fixed it by limiting the maximum and minimum degrees that it is capable of rotating), and the aiming fails in the sense of that many times it does not seem to detect when it changes from one state to another (from right to left), giving rise to an erroneous aim, however once the camera is turned in the third person it seems to pick up the correct state.

Finally, there is a problem related to a function of animations in Unity, called animation events, which allows communicating these with the scripts assigned to the object they modify, however I have found that the events inserted in the animations of the different elements of the game are never performed again for no reason, having to replace this failure with functions from the scripts themselves, executing them after a certain time (the duration of the animation for example) as soon as that animation is executed due to the change of variable that produces the change between the states of the animations.

7 WEBGRAFHY

[1] Adobe Illustrator: https://www.adobe.com/es/products/illustrator.html.

[2] Adobe Photoshop: https://www.adobe.com/es/products/photoshop.html.

[3] Visual studio. https://visualstudio.microsoft.com/es/.

[4] Google documents. https://docs.google.com/.

[5] Enemy behaviour:
http://natureofcode.com/book/chapter-6-autonomous-agents/.

[6] Water explosion: https://youtu.be/KAGx8UMXUwU.

[7] Main character movement: https://youtu.be/Zmh1FFywwx4.

[8] Flag Shader: https://youtu.be/pzo4mitkY8k.

[9] Free camera/Main character movement: https://github.com/Bala-7/Fortnite.git.

[10] Cannonball explosion: https://youtu.be/hCxRPMzhSAo.

[11] Fixed camera: https://youtu.be/1UhT80sbpgU.

[12] Ray "Gaon Cannon": https://youtu.be/TR1xM1HMNQ8.

[13] Ocean Shader: https://youtu.be/kgXeo2SRDd4.
https://www.patreon.com/polytoots/posts.

[14] Trajectory prediction: https://youtu.be/agSRx6Lgnig.

[15] Settings: https://youtu.be/JivuXdrIHK0.
https://youtu.be/9ROolmPSC70.

[16] Unity manual: https://docs.unity3d.com/Manual/index.html.

[17] Toon Shader: https://roystan.net/articles/toon-shader.html.
https://dev.to/jfcarocota/como-crear-un-shader-toon-con-un-modelo-de-luz-custom-y-shader-graph-1c09.