Social game through asymmetric design and interaction with physical objects

Javier Lebrija Morillas

Advisor: Dr. Diego José Díaz García

Universitat Jaume I

This dissertation is submitted for the bachelor's degree of Video Game Design and Development.

Castellón de la plana June 2018

ABSTRACT

Social effects of video games are frequently examined from different angles. The objective of this document is to create a social gaming experience keeping in mind the major traits of videogames for improving social relations. For this purpose, a local-multiplayer game named Pocket Crew has been developed for Android using unity engine. Several studies of different disciplines have been taken as reference when designing the concept of this game.
Players must collaborate to take a pirate ship to an island and find a treasure. What stand out in this idea is the use of Asymmetric Design, so that each one will perform independent actions in their device, that combined with those of the rest will make the ship to reach the isle. Also, the use of physical objects, like the isle map and the nautical. They will have to work as a team if they want to be successful in their pirate adventure.
The experience is defined by three elements: The printed map of the isle to which players will travel, the printed nautical chart from which obtain the selected isle coordinates and the android app to play the game.

Keywords: Collaborative, Social Game, Interdependence, Asymmetric Design, physical objects, Social experience

INDEX

LIST OF FIGURES

LIST OF TABLES

1. TECHNICAL PROPOSAL

This section of the document presents a general overview and a very early estimation in time about the workflow that project will require. Also the related literature is reviewed at the last part.

1.1. INTRODUCTION AND MOTIVATION

Introduced by Nintendo in 2012 with the Wii U, asymmetric design has become an increasingly used resource on the industry. A close definition would be as Raphael Guilleminot said: “the gist of it is providing players different control schemes and specialized objectives in the same game scenario” [1]. It is important to not confuse with asymmetric balance in games. Balance is about how players starts the game, chess has symmetric balance because the players starts with the same pieces, only with different colors. StarCraft for example, has asymmetric balance because each team have different units with their own fortress and weakness, but all can defeat each other’s [2]. Asymmetric design is based on hardware, creating new mechanics through it. There are other fields of use, like remote controlled cameras, space launch control rooms or remote surgery installations [1].

The motivation for this project is to show the social potential of video games, making it clear that them are not so far from board games. Board games are the beginning of video games, from chess to Role Playing Games (RPG) but it seems to be away from each other nowadays. People tends to play isolated, even in multiplayer games, players are in their own houses, separated from the rest. It is good for meet people, but we must not forget to interact with others. I think video games have an inherent potential to be social, because they are originated from board games, which are social by definition.

1.2. RELATED SUBJECTS

- Programming II (VJ1208): This subject not only pretend to teach C# language used in Visual Studio, also focus on the importance of creating software free of errors and well-structured in order to be easy for debugging and maintaining.

- Conceptual Design of Video games (VJ1222): This subject set the bases for game creation, separating each part of the video game and teaching different ways of joining them to create solutions adapted to our needs.

- Art of the Videogame (VJ1223): The objective of this subject is the learning of the creative process of a game in a parallel way to the development, as well as the bases for the correct representation of the ideas suggested in the game script.

- Game Engines (VJ1227): This subject introduces the basic architecture of a game engine with special emphasis on the graphic engine and gameplay. In relation to the creation of video games, the course focus on managing Unity Engine scripting system.

1.3. OBJECTIVES

Main Objectives:

- To create a functional game with asymmetric design
- To generate a fun game dynamic which encourage collaboration
- To develop a game which research the social potential of video games
- To develop a game which will be the most closer possible to a board game

Secondary objectives:

- To implement the connection between the devices in a simple way for the players
- To implement the shipment of random ship parts correctly to each device

1.4. SCHEDULE

The work methodology will be SCRUM with 4 sprints.
The schedule is planned to work 20 hours per week, making 300 hours in total.

Table 1: Sprint 1 schedule

Sprint 1		
Task	*Estimated Time (in hours)*	*Estimated Delivery*
Technical proposal	6	04/02/2018
Analogical prototype	10	17/02/2018

Table 2: Sprint 2 schedule

Sprint 2		
Task	*Estimated Time (in hours)*	*Estimated Delivery*
Game Design Document	10	25/02/2018
First functional demo (singleplayer)	40	11/03/2018
Implementing connection among several devices	60	01/04/2018

Table 2: Sprint 2 schedule

Sprint 3		
Task	*Estimated Time (in hours)*	*Estimated Delivery*
First multiplayer functional demo	40	15/04/2018
Art, riddles and Interface	50	29/04/2018

Table 4: Sprint 4 schedule

Sprint 4		
Task	*Estimated Time (in hours)*	*Estimated Delivery*
Sound searching and implementation	20	06/05/2018
Testing and Debugging	30	13/05/2018
Report	20	20/05/2018

Table 5: Final revision schedule

Final revision		
Task	*Estimated Time (in hours)*	*Estimated Delivery*
Final Report	4	03/06/2018
Final presentation	10	12/06/2018

1.5. EXPECTED RESULTS

The following items are expected to be achieved with the resulting project:

- To get a videogame with a board game format, in which players must be physically gathered and share objects directly with others.
- To implement the connection to each one's devices as in a very easy way and technical knowledge doesn't be a necessity.
- To create a gameplay easily extendible with new maps in a future.
- To create an enjoyable experience.

1.6. SOFTWARE RESOURCES

1.6.1. PROGRAMMING

- **Unity 2017.3:** Game engine for make 2D and 3D games.
- **Unity Multiplayer Services:** a multi user server service to allow your game to communicate across the internet. It provides the ability for users to create matches and advertise matches, list available matches and join matches.
- **Visual Studio community 2017:** IDE liked with unity.

1.6.2. UI & 2D ART DESIGN

- **Photoshop CC:** Graphic editor.

1.6.3. DOCUMENTS

- **Google Docs:** Word processor.
- **Google Slides:** Software for editing slide show presentations.
- **Google Sheets:** Software for editing spreadsheets.
- **Microsoft word:** Word processor.

1.6.4. RESEARCH

- **Mendeley:** Program for managing and sharing research papers.
- **ResearchGate:** Site for scientists and researchers to share papers.

1.7. RELATED LITERATURE

In this section the different traits that make up this game will be reviewed one by one: Cooperative, Local-Multiplayer, Physical interaction, Social game and Asymmetric design. Commenting on the different studies in each field that demonstrate the benefits of these in social relationships.
At point 1.1 of this document was commented that people tend to play in isolation, this being the idea that most people have in mind about "gamers", and the social potential of video games was also mentioned. This has been subject of study and there are many articles [3] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] that investigate the effects of different types of games in people and their lives.

According to the results of one of these studies [3], it has been concluded that the majority of young people do not believe that the video game conditions them to isolate or not themselves. Being those who consider games as educational media those who see in them a possibility of social relationship, compared to those who do not play anything, who tend to think that video games isolate. It also corroborates another study [14] in which it is observed that in the great majority the players' social relations were not affected for better or worse, while 10'6% saw an improvement in them.

Other article [5] focus on improvements in social relationships that involve the different types of games, such as social games [4], referring to the game that opposes to be played in solitude. In this specific field it is affirmed that video games are the ideal environment to improve interpersonal trust, showing that they were better than other social activities called "icebreakers" (exchange of personal information, for example) when creating links. They explain this by stating that conversations were subject to the affinity of the subject, with factors such as age, sex, personality or experience. While the game obtained positive results independently of these factors.

If we move to another aspect such as cooperation in games, there are also many studies [6] [7] [8]. "Cooperation is a common mechanism present in real world systems at different scales and in different environments, from the biological organization of organisms to human society" [6]. Specifically, it has been shown that playing a cooperative game with a group of strangers before carrying out a group task improves the results of the task concretely in terms of communication [7]. Another study has concluded that cooperation and interdependence are factors that can be used in combination or separately to improve social relations [8]. "... Social interdependence exists when individual goal attainment is affected by others' actions (Johnson & Johnson, 1989). There are two types of interdependence: Positive interdependence refers to a situation in which there is a positive relation between goal attainments of individuals, whereas negative interdependence exists when individuals perceive that they can obtain their goals (only) if the other individuals with whom they are competitively linked fail to reach their goals (Deutsch, 1949Deutsch, 1962Johnson & Johnson, 1989). "[6]. Positive interdependence fosters greater acceptance and positive relationships, as well as an increase in motivation [6].

The use of asymmetric design is also studied as a tool for players of very different characteristics (age, style of play, personality, etc.) to have fun playing the same game, since they can choose the mode that best suits them. Facilitating the collaboration in the game [9] [10].

This literary context is the reason why my game has these characteristics, such as cooperation, social play and asymmetric design. Even so, there are others such as forcing local multiplayer and including the use of physical objects that have also been motives for study.

There are many gaming websites where you can participate with other players and offer many tools for social interaction, but the analysis of the three-month archives was the generation of the transient era and people did not talk to each other [eleven]. Another reason for this feature was also refuted by a study [12] and this time, my personal experience, since the study deals with social interactions in World of Warcraft, a massive online multiplayer game that can fly since 2007 (11 years). The study shows that the prevalence and scope of social activities in MMOGs may have been previously overestimated, and that gambling communities may have important movements that affect their cohesion and eventual longevity. My personal experience shows me that socialization within the game has reached the point of being difficult. At the beginning, it was normal to be in the zones and to be helped with the personal way, this means that we asked for a person that they were on the way, or in the chat. And it was something

normal to have chats while playing because getting the objective taken a while. Nowadays, if the player gets stuck he just have to click a button to look for a group of random players that are in the same point. After finishing, everyone leaves the group and follows their own, and of course during the mission is not usually talk. Not even in the dungeons, the difficult areas of the game, is normal to spoke. Now dungeons get rushed, and if a player is left behind could be even kicked from the group. Before there was a feeling shared by all the players of belonging to the same world and being part of something greater than themselves, what they really had to face together. Now the feeling is in a hurry at all times and that instead of collaborating, they prefer to keep on running. Of course there are players who still take walks and walk at night to enjoy the scenery, or upload their character without haste and enjoying at their pace. Players who are always there to lend a hand. But they are a clear minority, and this only makes it clear that: Multiplayer is not the same as Cooperative.

The last characteristic point of my project is the interaction with physical objects. An important reason for this is to remember a board game. These games are always associated with social concepts, sharing fun, friends ... while videogames are still wrongly related as causes of isolation and even violence. But there are also studies [13] that defend how the physical interaction between players can create new tensions and interesting situations of both cooperation and competition.

1.8. INSPIRATION

Several concepts of this project have been inspired by other games and films. In this chapter each one will be defined.

- **Game Concept**
  The major inspiration of this project is the game *Sea of Thieves*, where players takes the role of a pirate and go across the seas looking for treasures. This video fragment shows all the entire parts of this project gameplay. Being these: Choosing an isle – Searching the isle in a nautical chart – Taking the ship to the isle by controlling each ship part separately - Find the treasure.
  (https://www.youtube.com/watch?v=5a2fofoTTDI&t= | From minute 13:20 to 26:00)

- **Game dynamic and buttons orders/buttons mechanic**
  The ship phase of the developed game has been inspired by other game named *Space Team* (https://www.youtube.com/watch?v=y3fsvKnIVJg&amp=&t).
  In *Space Team* players must follow and give orders in order to scape with a space ship. The main difference with the developed game is that in *Pocket Crew* only one player, the captain, will give orders to all others in contraposition with *Space Team* where all players give and follow orders at the same time, making the gameplay confusing by having four people speaking at the same time.

- **Using of printed resources**
  The use of the isle map and the nautical chart as printed resources has been inspired by *Keep Talking and Nobody Explodes.*
  (https://www.youtube.com/watch?v=kkdbzFV1NoQ)
  In this game one player plays on the PC while rest of the players read a physic manual in order to help the player on the PC to deactivate a bomb.

- **Landscape composition and relation with the isle map**
  Landscapes from this video of *El Dorado* film are close to what type of composition and elements representation is aimed to achieve in this project.
  (https://www.youtube.com/watch?v=KWs00YVECmc | From minute 2:00 to 2:15)

- **Kind of pirate riddle**
  This video shows the type of riddles what is aimed to write for this project.
  (https://www.youtube.com/watch?v=5a2fofoTTDI&t= | At minute 2:34:50)

2. GAME DESIGN DOCUMENT

2.1. OVERVIEW

2.1.1. THEME | SETTING | GENRE

The game Theme is a pirate treasure seeking. The setting is an ocean, on board a pirate ship and mysterious isles which hide treasure. The genres are Social and Puzzle game.

2.1.2. TARGET AUDIENCE

The audience this game is planned for are people from 8 years old. Basically everyone can enjoy and play this game, but there is a minimal skill needed to react in a short time, and this could be complicated for old age people. Also, too young children could have troubles understanding the riddles or the game mechanics. Beyond the age, the target which this game is designed for are gamers who enjoy board games and have fun playing with friends. Despite of this, solitary gamers or people who does not tend to play board games can also enjoy this game.

2.1.3. GAME ELEMENTS

This project is a social experience formed by three elements showed in Figure 1. Two physical elements as they are the isle map (Figure 2) and the Nautical chart (Figure 3), and one android application (Figure 4).

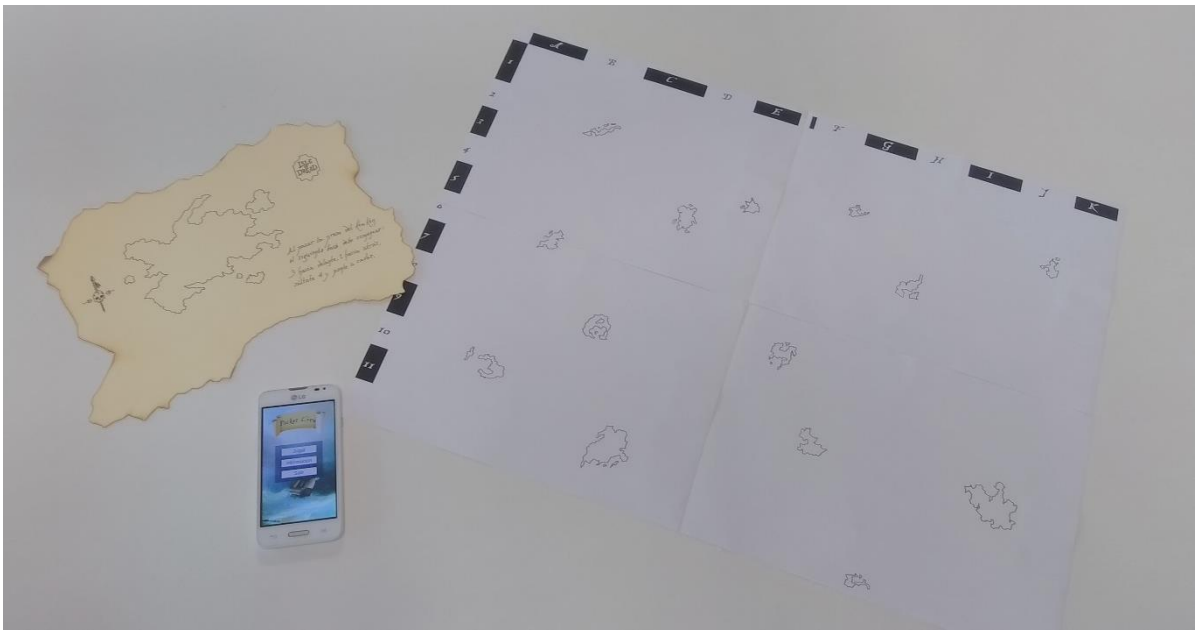


Figure 1: Game Elements

Figure 2: Isle map example

Maps contain clues for the treasure and allow players to seek the isle silhouette on the nautical chart in order to get its coordinates. The clue here says: *"After passing the kraken dam, the next dance should start: 3 forward, 1 backward, jump 4 and start digging".*

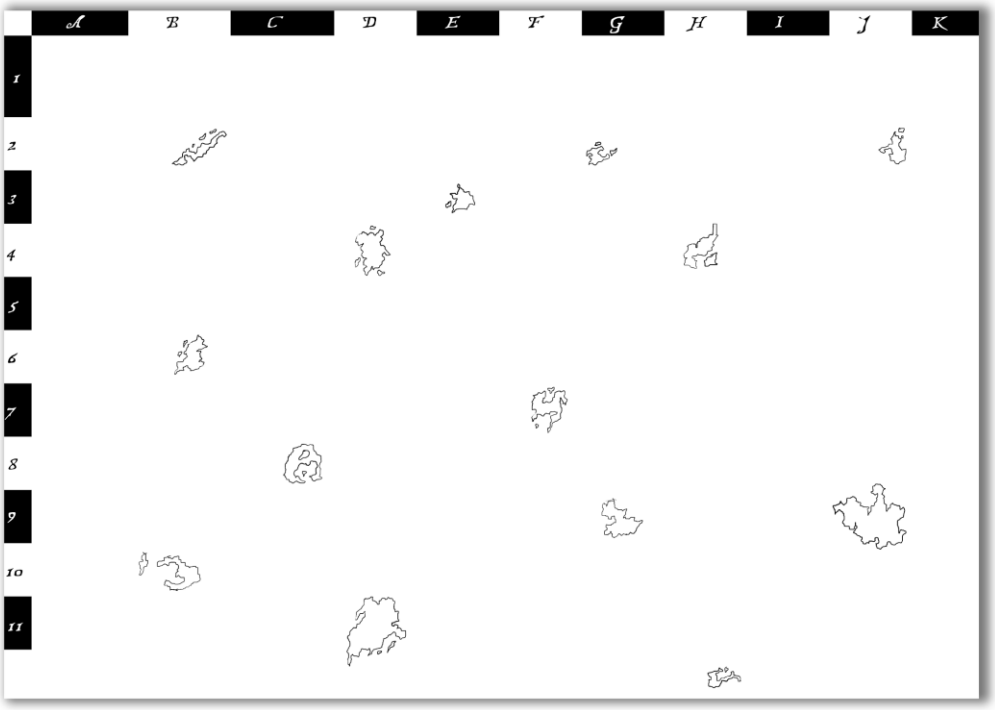


Figure 3: Nautical chart

The nautical chart has an A2 size, although it can be printed on 4 separate sheets and be assembled later. Here all the isles appear, including those that are represented on the maps. Players must find the isle of the selected map and obtain its coordinates with the format “NumberLetter” ex: 9G.

Figure 4: Pocket Crew app

In the application players will connect with each other, enter the coordinates of the isle to create a game, take the ship to the isle and go through the landscapes to find the treasure. At the landscapes phase players will need to share the isle map again in order to solve the riddle printed on it. If one player finds the treasure, all of them win.

2.1.4. GAME FLOW SUMMARY

Figure 5 shows the GameFlow scheme. Each player will start at the Main Scene, where is the Game Title and the main menu. After tap on “Jugar” the island selection scene appears. Entering the correct coordinates, a trip to that island is created and that player will go to the lobby. Other players can click on the button "Unirse a viaje" and select it from the list of trips to go to the lobby with the player who created it. Once everyone is Ready, the game starts and all of them except one go to the Ship Scene, the other one goes to the Captain Scene. If they are successful driving the ship the isle will appear, if they are not, Game Over Scene. Tapping on the Retry button, game starts from the beginning. Once at the Isle players will be able to move between different scenes named Landscape 1, Landscape 2, etc... After find the treasure in one of the landscapes Win Scene is shown to all players and the game ends.

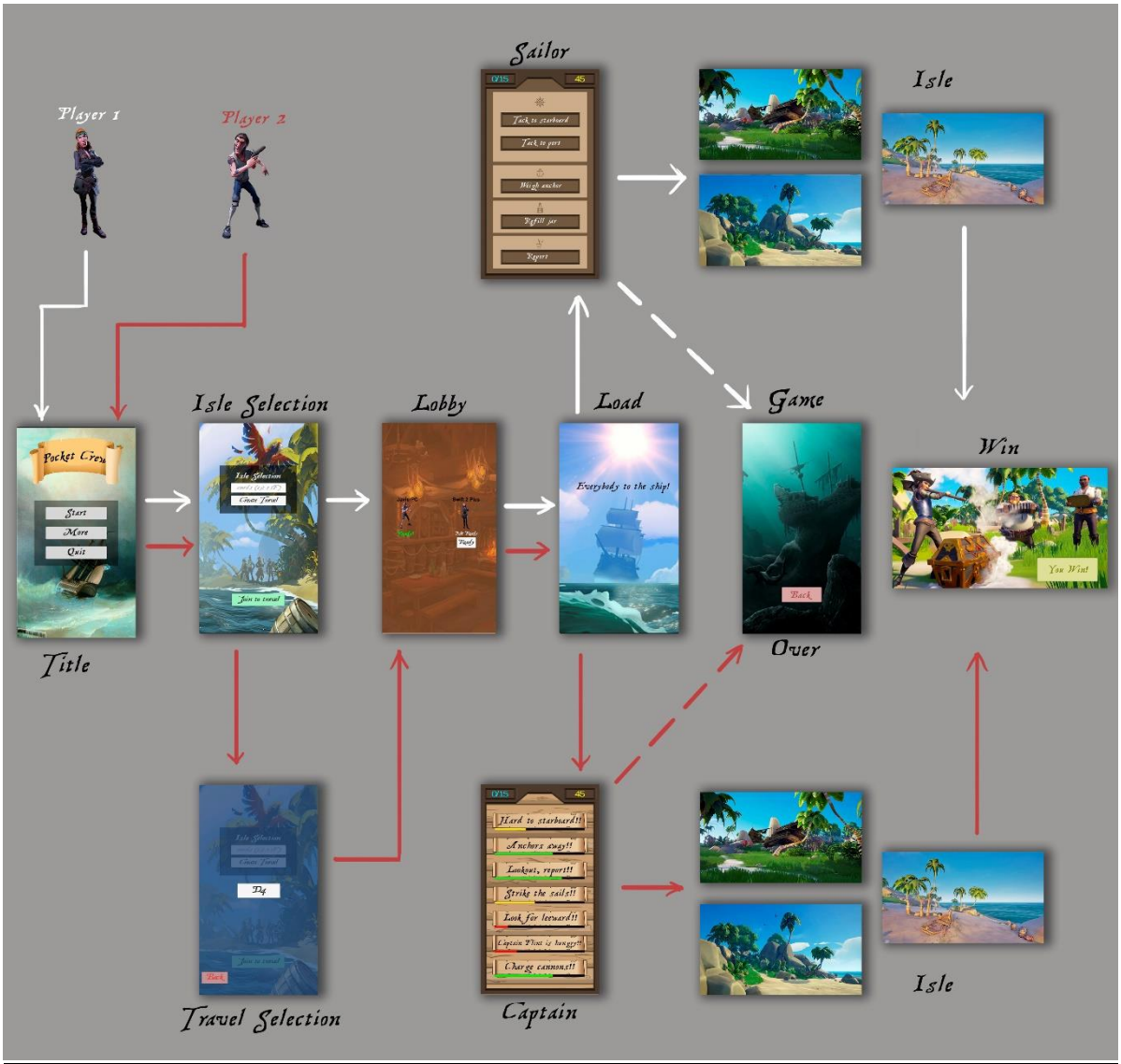


Figure 5: Game flow

2.1.5. CORE GAMEPLAY MECHANICS (Brief)

The core mechanics are to choose and share the map, to find the isle in the nautical chart, to follow the captain's orders and to cooperate in order to decipher the treasure clues. The key is that core of the game maintain a board game spirit, making players talk and interact with each other's.

2.1.6. TARGETED PLATFORMS

The main platform is Android. Could be iOS or Windows Phone in a future. This game is meant to be played only in smartphones and tablets.

2.1.7. LOOK AND FEEL

The expected visual style is a "soft cartoon", like a low poly cell shading in 2D. With low saturation and pleasant contrasts. Figure 6 shows a reference image from the art of "Sea of Thieves":

Figure 6: Sea of Thieves Art reference

All game graphics showed in this documents are placeholders, they have been generated using resources from "Sea of thieves" and the internet in order to set the element schemes and the intended style. Original art for the game will be developed in a next phase of the project. In this chapter those schemes and style decisions will be explained.

Figure 7 shows a graphic style test. In order to decide the final graphic style of the game a low poly art was generated from a reference image. Low poly was discarded and it was decided that the expected style get closer to cartoon. With low poly style could be hard to differentiate the elements at the isle, making the riddle even harder to solve. This test is showed with higher resolution at the following link: https://imgur.com/a/VqOm3b7 .

Figure 7: Low poly art test

Next, each screen composition and elements scheme will be explained. Examples for all the application screens are showed with higher resolution at the following link: https://imgur.com/a/8iWwkAC .

Figure 8: Title screen scheme

Figure 8 shows the scheme of the Title screen elements. There are expected to be a label with the game name: “Pocket Crew” and a graphic which contains three buttons. The theme of these object is planned to be “pirates”.

Figure 9: More screen scheme

Figure 9 shows the More screen scheme. The screen is planned to show a message about the project. Background image and the name label are the same as in the Title screen. It is expected to be a graphic containing the informative text, the “More” label and the “Back” button to return to Title screen. This graphic’s style will be the same as the graphics used in the Title screen.

Figure 10: Isle Selection screen scheme

Figure 10 shows an example of the Isle Selection screen. Background image will show part of an island. Elements in this screen are the "Join to travel" button and the "Isle Selection" graphic. This graphic's style will be the same as the previous screens, also for the buttons. All buttons in the game will maintain this style unless it is specifically mentioned.

Figure 11: Travel Selection screen scheme

Figure 11 shows an example of the Travel Selection screen scheme. The background in this screen it will be expected to be a translucent colored layer over the Isle Selection screen. The elements here will be a button to return to the Isle Selection screen and the Travels, which will appear at the center of the screen. Each travel will show a label with the coordinates from which it was created and they are expected to have a pirate graphic style different from regular buttons.

Figure 12: Lobby screen scheme

Figure 12 shows an example of the Lobby screen scheme. Players are planned to appear at the middle of the screen. Each player will show the name, a unique image of a pirate, the status (Ready/Not ready) and the button which change the status. The background image is planned to represent a pirate rest zone, like a tavern.

Figure 13: Loading screen scheme

Figure 13 shows an example of the loading screen. Loading screen is planned to be showed for only two seconds after the Lobby, before the game starts. The screen only will show a text “Everybody to the ship!” over a background image of a ship sailing across the sea.

Figure 14: Captain screen scheme

Figure 14 shows the Captain screen scheme. This screen will be only visible for captain player and there always will be seven different orders in screen. It is composed of three elements: HUD, background and the orders.
HUD is explained in chapter 2.5 (2.5.1. Visual System). Background image (Figure 15) is planned to be a "pirate style" frame with a wooden planks background. Each order (Figure 16) will seem like it is written over a paper and a priority slider will appear under it. The color of the slider is expected to change when it decreases.

Figure 15: Captain screen background

Figure 16: Order example

Figure 17: Sailor screen scheme

Figure 17 shows an example of the Sailor screen. All players who are not captain will see this screen. Like in the captain's one, there will be a HUD, a background and the ship parts. Background image is the same as in the Captain screen, this time without wooden planks.
Ship parts are composed of a graphic which contains the icon and the action buttons. The style of this graphic and buttons is expected to be similar to the background frame's one.

Figure 18: Ship part example (Anchor)

Figure 18 shows the graphic for the anchor as a ship part. The graphic shows an icon of the ship part and a button with the pertinent action below it. Some ship parts, like the rudder have two action buttons.

Figure 19: Game Over screen scheme

Figure 19 shows an example of the Game Over screen. This screen will show a shipwreck and a "Quit" button.

In order to find the treasure with map showed in (Figure 2) three different landscapes were planned for that isle. Next each one will be explained. The showed landscapes were obtained from images of the game "Sea of thieves" and the comparison from the original images can be checked at the following link: https://imgur.com/a/HBQudto .

All landscapes share same graphic style and it is important that the lightning being similar for each one at the same isle. One isle could have all the landscapes with night lightning and other isle could have them with day lightning, but never an isle could have one landscape with day lightning and other with night lightning. This could be confusing for players.
Another important aspect of the graphic style is that each element in the landscape must be easily recognized and differentiated from the rest.

Figure 20: Isle Landscape 1 scheme

Figure 20 shows an example of the first landscape which players will see when they reach the isle. This landscape is planned to do not contain nothing in particular and it can show anything.

Figure 21: Isle Landscape 2 scheme

Figure 21 shows an example of the landscape which contains the kraken's prey (*"la presa del kraken"*) mentioned in the riddle of the map. The process of generating this image can be checked at the following link: https://imgur.com/a/Sd0qVwB . The only important thing in this landscape is that it must contains a shipwreck. The composition of this landscape must guide the player's attention to the shipwreck.

Figure 22: Isle Landscape 3 scheme

Figure 22 shows an example of the landscape which contains the treasure. There only two important things with this landscape are: Make it appears after the shipwreck's landscape and making it to contain six palms with an image composition which make them to stand out for the players.

To explain the importance of the mentioned above, the riddle will be deciphered.
"After passing the kraken dam, the next dance should start: 3 forward, 1 backward, jump 4 and start digging".

"After passing the kraken dam" indicates that treasure will be in the landscape next to the landscape with the shipwreck (Krakens eat ships). In that landscape, this dance is supposed to start *"3 forward, 1 backward, jump 4 and start digging".* There are 6 elements jumped at the dance, exactly the same number of palms in foreground, this way, if players tap under the 6th palm they will find the treasure.

Figure 23: Win screen scheme

Figure 23 shows the Win screen scheme. The visual style is planned to be the same as in the landscapes, showing a background image with several pirates finding a treasure. The screen is also planned to have a button to quit the game.

2.1.8. CORE GAMEPLAY MECHANICS (Detailed)

This project seeks to create an experience of collaboration and social interaction not only within the game but also outside of it. All players will have to share the treasure map and the nautical chart in order to find the isle coordinates. On the ship, each player will take randomly the role of a crewmember, the screen of each one will be filled with random parts of the ship and they will have to follow the Captain's orders. The Captain's screen will be filled with orders instead of ship parts, each one with a priority, so this will be a timed phase. If captain says: "Sails up! Arr!", the crewman with the sails in the screen will have to perform the order. In this way they will reach the island through collaboration. Once there, landscapes of the island will appear in their devices, being able to go from one to the other. They will have to share again the map and try to decipher the place referred on the clue. At the time that someone tap the correct place on the screen, they will find the treasure.

2.1.8.1. Sharing the map and the nautical chart

First of all, players must discuss which map they will use. All isles are in the game, but each map begins a unique adventure with unique clues and treasure. When the map is chosen, all players must share it in order to find the isle in the nautical chart. By introducing the isle coordinates on the isle selection screen, the adventure will start.

2.1.8.2. Following Captain's orders

Once on the ship, all player must pay attention to what captain is saying, one of the players will take this role and the screen will get full of orders to the crew. Each order will have a priority, if time expires the order will be marked as failed, so the captain must be quick and the crew must obey even faster. The crew members will see different parts of the ship so never is known who is responsible from which order.

There will be a different number of parts in screen, depending on the number of players. Keeping in mind that one player must be Captain:

Table 6: Ship parts - Players relation

Players	Parts in screen
4	4
3	6
2	12
1	X

Parts in screen = Total Parts % (Number of players - 1)

There are 12 different parts of the ship and 16 possible orders:

Table 7: Ship parts - Orders relation

Ship part	Order
Rudder	Hard to starboard!! Hard to port!!
Anchor	Anchors away!!
Captain's jar	Bring me more Rum!!
Lookout Top	Lookout, Report!!
Mooring	Ropes away!!
(Sails up / down) Mast	Strike the sails!! Hoist the sails!!
(Sails to windward / leeward) Pulley	Look for windward!! Look for leeward!!
Cookie for the parrot	Captain Flint is hungry!!
Bucket	Scoop out the water!!
Cannon	Charge cannons!! Fire!!!
Wooden plank	Board up those holes!!
Map	Where's the map!

Figure 24: Sailor screen with four ship parts

Figure 24 shows an example of the sailor screen in a game with four players (1 captain, 3 sailors). 12 ship parts are divided equally among the three sailor, showing each sailor 4 ship parts in screen. In this example these are the Rudder, the Anchor, the Jar, and the Lookout Top in this order.

2.1.8.3. Solving the clues

At the isle, players will need to go back to the treasure map and share the clues. By exploring the different landscapes and discussing possible solutions they could be able to tap on the correct place and find the treasure. They will receive a penalty if they touch the screen randomly, so it is important to think where the treasure could be. Also, players could need to collaborate by asymmetric design mechanics, for example, one player must press a wall in a landscape, while in other landscape, another player must pull from a rope at the same time. This could activate a mechanism and bring the treasure one step closer. Final objective is always the same, tap on the correct place and the treasure will appear.

2.2. GAMEPLAY

The gameplay consists basically on tapping the appropriate button. Sailor must press the button related with the order that the captain has just given. Although this mechanics is simple, the game is enriched by social interaction, which is also part of it. First players must find the island and its coordinates. Then, while the sailors must press the buttons, the captain owes them the orders. And finally, once on the island, players must collaborate to solve the riddle. Although the game within the application is quite simple, the experience as a whole is what matters. The project purpose is to encourage collaboration and social gaming through asymmetric design, by reducing the in-game mechanics' complexity it is assured that all players understand it, allowing them to focus on the social experience.

2.2.1. PUZZLE STRUCTURE

As explained above, the complete set of experience can be divided into three distinct parts:

- Find the coordinates of the island among all
- Take the ship to the island
    - Shout orders in the proper order according to priority
    - Press the appropriate button
- Solve the riddle to find the treasure

2.2.2. OBJECTIVES

While the parts of the island and the treasure are defined by itself. In the part of the ship it is necessary to clarify that the objective of the players is to successfully complete a minimum number of orders before the time runs out. If the wrong order is executed, it will be penalized by subtracting an order from those already completed.

2.3. MECHANICS

In this game's world, before you start an adventure you need to choose the place (isle). Once with the image of that place, you need to find it in a map, in order to be able to reach it. When you have the route (coordinates.) to your adventure, you need to arrive to the place, and in a pirate world, this means to drive a ship across the ocean. If you arrive to the isle, the collaboration of all the crew will be needed, and the exploration of the entire isle is something highly recommended.

2.3.1. OBJECTS

Basically all intractable objects of the game are ship parts. Which are basically buttons to perform the different orders.

The isle landscapes at the final part, there are a little area on one of them to players to tap on and find the treasure.

2.3.2. ACTIONS

Individual actions that will be carried out in the game:

- Choose one of the available maps
- Collaborate to find that island on the nautical chart
- Create a trip by entering the coordinates of the island
- Join the journey that a player has created
- Decide the order in which to carry out the orders and tell them to the rest of the group
- Give the button that fulfills the order that the captain has just said
- Collaborate to solve the riddle of the map while changing the landscapes of the island
- Touch the screen at the point where the treasure is

2.4. LEVELS

It is important to specify that there are no levels in this game. Each map allows players to seek one isle, which have its own treasure. All the gameplays start with the seeking of the isle and end when the treasure is found. Instead of levels there are three different phases: 1st seeking the isle, 2nd driving the ship and 3rd solving the clues and finding the treasure.

2.5. INTERFACE

2.5.1. VISUAL SYSTEM

Figure 25 shows the HUD, which is visible only in the ship phase. It is composed of a timer (yellow) and a counter of the accomplished / minimum orders (cyan). If timer gets zero and the accomplished orders have not reached the minimum it is a game over.

Figure 25: HUD

2.5.2. CONTROL SYSTEM

All controls are tap on the screen.

2.6. EASTER EGGS

2.6.1. PARROT'S NAME

This game parrot is called the same as Captain Silver's parrot, from "Treasure Island" book. Written by Robert Louis Stevenson.

3. DEVELOPMENT

The concept for this project has been planned from other initial ideas which were changed through the time.

The first idea was an initial proposal of the advisor of this: *"Development of a videogame for adults and seniors that encourages active play and exercise (e-fitness) in the open air"*. This was the idea from which the project was started. Right at that moment Nintendo published Labo, which inspired the idea of opting for a more social type of game and using objects with which to interact. After not finding facilities to develop for Switch or using the joy-con in an innovative way, the idea was raised of approaching a social game using more traditional elements, which is increasingly forgotten. And so the project approached the board games, which along with the asymmetric design and the articles referenced in the proposal (1.7. Related Literature) has concluded in the actual document.

3.1. PHYSICAL PROTOTYPE

At the beginning of the project a paper prototype of the game was created to determine the game dynamics (Figure 26). The prototype consists of a map, a nautical chart and several cards showing the different screens. There was made four sets of eight different cards, for up to four players.

Figure 26: Physical prototype

When this prototype was devised, the search of the island was thought as part of the application. After the lobby, the nautical chart will appear on the screen of each device and the players need to moving around it. The first one to find it, would touch it and the ship's screen began. This decision changed for two reasons, the possibility of increasing physical interaction between players by seeking in a real card that everyone should share. And the unity matchmaking system use in the project, which integrates perfectly with this mechanic.

This prototype was tested in several sessions simulating one gameplay, whose results are the following:

Table 8: First physical prototype test

Players	Feedback	Comment
1 (Woman | 25 years old | Little game experience)	At the riddle player could not know which direction is *"forward"* and *"backward"*.	-

Table 9: Second physical prototype test

Players	Feedback	Comment
1 (Woman | 24 years old | Medium game experience)	She liked the game and solved the riddle without help.	-

Table 10: Third physical prototype test

Players	Feedback	Comment
1 (Man | 43 years old | High game experience)	It should be a HUD to see if we will reach the isle soon or not. Also there should be able to play no matter the number of players between two and four.	The lookout will be able to activate the HUD by accomplishing the lookout order. Also the number of ship parts in screen will be proportional to the number of players. This way one sailor can be able to accomplish all orders in two players mode.

Table 11: Fourth physical prototype test

Players	Feedback	Comment
2 (Man | 29 years old | High game experience) (Woman | 24 years old | High game experience)	Riddle is too difficult. What happens if a sailor press an order button at the wrong time?	A clue button could be added. If a button is pressed at the wrong time, the accomplished orders score decrease by 1.

3.2. SETTING UP THE UNITY PROJECT

The basic structure of this project consists of two scenes ("lobby" and "Game") in which panels are distributed that activate or deactivate depending on the state of the game. Figure 27 shows the different game screens.

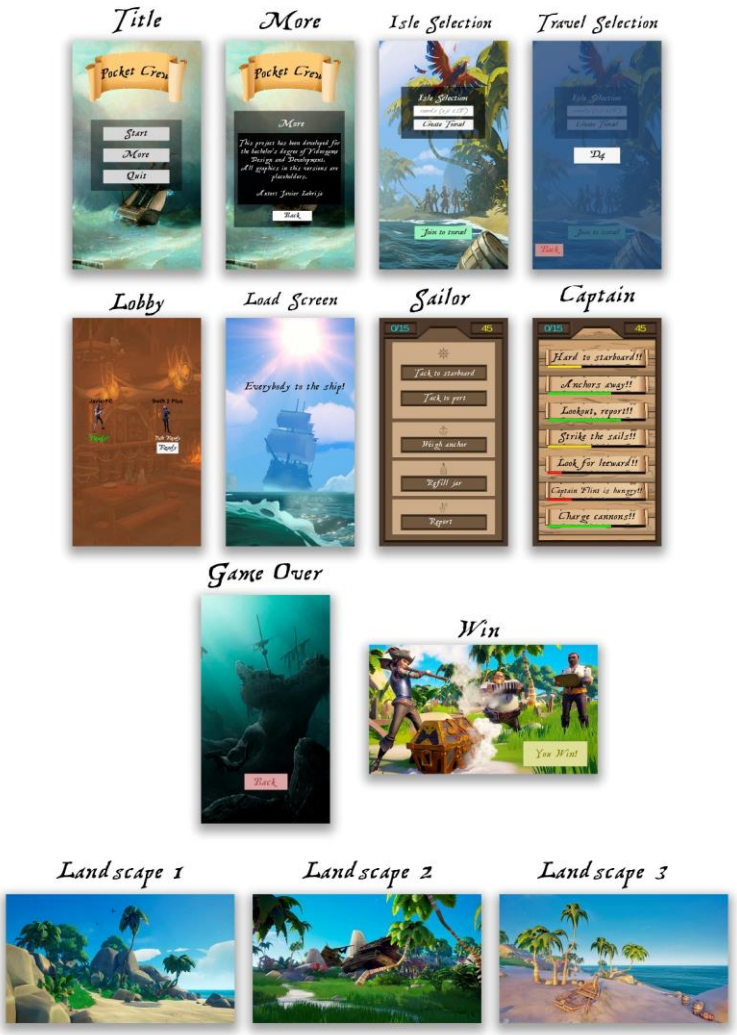


Figure 27: Game Scenes

Figure 28 shows Lobby scene hierarchy:

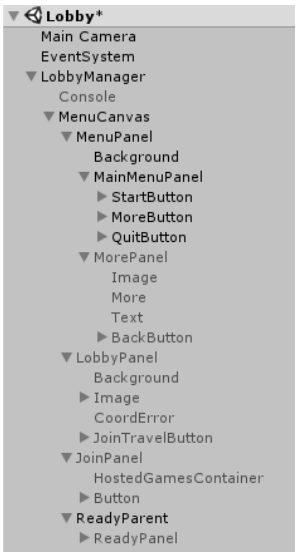
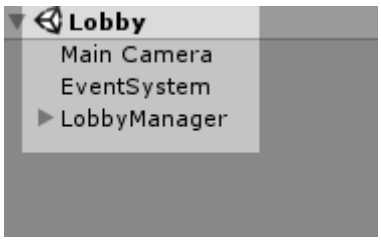
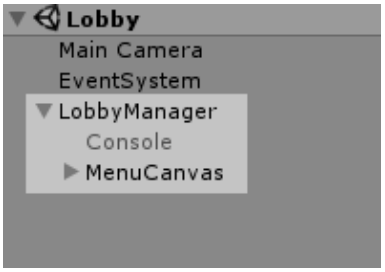


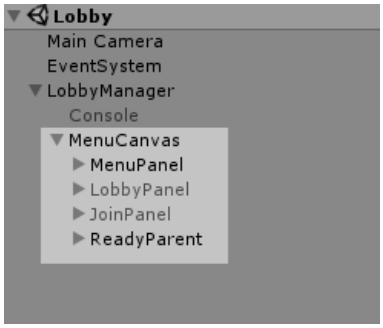
Figure 28: Lobby scene hierarchy

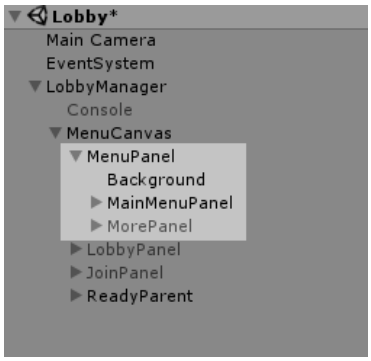
It is important to mention that the elements that are below in the hierarchy, are ahead of the rest on the screen. The order of each element is not random in addition to the state (active / inactive). Next, we proceed to explain the scheme of these elements one by one. To follow the movements between the elements it is recommended to complement this table with *figure 5: Game flow* as well as *figure 32: Lobby Scene classes*. Implementation details are explained at chapter 3.4.1.1. The isle selection and entire chapter 3.4.2. Connecting to the lobby.

Table 12: Lobby Scene Scheme

Lobby Scene

The Lobby scene consists of the camera, the *EventSystem* and a *LobbyManager* object containing the *NetworkLobbyManager* script. This is the script responsible for managing the game's network.

Lobby Manager

The *LobbyManager* object contains another *Console* object and a *MenuCanvas* canvas. The console object simply contains a script that shows the logs on the screen of the application, to be able to see them on the mobile device. The *MenuCanvas* is the canvas that contains all the panels that are used in the scene.

Menu Canvas

Within *MenuCanvas* we have three panels and an object. The panels are *MenuPanel*, *LobbyPanel* and *JoinPanel*. The object is *ReadyParent* which will be explained below.

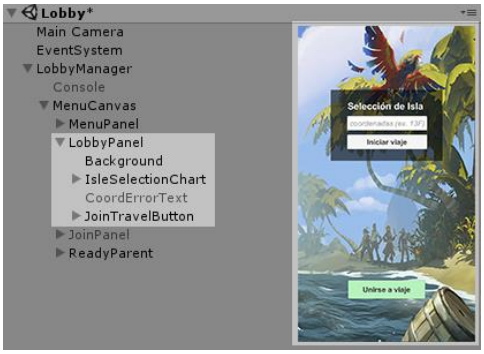
Menu Panel

MenuPanel consists of a background image and two panels: *MainMenuPanel* and *MorePanel*

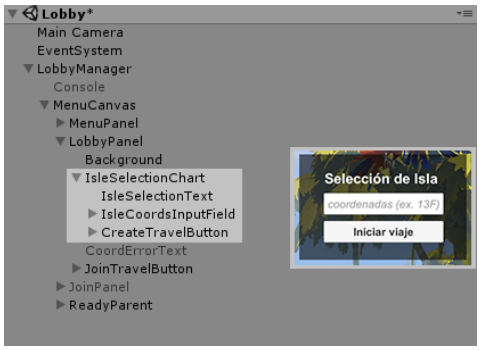
Main Menu Panel

MainMenuPanel contains the buttons showed at start of the game. Specifically *Start, More* and *Exit*. *Exit* quits the application. *Start* activates the *LobbyPanel* panel that is discussed later. *More* disables this panel and activates *MorePanel*. In the case of *LobbyPanel*, it has an image that completely covers the current panel, but *MorePanel* shares background with *MainMenuPanel*, so if we do not deactivate *MainMenuPanel*, the elements of both panels are overlapped.

More Panel

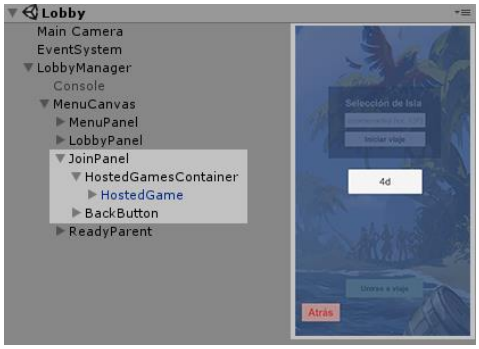
MorePanel shows the information screen, composed of a text *MoreLabel,* another *Text* that stores the description that is displayed, an image of a *BlackChart* that acts as background for the text and a *BackButton* that when pressed, disable this panel and re-activate *MainMenuPanel*.

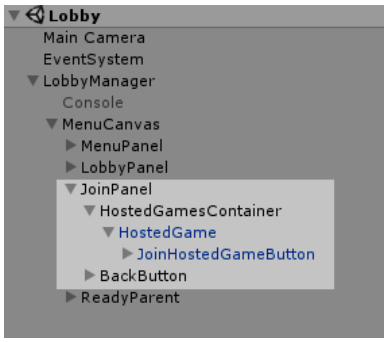
Lobby Panel

LobbyPanel has assigned the *CreateTravel* script. It consists of a background image, an *IsleSelectionChart*, an error text and a *JoinTravelButton*. Pressing the "*Unirse a viaje*" button activates the *JoinPanel* panel and executes the OnJoinTravel( ) method of the script. This method communicates with the *JoinRoom* script to update the list of created games.

Isle Selection Chart

IsleSelectionChart consists of a text Island selection, an *IsleCoordinateInputField* and *CreateTravelButton*. When the button is pressed OnCreateTravel( ) is executed in the *LobbyPanel* script and activates the *ReadyPanel* panel. This method checks whether or not the coordinates that we have entered in the input are in the list of possible ones. If they are, this client becomes Host and the Hosted Game of those coordinates is created. If not, an error message is displayed on the screen. (3.4.1.1. The isle selection, explains the implementation in detail)

Error Message

This message is displayed when the coordinates are wrongly entered to create the trip. It is a text that is activated from the *LobbyPanel* script.

Join Panel

JoinPanel only contains a button to deactivate this panel, making the *LobbyPanel* is seen again, and an empty object, which acts as the parent of all the games created (HostedGames). For this was assigned a *JoinRoom* script that is in charge of searching all the games created by the matchmaking system.

Hosted Game (Prefab)

HostedGame is a prefab that contains a button with the *HostSetup* script assigned. This object is the representation of a created travel. The text of the button is the coordinates of the island with which it was created, and if the button is pressed, the Join( ) method of the script is executed, which uses the matchmaking system to connect to this client in that game.

Ready Parent

ReadyParent is nothing more than an empty object that contains a *ReadyPanel* panel. This panel is very important that is inactive, and the reason of *ReadyParent* is to allowing us to access *ReadyPanel* in order to activate it after joining a game. This is necessary because *HostSetup* needs a reference to *ReadyPanel*, but being an object instantiated during execution, there is no way to give it to it from the beginning, so it accesses the panel from its parent.

Ready Panel

ReadyPanel is the Lobby screen itself. Where all the connected players are shown. As with the HostedGames, it is only composed of an object that acts as the father of each instantiated *LobbyPlayer.*

Lobby Player (Prefab)

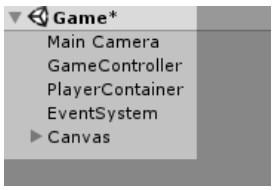
LobbyPlayer is a prefab composed of an image for the player (parrot), a *Name* text for the name, which is received by the code of the device itself. A *ReadyText* for the state (**ausente** / **¡Listo!**) and two buttons. The buttons are deactivated and activated by the code with the instruction "if (isLocalPlayer)" so that only the local client activates the button of that prefab. This makes each player only see his *ReadyButton* on screen. When pressed, change the status through the network, as well as the *ReadyText*. Also activate the opposite button *NotReadyButton*, being able to undo the action and return to the previous state. The script which controls this object is *LobbyPlayer* script.

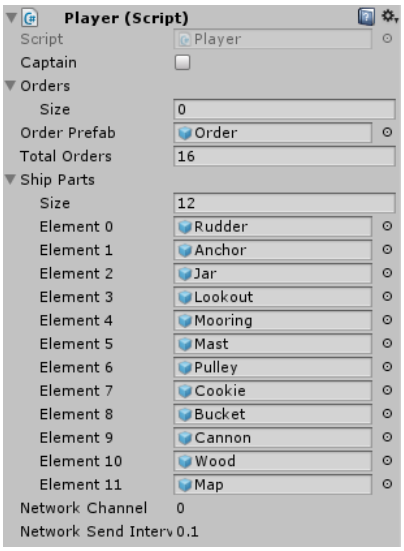
Now the Game scene will be explained. Figure 29 shows game scene hierarchy:

Figure 29: Game scene hierarchy

To follow the movements between the elements it is recommended to complement this table with *figure 5: Game flow*, as well as *figure 33: Game Scene classes*. For implementation details chapters 3.4.3 up to 3.4.7 are recommended.

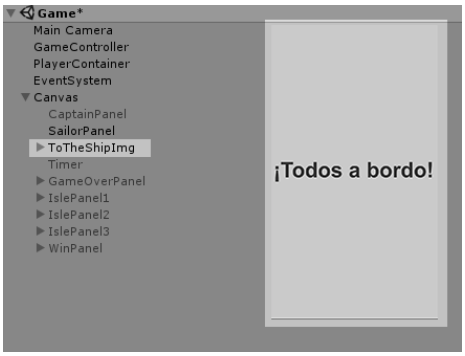
Table 13: Game Scene Scheme

Game Scene

Game scene has a camera, an event system, a *PlayerContainer* which who come from lobby (*Player* prefab). A *GameController*, with the *GameController* script assigned, which controls the general aspects of the game, such as the timer. Only server instantiated objects have authority to send a command, so *GameController* will sometimes call Commands through a Player's methods. Finally, there is a *Canvas*, where are all the panels to be used.

Player (Script)

It is not the same *Player* than *LobbyPlayer*. When *NetworkLobbyManager* starts the travel, it instantiates a *Player* prefab for each client. That prefab has the *Player* script. It is important to assign the prefab *Order* and especially *Each part of the ship*. This is because the orders are instantiated through the code from the prefab but ship parts are unique, and the server uses this list to tell each player what of these prefab must activate. As a result, the captain always has 16 instantiated orders but only 7 active ones. While the sailors will only have instances of the parts that correspond to them.

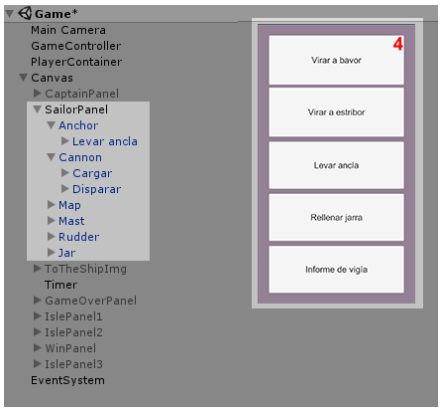
Canvas

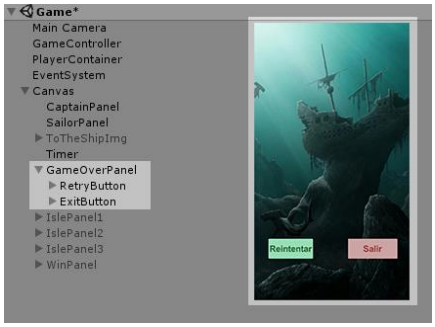
Canvas has seven panels, an image for the loading screen and a text which shows the timer. It is important the order in hierarchy and the initial status (active / inactive) of the elements.

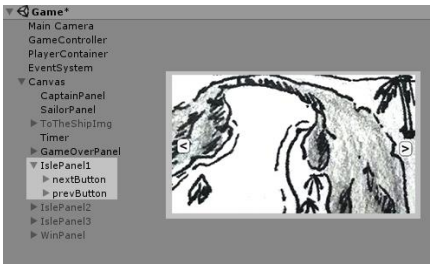
To The Ship

As soon as the Game starts, the loading image *ToTheShipImg* appears. This image is deactivated after 2 seconds to give players time to prepare. Meanwhile, the server creates a list with all the players and randomly choose one of them to be the captain.

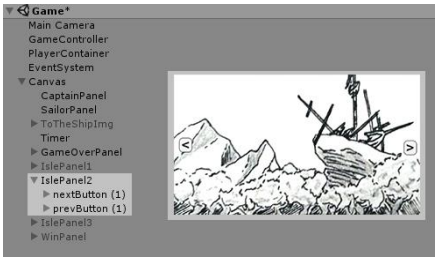
Captain Panel

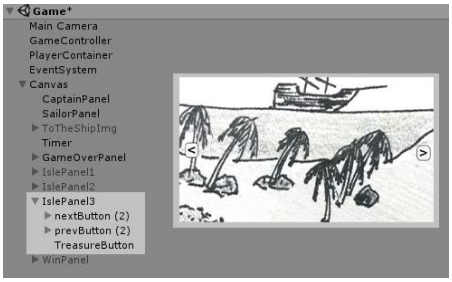
After the two seconds, the *CaptainPanel* and *SailorPanel* appear. Both panels must be active during the game because the server must access the parts of the ship, regardless of whether or not player is a captain. With the initial layout, *SailorPanel* would cover the *CaptainPanel*, so before activating *CaptainPanel*, *SailorPanel* is moved to the top of the list, by this mode it remains active behind and can still be accessed. In this panel all possible orders are instantiated, and 7 are activated randomly. When the slider reaches 0 it will disappear and another order get active.

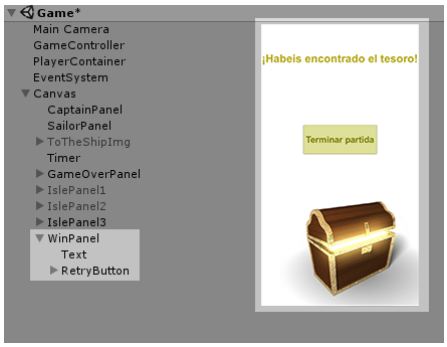
Sailor Panel

There is no need to care about the order of the panels for Sailor players. In this case, the server will tell each client which part of the ship will be instantiate. Each ship part is nothing more than an object with one or two buttons, which when pressed ask the server for check if the related order is active or not.

Game Over Panel

When time is gone, if the minimum of required orders has not been reached, the server says clients to active *GameOverPanel*. This panel has an *Exit* button, which quit the game.

Isle Panel 1

If, on the contrary, the minimum of the required tasks is exceeded, the first panel of the island will be activated, which shows the first landscape. The panels of the island only have two buttons, one to go to the next landscape and another to go to the previous one. This is done by activating and deactivating the corresponding panel.

Isle Panel 2

If the right button is pressed at the first isle panel, the previous one is deactivated and panel 2 is activated. It follows the same scheme as the previous one.

Isle Panel 3

In this particular panel there is a button that is not in the others, called *TreasureButton*, which is on the stone next to the palm is a hidden treasure and if a player clicks there, the server will tell clients to set *WinPanel* active.

Win Panel

WinPanel shows a congratulation message and a button that closes the game when pressed.

3.3. WORKING WITH AN EXTERNAL MULTIPLAYER LIBRARY

When starting the project, CaptainMess library [15], that facilitates the implementation of multiplayer for the type of game that is this project was used as reference. *SpaceTeam* is one of the games on which this project is based, and CaptainMess is the library which that game implements.

The mentioned library turned out to be based on the Unity Networking library (UNET) but developer implemented its own NetworkManager and other classes, even though Unity already facilities it. This caused many scripts with several inheritances among them, resulting in a mess and a resource with which it was very difficult to work.

After implementing the Lobby following this library and running the test which can be seen in Appendix 1 it was decided to stop using this resource, it was really difficult to find the bug in so many scripts and so many inheritances.

3.4. WORKING WITH UNITY MULTIPLAYER NETWORKING (UNET)

After two weeks working with the CaptainMess library, it was decided to implement a whole new multiplayer system. In this way will be easier to manage what makes each script and what methods there are. Which translates into more agility when it comes to work.

Before explaining the project, it is convenient to set some bases [17]. In a multiplayer game there is a client for each running instance of the code, which connect to a server that handles several aspects of the game and connects the clients to each other.
Unity uses High Level API [18], a set of instructions and systems for multiplayer games, which allows Dedicated Server (only acts as a server) or Hosted Server (server also acts as a client).
In this project the Hosted Server scheme is used. One player will create the game and the rest will join. The creator takes the place of Host, who acts as server and client at the same time. Each client also has a rest of clients, being the current machine execution the local client and the rest remote clients as is showed in Figure 30.

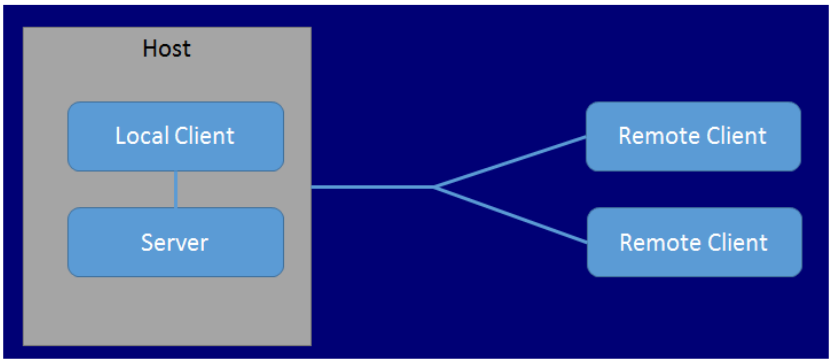


Figure 30: Hosted server diagram

Another important thing are the possible instructions. The local client of the host can communicate with the server directly through messages and methods, since they are running in the same instance. But the rest of clients need to use what is known as Remote Actions [19], showed in Figure 31.

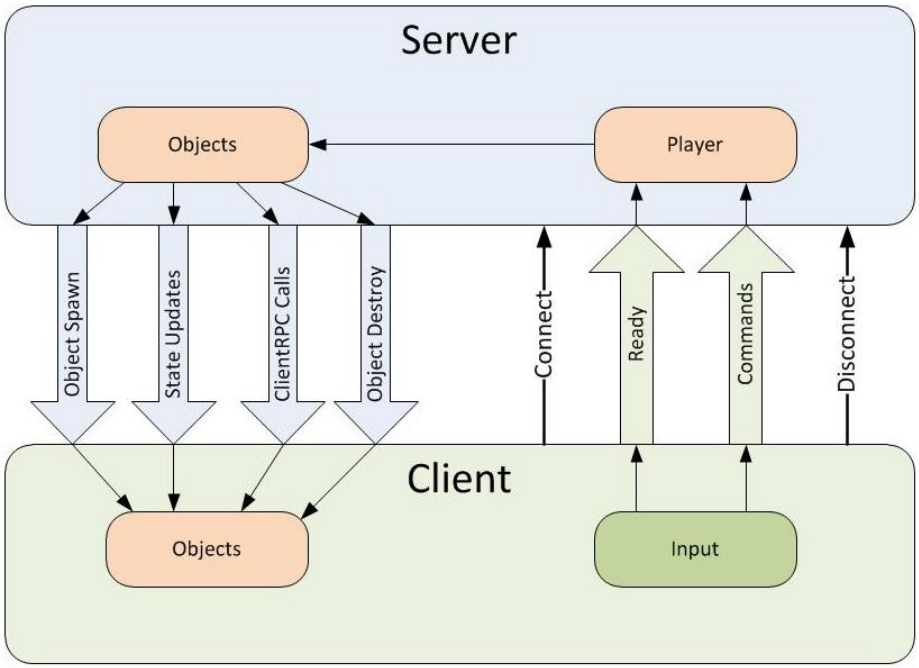


Figure 31: Remote Actions diagram

There are basically two fundamental types: [Command] and [ClientRpc]. Throughout the document examples of use will be shown. For now, it is only necessary to explain the basic concept:

> [Command] - Calls to methods that are made from the client. But they run on the Server, using the data of the client in particular.
>
> [ClientRpc] - Calls to methods that are made from the Server. But they run on all clients, using server data

Also [Syncvar] is used to synchronize variables. If a variable has this label and its value changes within the server, the value in the clients will be automatically updated.

Unity offers servers to host the games developed with the engine [20], to use it must access the concrete project from the unity website, activate the multiplayer service and enter the maximum number of players per room. As mentioned before, this service uses a matchmaking system, where clients connect by creating and joining games, all this without having to be in a local network. All that is needed is that the devices are connected to the internet.

Finally, any object that is instantiated by the server must have assigned a component called Network Identity [21]. This component is what allows the server to differentiate the different clients and detect them. In this project those objects are *LobbyPlayer*, *HostedGame*, *Player*, *Order* and all the *Ship parts*.

Below are the classes of the two scenes of the game Figure 32 shows the lobby scene classes and Figure 33 shows game scene classes:

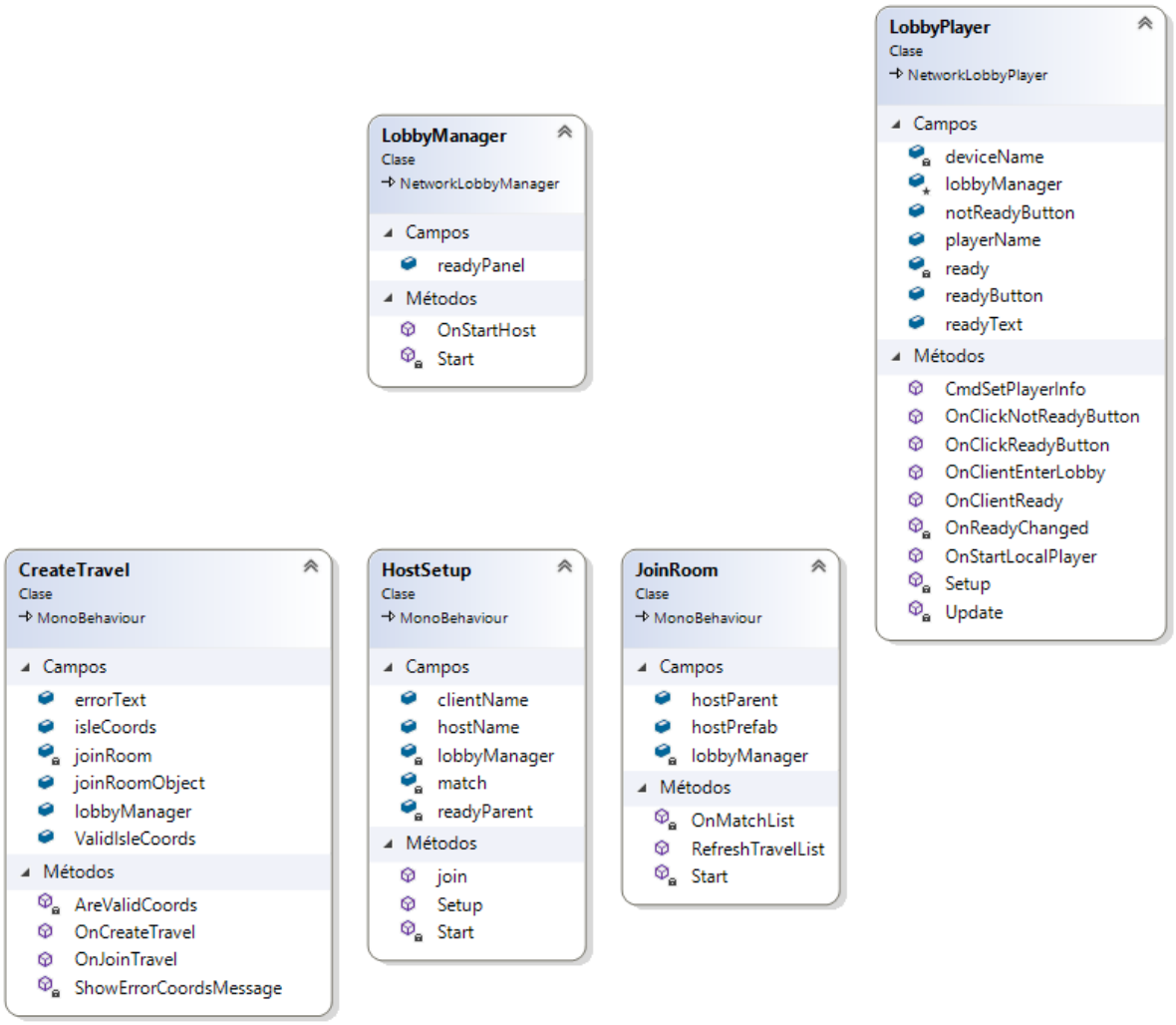


Figure 32: Lobby Scene classes

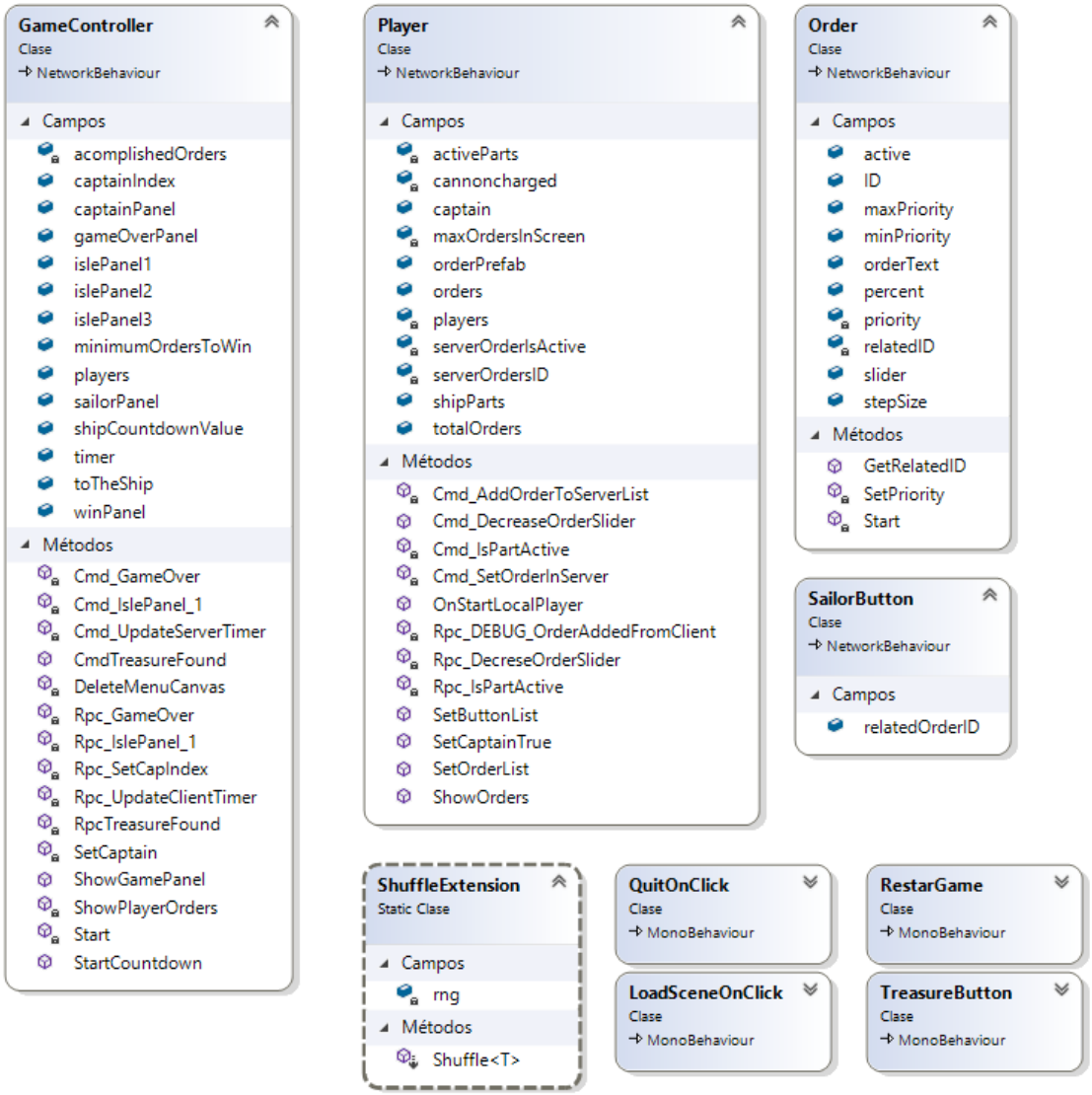


Figure 33: Game Scene classes

3.4.1. CREATING THE MAIN MENU

The first was the menu implementation, for this a unity tutorial was used [16] and as a result the panel scheme explained above was obtained.
Specifically, *MainMenuPanel* and *MorePanel*. When a button is pressed, a panel is deactivated or activated by changing the screen. In particular, if we press start, the game does not start, instead *LobbyPanel* is activated and the screen is changed.

3.4.1.1. THE ISLE SELECTION

To implement the isle selection, it was decided to use unity's matchmaking system, integrating by this way the creation of a game with the isle selection mechanic by introducing the coordinates.
LobbyPanel has the *CreateTavel* script, which is responsible for both creating a game and looking for those already created.
The script has a string array with all valid coordinate values. If OnCreateTravel( ) is executed and the input is not valid, the error text is activated for two seconds. When the coordinates are valid, the matchmaking system is started and the game is created with those coordinates as name. Figure 34 shows the OnCreateTravel( ) method.

```
public void OnCreateTravel()
{
    if (AreValidCoords(isleCoords.text))
    {
        lobbyManager.StartMatchMaker();
        lobbyManager.matchMaker.CreateMatch(isleCoords.text, (uint)lobbyManager.maxPlayers,
    true, "", "", "", 0, 0, lobbyManager.OnMatchCreate);

        Debug.Log("Se ha creado la partida de nombre: " + isleCoords.text);
    }

    else
    {
        StartCoroutine(ShowErrorCoordsMessage());
    }
}
```

Figure 34: CreateTravel.OnCreateTravel( )

This script also handles the button for join to travels, which executes the OnJoinTravel( ) method. This method starts the matchmaking, then activates the *JoinPanel*, which is saved as *joinRoomObject* here and immediately refreshes the list of created games so that they appear on the screen. Figure 35 shows the OnJoinTravel( ) method.

```
public void OnJoinTravel()
{
    lobbyManager.StartMatchMaker();
    joinRoomObject.SetActive(true);
    joinRoom = joinRoomObject.GetComponent<JoinRoom>();
    joinRoom.RefreshTravelList();
}
```

Figure 35: CreateTravel.OnJoinTravel( )

3.4.2. CONNECTING TO THE LOBBY

Whoever creates the game by running OnCreateTravel( ) activates the Callback LobbyManager.OnMatchCreated( ). This causes that within the *LobbyManager* script the *ReadyPanel* is activated, and in *NetworkLobbyManager*, from which inherits the previous one, the game is started by instantiating the host's *lobbyPlayer*.

If a player joins a game by clicking on the button in the *JoinPanel* screen, the Join( ) method of the *HostSetup* script is executed. This method selects a LobbyManager if it does not have it, activates the *ReadyPanel* and adds the client to the game. The latter causes *LobbyManager* to instantiate the *LobbyPlayer* of this client. Figure 36 shows Join( ) method.

```
public void join()
{
    if(lobbyManager == null)
            lobbyManager = GameObject.FindGameObjectWithTag("LMANAGER").
                           GetComponent<LobbyManager>();

    var go = readyParent.GetComponentsInChildren<Transform>(true);

    foreach(var item in go)
            item.gameObject.SetActive(true);

    lobbyManager.matchMaker.JoinMatch(match.networkId, "", "", "", 0, 0,
    lobbyManager.OnMatchJoined);
}
```

Figure 36: HostSetup.Join( )

3.4.2.1. SYNCHRONIZING PLAYERS INFO BETWEEN DEVICES

To synchronize player values on all devices, such as the name, the following reasoning is used. The variable to be synchronized must be [Syncvar]. Then in OnStartLocalPlayer( ) the values are assigned to these variables and CmdSetPlayerInfo( ) is called. This method is a [Command] that simply assigns the values to the variables but this time it is executed in the Server, so that it is shared with all the clients. In this way, clients can show their names and states, as well as the rest that appear in the lobby. Figure 37 shows the explained above.

```
[SyncVar]
string deviceName;

public override void OnStartLocalPlayer()
{
    base.OnStartLocalPlayer();

    ready = false;

    #if UNITY_ANDROID
        deviceName = SystemInfo.deviceModel;
    #else
                    deviceName = SystemInfo.deviceName;
    #endif

    CmdSetPlayerInfo(deviceName);
}

[Command]
public void CmdSetPlayerInfo(string name)
{
    deviceName = name;
}
```

Figure 37: Getting Player's name

3.4.2.2. START ON READY

LobbyManager inherits from *NetworkLobbyManager*, and *LobbyPlayer* inherits from *NetworkLobbyPlayer*. In the properties of LobbyManager we can choose the maximum and minimum number of connections to start a game. In this case it is 2 - 4.
Each player has a "Ready" button and a "Not ready" button that activate each other. Pressing one launches a message of whether or not it is ready SendReadyToBeginMessage( ). This is a message that the client sends to the server telling him that he is ready. These messages are already implemented in the *NetworkLobbyManager*, client just have to invoke them and the server receives that client is ready.
To change the text a [Syncvar (hook = "OnReadyChanged")] was made, so that every time the value of ready changes, that method will be executed. In OnReadyChanged( ) the value in the local client is changed, calling the method that changes the text OnClientReady( ) and then, if ready is true, CheckReadyToBegin( ) is called, method of the *LobbyManager* script, which is responsible for changing to the game scene if everyone is ready. This is also base implemented.

3.4.3. SETTING UP A RANDOM CAPTAIN

When moving to the game screen, *LobbyManager* creates instances of the *Player*. After this, *GameController* creates an array with all the players and asks [Server] to assign a captain.
It is important to note that each client executes the same code, so each client has its own player array, independent of each other although in the same order.
Server takes a random index of the array, which passes to the clients through a [clientRPC], with this each client assigns the player that occupies that index in its particular array as a captain. Then, each client runs ShowGamePanel( ) which will show the captain's panel or the ship's panel depending on whether capitan bool is true or false. The host and server are synchronized, and the server must have both panels active in order to access buttons and commands. Figure 38 shows SetCaptain( ) method in *GameController* script. Figure 39 shows the referenced method in *Player* script.

```
[Server]
void SetCaptain()
    int numOfPlayers = players.Length;

    captainIndex = UnityEngine.Random.Range(0, numOfPlayers);

    Rpc_SetCapIndex(captainIndex, numOfPlayers);
}

[ClientRpc]
void Rpc_SetCapIndex(int index, int numPlayers)
{
    players[index].GetComponent<Player>().SetCaptainTrue();
}
```

Figure 38: GameController.SetCaptain( )

```
public void SetCaptainTrue()
{
    captain = true;
    orders = new List<GameObject>();
}
```

Figure 39: Player.SetCaptainTrue( )

3.4.4. THE CAPTAIN'S ORDERS

When a Player is assigned captain, the Orders List get initialized. Then after activating the captain's panel, all the orders are instantiated and put in the list. The orders will have different priorities when instantiated, this means that they will vary among them and that the priority of the same order will also vary from one game to another. To instantiate each order the ID is used, and when it is created, a switch(ID) will assign the appropriate values. The priority is the amount of the slider that is full and the relatedID is the order with which it cannot appear on the screen because they are incompatible. Figure 40 shows part of the Order Start( ) method.

```
private void Start()
    {
        active = gameObject.activeSelf;
        relatedID = -1;
        priority = SetPriority();
        percent = (slider.value * priority) / 100

        #region Switch ID
        switch (ID)
        {
            case 0:
                orderText.text = "¡Todo a estribor!";
                slider.value -= percent;
                relatedID = 1;
                break;

            case 1:
                orderText.text = "¡Todo a bavor!";
                slider.value -= percent;
                relatedID = 0;
                break;

            case 2:
                orderText.text = "¡Levad Anclas, merluzos!";
                slider.value -= percent;
                break;

            case 3:
                orderText.text = "¡Mi jarra está vacía!";
                slider.value -= percent;
                break;

            case 4:
                orderText.text = "¡Vigia! ¿Que ven tus ojos de besugo?";
                slider.value -= percent;
                break;

            case 5:
                orderText.text = "¡Soltad amarras!";
                slider.value -= percent;
                break;

            case 6:
                orderText.text = "¡Arriad la mayor!";
                slider.value -= percent;
                relatedID = 7;
                break;

            case 7:
                orderText.text = "¡Arr! ¡Izad la mayor!";
                slider.value -= percent;
                relatedID = 6;
                break;
```

```
            case 8:
                orderText.text = "¡Buscad el sotavento!";
                slider.value -= percent;
                relatedID = 9;
                break;

            case 9:
                orderText.text = "¡Las velas a barlovento!";
                slider.value -= percent;
                relatedID = 8;
                break;

            case 10:
                orderText.text = "¡¿Y las galletitas del Capitan Flint?!";
                slider.value -= percent;
                break;

            case 11:
                orderText.text = "¡Achicad mas agua, sabandijas!";
                slider.value -= percent;
                break;

            case 12:
                orderText.text = "¡Cargad los cañones!";
                slider.value -= percent;
                relatedID = 13;
                break;

            case 13:
                orderText.text = "¡Fuegoo!";
                slider.value -= percent;
                relatedID = 12;
                break;

            case 14:
                orderText.text = "¡Tenemos un hoyo! !Tapiadlo!";
                slider.value -= percent;
                break;

            case 15:
                orderText.text = "¡Traedme el mapa!";
                slider.value -= percent;
                break;

            default:
                Debug.Log("ERROR: No se ha encontrado el ID de la orden");
                break;

        }
        #endregion

        gameObject.SetActive(false);
    }
```

Figure 40: Order.Start( )

After filling the list with the maximum orders indicated in the editor, the extended method Shuffle( ) is used to disorder the list according to the Fisher-Yates Shuffle method. Figure 41 shows SetOrderList( ) method.

```
public void SetOrderList()
    {
        for (int i = 0; i < totalOrders; i++)
        {
            GameObject orderClone = Instantiate(orderPrefab) as GameObject;
            orderClone.GetComponent<Order>().ID = i;
            orderClone.transform.SetParent(GameObject.Find("CaptainPanel").transform);

            orders.Add(orderClone);
        }

        orders.Shuffle();

        for (int i = 0; i < totalOrders; i++)
        {
            Cmd_AddOrderToServerList(orders[i].GetComponent<Order>().ID);
        }
    }
```

Figure 41: Player.SetOrderList( )

Now the client who is captain, has his local list of orders created. If the captain is the host, this list is synchronized with the server. Anyway, to synchronize the server and the client there will be used two other lists as a tuple.
The idea is to have a tuple (int, bool) that tells us the order ID and if it is active or not. Unity uses NET.Framework 3.5 and only supports 4.6 experimentally. To use a tuple natively you need C#7 or C#6 with a package installed, which translates to NET.Framework 4.6+ After trying to install this, it was decided to use two lists as a tuple and not waste any more time. After Shuffle( ) the order list in the client, a Command will tell the server for each order to add that ID to its list of IDs and to put the element with that index in its list of bools to false (for the moment). This leaves the three lists "synchronized" regarding what position refers. Then, each time the client activates or deactivates an order, it tells the server to change the value with that index in the list of bools.

To show an order on screen, client counts how many orders in the list are activated, if they are less than the orders that fit on the screen, a random index is generated, if the order of that index is not already activated, see if it is the Order "Load Cannons" if so and the variable "Cannons Charge" is set to false, see if the order "Fire!" is activated, if this is so, Load Cannons cannot be activate it. If, on the other hand, the order for shooting is not activated, it activates the load cannons order and changes the value of Cannons Charge to true. If instead the command is "Fire!", see if the cannons are loaded, if so, see if the command "Load cannons" is active, if not, activate the order and change Cannons Charge to False. If, on the other hand, it is any other order, see if that order has any other realted order (with which it cannot appear on the screen at the same time), if not, activate it. If you have a related order, see if that order is activated and if it is not, activate the first order.

3.4.5. THE TIMER AND THE ORDERS SLIDER

Host starts the timer coroutine, and every time it reduces the value by one point, it tells the server to update this value with a Command. That Cmd has an Rpc that tells the rest of the clients the new value they have to show.
This is the synchronization of the timer. Now, for the sliders, each time the value is decreased,
players [captainIndex]. GetComponent <Player>( ). Cmd_DecreaseOrderSlider( ); is called. This Command simply executes a ClientRpc to run on all clients, where "if (captain)" performs the method, and if not, it does nothing.

3.4.6. THE SHIP PARTS

Server is responsible for distributing the parts of the ship among the sailors. The parts of the ship are 12 and the sailors are always the number of players less 1. With this data the parts can be divided equally to each sailor.

Each client has a list with all the parts of the ship, although deactivated. Each sailor player will iterate through the players array in order to detect which sailor is, sailor#1, sailor#2… with this information the player creates a sublist of the ship part list and activates the parts of that sublist. For example, if is sailor#1 the sublist will contain ship parts from #0 to #5 sailor#2 the sublist will contains ship parts from #6 to #11. In this way each sailor has the correct number of parts and all are different, always being all parts present in the game.

3.4.7. THE ISLE

Server sends all clients to the island at the end of time, but once there, each client can move independently through the landscapes, there is no need to go in group. The moment the treasure button is pressed by a client, the [Syncvar] TreasureFound of that client will become true and Cmd_TreasureFound( ) will be called. This method changes the value of the variable in the server, so that it is changed in the rest of the clients.
Being this variable true, the *WinPanel* is activated.

4. RESULTS

The Android app developed in this project can be downloaded from the following link or by scanning the QR code in Figure 42:
https://mega.nz/#!NUUAnbiK!eeJ04XKDJkSdXdHuimJ87FtYRdGxJmDlw0Yya1oq8oq

The map of the implemented isle and the nautical chart can be downloaded from this other link or the QR code in Figure 43:
https://imgur.com/a/cAq9XvX

Figure 42: Pocket Crew QR

Figure 43: Map and Nautical chart QR

The complete unity project can be downloaded from the following link:
https://mega.nz/#!1IMF3YKZ!VClO0UnhpV6rNR2j3FPkL4L47N0a8pi_0QauiOwH6II

It has been developed a game which achieves all the expected results mentioned in the proposal:

- To get a videogame with a board game format, in which players must be physically gathered and share objects directly with others.
- To implement the connection to each one's devices as in a very easy way and technical knowledge doesn't be a necessity.
- To create a gameplay easily extendible with new maps in a future.
- To create an enjoyable experience.

The use of physical objects as they are the map and the nautical chart makes from this game a social experience in which players must gather around this objects, share them and interact with them. Making this way hard to differentiate this game from a board game.
The connection does not need any technical knowledge from the users, they only need internet connection, no matter if they are all connected to the same router or not. The connection system is matchmaking, in which one player creates a game and others join it. At the game the game is created by introducing the isle coordinates and this player will be established as host in the network.

This approach allows the easily implementation of new isles. Each isle coordinates can be related to a unique set of landscapes, this way when the coordinates are introduced by a player no only the game is created, also the corresponding landscapes are selected to be activated later during the game.
During a gameplay test of the project where four people played the game it could be appreciated the real enjoyment and reactions of the players. Connection was easy and quick every time, and the ship phase provides the experience with the most part of the fun. This phase only takes one minute and there are needed 15 accomplished orders to reach the isle, that makes players nervous by not to have time but also excited by seeing the accomplished orders to grow up. Sometimes it takes two or three tries to reach the isle but in each try, the ship phase is full of laughs, nervous screams and fun.
The gameplay of the game, where one player reads orders, the others press the buttons and later all of them solve a riddle, makes it capable to all kind of people, no matter age or video game experience. These factor makes this game the perfect base on which verify social theories and explore the social potential of video games.

5. PROJECT DEVIATIONS

The following table shows the contraposition of the estimated and the real hours which each task has needed to be accomplished. Final hours in the table are round, see appendix 3 for the detailed workflow (Appendix 3).

Table 15: Planning comparison

Task	**Estimated** (in hours)	**Final** (in hours)
Technical proposal	6	9
Game Design Document	10	15
Report	20	45
Final Report	4	18
Final Presentation	10	10
Analogical Prototype	10	10
Project Setup	-	19
Art, Riddles, Interface	50	17
Music	20	2
Meeting with tutor	-	9
Codding	140	151
Total hours	300	305

5.1. DECISION-MAKING

Here are the different decisions that have been taken throughout the development and its reasons:

- **Why not player have orders and ship parts at the same time in screen?**
  If all the players are talking at the same time they will not be able to listen to the orders that are said

- **Printed nautical chart?**
  To encourage social interaction have been thought about printing the nautical chart together with the maps, so that all the players must search together on a single physical object.

- **Do players need to be in group at the isle?**
  If each player can be in different landscapes, there is the possibility of introducing asymmetric design mechanics for finding the treasure "Pull this rope while raising that rock to open the way".

- **Can players tap on the isle landscapes at their choice?**
  To prevent them from finding the treasure by touching the entire screen without reasoning, a mechanic will be implemented so that if the screen is touched more than once in less than 10s, the screen will stop responding for 30s for all players.

- **Captain just need to read?**
  To keep the captain from getting bored, the orders will have priorities. Each will appear on the screen for a different time, so the captain should organize the reading order.

- **Bluetooth or wi-fi?**
  Due to the accessibility and possible problems that Bluetooth may cause, Unity Multiplayer (UNET) will be used, which allows you to easily create multiplayer games. In our case, by wi-fi, acting a device as host.

- **Why are the graphics in Spanish?**
  This makes it easier to test with all kinds of people, young and old.

- **Local multiplayer?**
  As was explained in the proposal (1.7. Related Literature), multiplayer does not ensure a real social experience. For it has preferred a point of view closer to the games of table, where the people must meet in a place and interact with others directly.

6. CONCLUSION

With this project it has been intended to create a social experience by using asymmetric design. To promote the social effect, the format of the project has been inspired by board games, for this reason, physical interaction has occupied an immovable place in the gameplay design.

The final result has reached the objectives stated at the proposal (1.3. Objectives).
Asymmetric design has been implemented satisfactory at the ship part of the game, where each player must execute a different action in order to reach the minimum required and arrive at the isle. That is directly related with the random shipment of ship parts to each device, which make each player to have completely different elements in screen, encouraging this way the asymmetric design previously mentioned.
Objectives related with creating *"fun game dynamic which encourage collaboration"*, developing *"a game which will be difficult to differentiate from a board game"* and researching *"the social potential of video games"* are depicted at results, chapter four of this document.
User friendly connection among devices has been managed by using unity multiplayer service [20]. It is only needed to appoint the person in charge of creating the game, once the game has been created it is only required the rest of players to join it.

Despite of all marked objectives have been reached, there are some traits of the project which have not been accomplished due to planning deviations which was mentioned at chapter five.
It was planned to generate original art for the project and add music and sound effects to scenes. Also some technical implementations have remained unfinished, as adding penalties to the isle scenes by freezing the devices when a player try to find the treasure by touching random screen parts. Also making the HUD on the ship scene (time and score) visible to players only when the *"Informe del vigía"* button were pressed.

In spite of the unaccomplished traits it has been developed a functional project which allows to verify the theories about the social potential of video games and sets up the base on which to add future implementations.

6.1. FUTURE

There are several points in which this project can expands in the future, such as performing user tests to corroborate the studies and theories on which it was based, which are mentioned in the proposal (1.7. Related Literature).

For this it is planned to make different groups of people play the game. Without disintegration of sex or office, but taking notes on various topics such as:

- Experience with videogames.
- Experience with board games.
- Sex.
- Have brothers/sisters.
- Have children.

Some groups will be formed by people with near ages, other groups will be formed by people of different ages. After playing the game they will be asked to participate in a group activity and their results will be compared to similar groups that have not played the game.

This is expected to show that the project developed in this document favors relationships and is fun for both kind of people, those who are accustomed to video games and those who do not.

6.1.1. FUTURE IMPLEMENTATIONS

It has been developed a project with a great expansion potential. Some of the planned implementations to be added in a future are:

- Implementing a system for players to add their own isles and maps.
- Adding meteorological effects during the ship phase.
- Adding more possible isles.
- Asymmetric design mechanics for treasure finding. An example of this could be that one player must touch a rock while another one touch a tree in order to activate the path to the treasure.

7. REFERENCES

[1] Guilleminot, R. (July 18, 2012). Asymmetric design arrives. Cooper journal. https://www.cooper.com/journal/2012/07/asymmetric-design-arrives

[2] Cagnon, D. (2009). Symmetrical vs. Asymmetrical Balance in Game design. https://davidgagnon.wordpress.com/2009/08/16/symmetrical-vs-asymmetrical-balance-in-game-design/

[3] Sánchez, P. A., Alfageme, M., & Serrano, F. J. (2010). Aspectos sociales de los videojuegos. *Revista Latinoamericana de Tecnología Educativa*, *9*(1), 43–52.

[4] Juego social - Wikipedia https://es.wikipedia.org/wiki/Juego_social

[5] Ansgar E. Depping, Regan L. Mandryk, Colby Johanson, Jason T. Bowey, Shelby C. Thomson. (October, 2016). Trust Me: Social Games are Better than Social Icebreakers at Building Trust. Conference: the 2016 Annual Symposium. DOI 10.1145/2967934.2968097

[6] David W Johnson, Roger T. Johnson. (January, 1989). Cooperation and Competition: Theory and Research.

[7] Maaz Nasir, Kelly Lyons, Rock Leung, Ali Moradian. (June, 2003). Cooperative Games and Their Effect on Group Collaboration. Conference: Proceedings of the 8th international conference on Design Science at the Intersection of Physical and Virtual Design. DOI 10.1007/978-3-642-38827-9_43.

[8] Ansgar E. Depping, Regan L. Mandryk. (October, 2017). Cooperation and Interdependence: How Multiplayer Games Increase Social Closeness. Conference: the Annual Symposium. DOI 10.1145/3116595.3116639.

[9] John Harris, Mark Hancock, Stacey D Scott. (October, 2014). "Beam me 'round, scotty!": Exploring the effect of interdependence in asymmetric cooperative games. DOI 10.1145/2658537.2661311.

[10] John Harris, Mark Hancock, Stacey D Scott. (October, 2015). "Beam Me 'Round, Scotty!". Conference: the 2015 Annual Symposium. DOI 10.1145/2793107.2810274.

[11] Gregor Mcewan, Carl Gutwin, Regan L. Mandryk, Lennart Nacke. (February, 2012). "I'm just here to play games": Social dynamics and sociality in an online game site. Conference: Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work. DOI 10.1145/2145204.2145289.

[12] Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, Robert J. Moore. (January, 2006). Alone Together? Exploring the Social Dynamics of Massively Multiplayer Online Games. Conference: Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006. DOI 10.1145/1124772.1124834.

[13] Hieronymi, A. (2009). Physical games, beyond mini-games. *Interactions*, *16*(3), 34–41. DOI 10.1145/1516016.1516025

[14] Tejeiro Salquero, R. A. (2002). ¿Fomentan los videojuegos el aislamiento social? Eúphoros, nº 5, pp. 233-239. ISSN: 1475-0205.

[15] CaptainMess library
https://github.com/hengineer/CaptainsMess

[16] Unity Main Menu Tutorial.
https://www.youtube.com/watch?time_continue=1886&v=OWtQnZsSdEU

[17] Unity Manual: Networking HLAPI System concepts
https://docs.unity3d.com/Manual/UNetConcepts.html

[18] Unity Manual: The multiplayer High Level API
https://docs.unity3d.com/Manual/UNetUsingHLAPI.html

[19] Unity Manual: Remote Actions
https://docs.unity3d.com/Manual/UNetActions.html

[20] Unity Manual: Setting up Unity Multiplayer
https://docs.unity3d.com/Manual/UnityMultiplayerSettingUp.html

[21] Unity Manual: Network Identity
https://docs.unity3d.com/Manual/class-NetworkIdentity.html

Appendix 1 - CaptainMess Tests

These tests consist on playing two consecutive plays. The used devices was my Personal Computer and two smartphones (Wileyfox Swift 2 plus & LG D280n).

CaptainMess Test 1 Two Devices (LG - Swift)

	LG	Swift	Host
1	La imagen del pj de swift no aparece. Se ha desconectado al lanzar el segundo dado. "ServerDisconected due to error: Timeout" Problemas ID: 1	ok	LG
2	ok	ok	LG
3	En la segunda partida, se ha desconectado al lanzar el segundo dado. Problemas ID: 1	ok	LG
4	Se ha desconectado al lanzar el segundo dado Problemas ID: 1	ok	LG
5	La imagen del pj de swift no aparece. Tras darle a play again, ha salido el botón de "roll die" y se ha desconectado Problemas ID: 1	ok	LG
6	La imagen del pj de swift no aparece. Se ha desconectado al lanzar el segundo dado durante la segunda partida Problemas ID: 1	ok	LG
7	Al conectarse swift al lobby, LG se ha desconectado Problemas ID: 1	ok	LG

CaptainMess Test 2 Two Devices (PC - Swift)

	PC (editor)	Swift	Host
1	ok	ok	swift
2	ok	ok	swift
3	ok	ok	swift
4	ok	ok	swift
5	ok	ok	swift
6	ok	ok	swift
7	ok	ok	swift

CaptainMess Test 3 Two Devices (PC - LG)

	PC (editor)	LG	Host
1	ok	Tras la cuenta atrás se ha desconectado. Problemas ID: 1	LG
2	ok	Ha contado el tercer dado. Luego se ha desconectado Problemas ID: 1	LG
3	ok	ok	LG
4	ok	Tras contar el segundo dado de la segunda partida se ha desconectado Problemas ID: 1	LG
5	Ha salido "Conected" y luego "offline" se ha quedado sin entrar al lobby. Problemas ID: 3	¿ok?	LG
6	ok	Ha contado el tercer dado. Luego se ha desconectado Problemas ID: 1	LG
7	ok	Se ha desconectado en la cuenta atrás Problemas ID: 1	LG

CaptainMess Test 4 Three Devices (PC - Swift - LG)

	PC (editor)	Swift	LG	Host
1	ok	ok	Se ha quedado en un lobby solo, con él como host	LG / Swift
2	ok	Ha entrado al lobby pero luego se ha desconectado	ok	LG
3	ok	ok	Se ha quedado en un lobby solo, con él como host	LG / Swift
4	ok	ok	No aparece la imagen del jugador de swift. Tras lanzar el segundo dado se desconecta. Problema ID: 1	LG
5	no se ha conectado a nada	se ha conectado a algo e instantaneamente se ha desconectado	Se ha quedado en un lobby solo, con él como host	-
6	Se ha desconectado solo. Problema ID: 3	se ha conectado a LG pero no ha ido al lobby	Se ha desconectado solo. Problema ID: 1	-
7	No conecta Problema ID: 2	No conecta Problema ID: 2	No conecta Problema ID: 2	-

Appendix 2 – Versions

The following table shows all the project versions developed until de final demo state:

Versión	Descripción
v0.1	Menu and Lobby. Input recognizes the correct coordinates and shows error text if the introduced ones are not valid.
v0.2	One player creates a match, another join it and both are added to the lobby.
v0.3	Name of each player visible in all devices
v0.4	Game only start when all players are ready
v0.5	Captain is chosen randomly and each player goes to the pertinent panel (Sailor/Captain)
v0.6	Adaptable screen aspect
v0.7	Seven random orders are showed in screen for captain. Having in count that some orders can not appear at the same time than others.
v0.8	Timer has been added and orders slider decreases each second. When it comes to zero, the order deactivates and another random order pops up. If timer gets zero the screen isle or game over screen appears, depending on the accomplished orders.
v0.9	Sailor button list implemented. Also all screens until the win screen has been implemented.
v0.10	Captain creates and synchronizes his order list with the server.
v0.11	Clients have a synchronized timer which change all device screen when it comes to zero.
v0.12	Number of ship parts are correctly sent to each sailor. Also each device has different parts.
v0.13	Accomplished orders score increase and decrease correctly by pressing the sailor buttons.
v0.14	By pressing the correct button, the related order disappears from captain screens and another one appears.
v015	All devices show win screen when one player find the treasure.
v1.0	Final functional demo.
v1.1	Landscapes improved, as the font and the timer and score colors.

Appendix 3 - Real workflow detailed

Leyenda	Corrección
Memoria	Proyecto en general
Reunión	Código
Tarea analógica	Arte

DÍA	Horas por tarea	TRABAJO REALIZADO	ESTADO	OBSERVACIONES	Horas al Día	Horas Acumuladas
16/02/2018	2:00	GDD	-	-	3:30	3:30
16/02/2018	1:30	Reunion	-	Hay que corregir la propuesta y hacer prototipo en papel		
17/02/2018	10:00	Prototipo en papel	Finalizado	-	10:00	13:30
22/02/2018	1:00	Reunión	-	Pensadas posibles ideas de diseño	1:00	14:30
25/02/2018	2:30	Corregir propuesta	Finalizado	-	2:30	17:00
26/02/2018	3:00	GDD	-	-	3:00	20:00
27/02/2018	4:00	Crear Repositorio, instalar Unity, Visual Studio, Android SDK, JDK	Finalizado	Tuve que mirar un tutorial y olvidé que no tenía unity en este PC. Perdí mucho tiempo descargando e instalando, tuve que instalar unity dos veces para que reconociese android. Instalar el SDK me ha dado muchos problemas, al final he tenido que hacerlo con android studio.	5:15	25:15
27/02/2018	1:15	Conseguir exportar a android desde unity	-	Una vez instalado todo siguiendo un tutorial, Unity no exportaba a Android y tuve que instalar una versión más vieja del SDK. Luego me dio un error raro que solucioné cambiando el nombre de la carpeta del proyecto. Source tree me dio un error de logueo y tuve que volver a crear el repositorio. No se solucionó.		
28/02/2018	2:00	Exportar a Android	Finalizado	Tutorial para exportar	2:00	27:15
28/02/2018	0:00	"Hola mundo" funcional en	Finalizado	La misma app que he exportado muestra un texto en pantalla.		

		android				
01/03/2018	3:00	Comprobar que funciona en mi móviles	-	Debo comprobar si cambiando el SO los moviles que funcionan mal se arreglan. (Son moviles de propaganda y la pantalla va fatal)	3:00	30:15
12/03/2018	0:45	Comprobar si bajando la Api me deja exportar para Android anteriores	**Finalizado**	He instalado la Api de Android 2, pero no he podido instalarlo en dispositivos. Resumen: 1 no tiene memoria / 2 no encienden / 1 no funciona bien la pantalla / 3 funcionan (incluido el mío)	2:45	33:00
12/03/2018	2:00	Entender multijugador en Unity	-	Leer manual		
13/03/2018	0:40	Probar la librería de SpaceTeam	**Finalizado**	He bajado la libreria de spaceteam para probar. Pero me ha dado algunos problemas de compilación (Detalles en la pestaña Problemas). Finalmente he podido exportarlo y comprobar que funciona en mis dispositivos. Ahora tengo que saber cómo funciona exáctamente. Cuando los 3 dispositivos (PC, LG, Swift) hacen auto connect, funciona. Cuando uno hace Host y los otros auto connect, funciona. Cuando uno hace Host y el resto Join, no funciona.	1:50	34:50
13/03/2018	1:10	Crear diagrama de escenas	**-**	Aquí		
14/03/2018	1:00	Crear diagrama de clases	-	https://www.draw.io/#G1IdBVaviC3NT4Pokd2FAR-nH3trnzWR6E	1:00	35:50
18/03/2018	2:00	Diseñar clases y escenas	**Finalizado**	He aclarado todo el esquema base de clases y escenas. En la escena del juego se usarán paneles de unity	2:00	37:50
21/03/2018	3:30	Ingeniería inversa a la API de Captain Mess	-	He estudiado cómo usa cada clase, lo que me ha servido para detallar mejor mi esquema. La parte de la red aún necesito estudiarla mejor	3:30	41:20
22/03/2018	1:00	Reunión	**Finalizado**	Para el día nueve, tener un prototipo funcional con un primer arte. Usar en primer lugar la API para tener una demo lo antes posible, más tarde crear nuestra	1:00	42:20

				API o adaptarla.		
19/04/2018	1:00	Menu Principal	-		1:00	43:20
21/04/2018	2:00	Menu Principal	**Finalizado**	Menú implementado con el cambio de escenas en la aplicación. Hay que solucionar la resolución en Android	4:30	47:50
	1:00	Que se vea con la resolución correcta en Android (Sin deformar)	**Finalizado**	Se soluciona poniendo 16:10 en el aspecto de la pestaña "Game" del editor, y en el canvas scaler poner 800x1280. Esa es la resolución elegida para el juego. También se ha bloqueado el giro de pantalla		
	1:30	Implementar conectividad (CaptainMess)	-	Cuando da al start se conecta al servidor y cambia a la escena de lobby. En el lobby cuando se le da a Back, vuelve al estado inicial.		
22/04/2018	7:40	Implementar conectividad (CaptainMess)	**Finalizado**	Se usa un objeto "Game Manager" que contiene lo necesario para la conexión y está presente en todas las escenas del juego. Aparecen los jugadores y la cuenta atrás cuando todos están listos, y se cambia a la escena del barco. En varios dispositivos, solo uno va a la escena y los mensajes de SERVER aparecen solo en ese.	7:40	55:30
23/04/2018	3:00	Solucionar cambio de escena en el cliente al iniciar partida. Entender funcionamiento de la librería más a fondo.	**Finalizado**	Entender cómo funcionan el networkManager, el server y el cliente. También los comandos y las RPC. Ya está casi todo para empezar a implementar las funcionalidades propias de mi proyecto.	3:00	58:30
24/04/2018	1:45	Botón GO	-	Estoy debugueando para seguir el rastro de instrucciones del botón Ready. Mientras probaba, han aparecido fallos de conexión. Cuando el PC es host, los moviles no se conectan. Tampoco con el juego original de CaptainMess	9:15	67:45

	2:00	Organizar Scripts para PocketCrew	**Finalizado**			
	5:30	Testear conexión. Aprender a ver los Debug de un móvil. Control de versiones	**Finalizado**	Tras perder tiempo con bitbucket he decidido manejar manualmente las versiones. Más adelante crearé correctamente el repositorio. Detalles de las pruebas en la pestaña TEST		
25/04/2018	2:10	Botón Cancel	**Finalizado**	El boton Cancel funciona pero al volver al menu, el gameManager de la partida actual se mantiene, por lo que a partir de aqui existen dos GameManagers. Debo eliminar el GameManager tras cancelar la conexión. Para evitar esto voy a meter el lobby y el menu en la misma escena, en paneles separados, así al volver al menú no se duplicará el gameManager.	6:20	74:05
	0:20	Preparar imagenes para demo	**Finalizado**			
	0:20	Distribución de los elementos	-	La imagen de fondo del lobby tapa al personaje y al estado de la conexión. (Canvas del gameManager). Voy a seguir avanzando con cosas más importantes.		
	3:30	Añadir selector de isla en el Lobby	**Finalizado**	El script de InstancedPlayer tiene una variable "isleCoord" que se sincroniza con el input field de selección de isla. El input field tiene adjunto el prefab del player. He tardado en conseguir sincronizar el input con la variable del script		
28/04/2018	5:40	Se debe comprobar que todos los players tengan las mismas coordenadas de isla antes de empezar la partida	-	La idea es que el host sea quien mete las coordenadas pero no consigo que le salga el input sólo a él y tampoco que la variable se actualice para saber si empezar o no la partida. Ahoramismo si que reconoce si la coordenada es valida o no para empezar a jugar, pero el inputfield no actualiza la variable.	10:10	84:15

	3:20	Crear proyecto multijugador basico para asentar coceptos	**Finalizado**	https://unity3d.com/es/learn/tutorials/s/multiplayer-networking		
	1:10	Crear menú con conexión de jugadores desde cero	-	La librería complica más que ayuda. Voy a comprobar si es viable hacer de cero todo.		
29/04/2018	5:40	Ver tutorial lobbyManager	**Finalizado**	He empezado de cero, usando el lobbyManager en lugar del NetworkManager. He perdido horas intentando entender por qué no me creaba la partida el LobbyManager, resulta que había que configurrar los ajustes multijugador del proyecto en la web de unity. Más detalles en la pestaña problemas. Tras el tutorial está en la versión v01	13:50	98:05
	0:05	Que el cliente tenga nombre del dispositivo	**Finalizado**			
	7:25	Ahoramismo siempre se es host. Solucionarlo.	**Finalizado**	Ya se conectan hosts y clientes al juego. Sólo falta ajustar que los elementos no se muevan entre dispositivos, se vean los nombres de todos y organizar un poco los elementos que pasan de una escena a otra. Version v02		
	0:40	Buscar música				
30/04/2018	1:30	Buscar música			1:30	99:35
01/05/2018	2:20	Que los nombres salgan en todos los dispositivos	**Finalizado**	Al hacer esto, se veian los botones ready de todos los jugadores. Para solucionarlo lo he desactivado en el prefab del player y en SetUp solo lo activo si isLocalPlayer es true. Version v03	4:45	104:20
	0:15	Mejorar la pantalla de seleccion de viaje	**Finalizado**	Actualmente sale la imagen del jugador, quiero dejar solo un botón y que el jugador salga ya en el lobby		
	2:10	Empezar la partida sólo cuando	**Finalizado**	Version v04		

		todos estén listos				
02/05/2018	0:40	Al empezar la partica, los paneles de Menu, lobby y Join se eliminan	**Finalizado**	El panel de join no se activa cuando el cliente es host, por lo que luego no se borra. No es que sea importante, pero debería solucionarlo en un futuro.	3:20	107:40
	1:20	Diferenciar entre capitán y tripulantes	**Finalizado**	Selecciona un capitán al azar, pero he encontrado un problema y es que duplica los jugadores.		
	1:20	Solucionar jugadores duplicados	-			
03/05/2018	3:30	Solucionar jugadores duplicados	**Finalizado**	El lobbyNetworkManager tiene un prefab de lobbyPlayer y otro de Player para el juego. Tenía puesto lobbyPlayer en los dos, he creado un nuevo prefab para el player dentro del juego. Version v05	6:40	114:20
	3:10	Que se mantenga la escala en distintos dispositivos	**Finalizado**	Me ha dado muchos problemas. El truco está en cambiar la posición de la cámara para cada dispositivo, pero los canvas deben estar en modo "Screen space - overlay" y luego el canvas scaler en "Scale with screen size" con el método "shrink". He perdido horas intentando que los botones de seleccionar partida de la pantalla join se escalen, no tiene sentido. Lo he dejado con un tamaño fijo que cabe en todos los dispositivos que uso, pero debería arreglarlo más tarde. Version v06		
04/05/2018	1:00	Asegurarse de que elige bien al capitan	-	He perdido tiempo probando la conexión, a veces no se conectaba, tiene pinta de ser por problemas de internet.	1:45	116:05
	0:45	Reunión	**Finalizado**			
05/05/2018	2:15	Asegurarse de que elige bien al capitan	**Finalizado**	Está claro que el capitán aleatorio funciona, el fallo está en que a veces el cliente no activa el panel correspondiente	2:15	118:20
06/05/2018	2:30	Activar los paneles	**Finalizado**	puse Random.range(0, numPlayers - 1) cuando debe ser	3:30	121:50

		correctamente si es capi o no		Random.range(0, numPlayers). Error estúpido. He perdido dos dias aquí.		
	1:00	Que el capitán tenga ordenes y el marinero las cumpla	-			
08/05/2018	1:05	Buscar Papers	**Finalizado**		5:35	127:25
	4:30	Ordenes	-	Esta dando un error de nullreference, creo que no puedo cambiar el texto si no hay un objeto ya instanciado. Posible solución sería crearme un prefab para cada orden e ir instanciando y destruyendo según convenga		
09/05/2018	4:00	Memoria	-	He revisado la memorioa. Reajustado el índice y la propuesta técnica tomando otras memorias como referencia.	11:30	138:55
	1:00	Leer papers	-			
	6:30	Ordenes	**Finalizado**	Se muestran 7 ordenes aleatorias en pantalla, teniendo en cuenta que algunas no pueden aparecer junto a otras. También tiene en cuenta que las órdenes de cargar cañones y disparar se deben ir alternando. Cada orden comienza con una prioridad aleatoria, esto es que a mayor prioridad, menos tiempo tiene el jugador para ejecutarla. Aún no tienen interacción ni se reduce el contador de tiempo. Version v07		
10/05/2018	2:20	Que los sliders de las órdenes se reduzcan con el tiempo	**Finalizado**	Se reducen y van cambiando de color. Cuando llegan a 0 desaparecen, y se activa otra con el slider reseteado tanto en valor como color. Se ha implementado también el timer general de esta fase, cuadno llega a 0 y no tenemos las ordenes minimas completadas, nos manda a la pantalla de game over. Si tenemos más de las minimas, nos manda a la isla. Version v08	18:20	157:15
	1:15	Botones de	**Finalizado**	He creado la lista de botones que		

		los marineros		se instancia en SailorPanel		
	2:30	Que al pulsar el botón se complete la orden	-	No consigo que funcione la comunicación. Creo que un fallo es que todos deben tener la lista de ordenes y botones.		
	6:30	Que las ordenes y los botones los cree el server y de ahi pasen a los clientes	-	Estoy intentcando sincronizar con SyncList, para eso he tenido que hacer un struct y meterlo dentro de la orden, el unico fallo que tengo es que la SyncList no esta inicializada. Las ordenes empiezan a aparecer y desaparecer a lo loco		
	3:10	Probar planteamiento simplificado	-	Lo he intentado con commands y Rpc pero sigo necesitando la lista de ordenes en el cliente. Hay que sincronizar la lista o encontrar un modo alternativo.		
	1:05	Arreglar escalado de pantalla	**Finalizado**	Hay que poner el canvas a "scaled with screen size" con 720x1280 de resolucion y "Match width or heigh" con un 0.5 Luego los que tienen vertical layout hay que retocar un poco		
	1:30	Implementar botones secundarios	**Finalizado**	Todas las pantallas y botones del juego, excepto por el botón de reintentar y el del tesoro, que no está multijugador. Si sale y entra de la pantallade join, se duplica el viaje Version v09		
11/05/2018	1:00	Corregir asignación del capitán	**Finalizado**	No lo estaba haciendo bien, ahora se elige un índice en el server y se pasa ese indice a los clientes por RPC para que todos asignen a ese jugador del array como capitán.	7:00	164:15
	6:00	Crear la lista de ordenes correctamente para el server y los clientes	-	Voy a intentar crear la lista en un Commad, ya que esto lo ejecuta el cliente y el server. Y pasarla al resto de clientes con RPCs. Cuando el capi es el host, este pasa crea la lista y la pasa correctamente al server con un Command. Cuando el capi es un cliente, dice que no tiene autoridad para ejecutar el command.		

12/05/2018	8:05	Que se pueda llamar un command desde el cliente	**Finalizado**	SE PEDE LLAMAR A COMMANDS DESDE LA CLASE DEL PLAYER.	13:05	177:20
	5:00	Gestionar las listas en la clase del Player	**Finalizado**	Solucionado. La lista ya se comparte entre Capi y Server. Los commands hay que llamarlos desde el player.		
13/05/2018	6:30	Crear y sincronizar (isActive) las ordenes entre cliente y servidor	**Finalizado**	UNITY NO DEJA USAR TUPLAS. Uso en el server una lista con IDs y otra con bools para el estado. La lista de ordenes del player, y estas dos del server están en el mismo orden de índices.	6:30	183:50
14/05/2018	1:00	Crear un mismo Timer para todos los clientes	**Finalizado**	El host lo inicia y cada vez que resta un segundo, manda un Cmd al server y este un Rpc al resto de clientes, con el nuevo valor	8:30	192:20
	0:25	Que los sliders se reduzcan y las misiones vayan desapareciendo	**Finalizado**			
	0:05	Que todos cambien de pantalla cuando se acabe el tiempo	**Finalizado**	commands y rpc		
	6:30	Que se creen los botones	**Finalizado**	Cada jugador crea el numero correcto de botones. Pero se repiten a veces entre clientes.		
	0:30	Que los clientes creen botones distintos	**-**			
15/05/2018	7:10	Hacer el "estado del arte"	**Finalziado**	Sacar lo básico de cada artículo y organizar las ideas	7:10	199:30
16/05/2018	1:00	índice final	**Finalizado**		14:00	213:30
	1:30	carta náutica	**Finalizado**			
	1:30	Preparar imagenes	**Finalizado**			

		para memoria				
	10:00	Completar GDD	-			
17/05/2018	1:10	Reunión	Finalizado	Resueltas dudas de la memoria	4:10	217:40
	3:00	Memoria	-			
18/05/2018	8:00	Memoria	-	diagramas, photoshop...	8:00	225:40
19/05/2018	16:30	Memoria	Finalizado		16:30	242:10
20/05/2018	2:00	repartir partes distintas a los marineros	-		2:00	244:10
23/05/2018	1:00	repartir partes distintas a los marineros			1:00	245:10
24/05/2018	3:00	repartir partes distintas a los marineros			3:00	248:10
26/05/2018	6:10	repartir partes distintas a los marineros	Finalizado	v012	12:35	260:45
	0:15	Aparece el marcador en todos los dispositivos	Finalizado			
	3:00	Los botones modifican el marcador	-	El contador aumenta y disminuye pero los botones no parecen corresponderse con las ordenes		
	0:20	Correspondencia botón-orden	Finalizado	Hay que buscar la orden en la lista de IDs de ahí se saca el índice para mirar la lista de bools. v013		
	0:20	Las ordenes desaparecen al dar al botón	Finalizado	Tras aumentar el contador hace un RPC para que el capitan desactiva esa orden de la lista y llame a ShowOrder. v014		
	0:30	Todos van al tesoro tras ganar	Finalizado	Se ejecuta un Cmd que llama un Rpc activadno el panel de win en cada jugador. v015		
	2:00	Retoques y	Finaliazdo	v1.0		

		testeo				
27/05/2018	4:20	Retocar paisajes isla	**Finalizado**		6:30	267:15
	1:30	Meter paisajes en el juego y cambiar la fuente	**Finalizado**	v1.1		
	0:40	Actualizar imágenes memoria	**Finalizado**			
28/05/2018	1:00	Reunión con miembro del tribunal	**Finalizado**		1:00	268:15
29/05/2018	11:00	Corregir memoria	**Finalizado**		12:00	280:15
	1:00	Reunión con miembro del tribunal	**Finalizado**			
30/05/2018	0:15	Esquema presentación	**Finalizado**		1:15	281:30:00
	1:00	Reunión con tutor	**Finalizado**			
02/05/2018	8:30	Nuevas imágenes	**Finalizado**	He hecho esquemas usando Arte de internet para tener una idea de cómo quedarían las pantallas. La idea es tomar luego estas composiciones de referencia y generar un arte propio.	12:50	294:20
	4:20	Hacer imágenes propias	**Finalizado**	He hecho una imagen low poly para probar el estilo. No me gusta.		
03/05/2018	6:00	Corregir memoria	**Finaliado**		6:00	300:20