Table of Contents

Section 1: Introduction

This section serves as a small introduction to our project, a videogame called "Dragon Boat Racing" as specified by our client Mr. Javier Cámara. Dragon Boat Racing is a 2D, single-player game developed for the PC platform using the Java programming language.

For the gameplay aspects, Dragon Boat Racing is an arcade racing game where the player will experience the rush of competition, the excitement of power-ups, the satisfaction of overcoming obstacles and most importantly the glory of winning a race. It consists of various levels, each one with increased difficulty and unique scenery. Players will have the ability to customize their boat´s appearance and stats to match their playstyle.

Every race is an adrenaline-fueled test of skill and strategy as it takes great skill for a player to control their boat in the lane and defeat their opponents. The game's mechanics are basic yet extremely exciting. Leaving the lane penalizes you making it harder for you to win the race and colliding with obstacles eventually results in the sinking of your boat. It is here that the game will give less skilled players a chance to play a "Simon-says" mini-game to repair and revive their boat. The different power-ups make the experience more exciting, enjoyable, and dynamic.

Section 2: Roles

To achieve our goal of creating an excellent, enjoyable, and exciting game, it is important that we distribute the work among our team. This is also essential for satisfying the needs of our client, Mr. Cámara, and delivering a great final project.

Keeping in mind the project's needs we have decided to have five roles. They are as following:

- **Programmers** will be responsible for writing all the code and carrying out all the tasks related to development.
- **Project Managers** are there to prepare our Trello, keeping an eye out on the project to make sure that we are advancing and most importantly they are the coordinators of the project, this means that they will make sure that we meet all our deadlines.
- **Spokespersons** will be responsible for coordinating and establishing a healthy relationship with our client. Their main job is to act as a communication channel between our team and Mr. Cámara.
- **Testers'** job is essential for any programming project. They will make sure that the code functions as intended and will give their feedback to the programmers making sure that the final product has no unexpected bugs.
- **Graphic Designers** will design all the graphical elements for our project. Be it mockups, different elements of the game like boats, levels and loading screens. Their job is essential to our project as it will ensure that the final project has stunning visuals.

Next up you will find a table listing all our team members with their assigned roles. Each team member has been assigned two roles to ensure each role has sufficient assignees.

Team Member	Role 1	Role 2
Allitt López Ricardo Juan	Spokesperson	Tester
Barrios Moreno Manuel	Project Manager	Graphic Designer
Bayon Pazos Ángel	Tester	Graphic Designer
Escaño López Ángel Nicolás	Spokesperson	Programmer
Hormigo Jiménez Pablo	Project Manager	Programmer
Jorda Garay Francisco Javier	Spokesperson	Programmer
Sicre Cortizo Diego	Project Manager	Tester
Sultan Sultan Muhammad Abdullah	Programmer	Graphic Designer
Torres Gómez Juan	Tester	Graphic Designer

Section 3: Risk Management

In this section we introduce the main risks we are facing with regards to the project. We have created a table in which we specify the risk, it´s type, a little description in which we present it and we have added a metric of the likelihood of it happening and the severity it presents, all of them are followed by the consequent contingency plan we would apply in case the risk materializes.

It is fundamental for the project to succeed to have an appropriate anticipation of the possible things that could go wrong and the way in which we plan to fix them. This will allow us to anticipate the problems before they occur and will allow us to save valuable resources.

Risk	Type	Risk Description	Severity / Probability	Contingency Plan
Underestimating complexity of a task	Organization	If a given task is underestimated, it may lead to delays in the project and a need for reorganization of tasks.	Medium-High Medium-Low	Redistribute the work force to rapidly get back on track and minimize the effects.
Lack of risk management	Organization and requirements	Failure to identify and proactively manage risks can lead to surprises during project development, resulting in delays, additional costs, and loss of client confidence	Very High Low	Develop a good study plan that encompasses all the risks that may arise during work, from the simplest and least probable to the most complex, to avoid surprises
Finished task is not what was expected	Project and Personal	Failure in communication between the project team and the client leading to an incorrect product.	Medium Low	Replan task with team making sure everyone understands the objectives needed and recommunicating with the client
Other assignments demand our time above the project.	Personal and Organization	A bad planning of personal time results in overlapping of other projects and deadlines that require our attention before this project.	Medium Medium	In a group meeting try to delegate such tasks to other members; this project must be the priority.
A group member leaves the project, is not disponible or does not dedicate enough amount of time	Project	The project might lack in a certain area due to the mediocre work of a team member	Very High Very Low	The project is a team effort, where we must all work equally to move forward, actions will be taken against this team member
The final product could fail and not perform as expected	Product	An unchecked command could make the difference between a correct and a completely disastrous delivery, leading to errors unknown for the group members	Catastrophic Low	Giving enough time to test the final delivery before the submission, considering all cases and their respective consequence (In case x happens, then each of the results y must be contemplated)

Requirements change mid project production	Project and Requirements	The customer could always ask for a substantial change mid project leading to massive changes	Medium-High Medium-Low	Try to adapt our Gantt Diagram to meet the latest changes with minimum change during the project.
Personnel without sufficient information on the tools to use	Personal and Organization	The lack of experience with the java programming language and its corresponding graphic interfaces could mean spending excessive time learning it and solving errors, thus subtracting time dedicated to other tasks	High-Very High Medium-High	When a doubt arises, asking someone from other departments of the project or another more knowledgeable programmer
Develop the project with incorrect focus	Project and Organization	We follow methods or focus a lot of resources in making something that in the end was not necessary	High Medium-Low	During each session revise the work done, work that will be done, and debate whether it helps us reach the desired goal and not the opposite.
Not understanding what the client wants	Project and Requirements	Our understanding of the client´s vision and ours aren´t in sink	Catastrophic Low	Maintaining active communication with our client and showing results, if possible, an effective way to keep track of the development of the project.
Sudden change in development environment	Technology	The libraries being used for the project could change leading to the need to adapt changes or completely restart the project.	Medium-High Very Low	Before redoing the project, we will consider other technologies that adapt to what we have developed and evaluate if it is better to migrate the project or adapt to changes.
Group members disagree between them	Personal	At the moment of implementing a feature or discussing an idea, teammates may differ in opinions, even resulting in conflict.	Medium Medium	Be open-minded and learn to accept different points of view. Take the opportunity to learn from each other.
Client no longer needs our help	Project	The client may cancel the project mid production for any reason.	Catastrophic Very Low	There is nothing we can do at this point

Section 4: Planning

In this section, we adopt the SCRUM methodology and agile practices to foster iterative development, adaptability, and continuous improvement throughout the project lifecycle. SCRUM emphasizes teamwork, collaboration, and delivering incremental value to stakeholders through short, time-boxed iterations called sprints. Our approach to project planning involves defining and prioritizing tasks, setting achievable goals, and allocating resources effectively to maximize productivity and project success. This robust planning will help us achieve our goals and make sure that we satisfy the needs of our client, Mr. Cámara.

- ***(March 8 - March 13)***: Dedicated to discussions regarding the video-game implementation, creating a list of questions to present to the client on March 13. These questions cover aspects such as boat movements, obstacle generation, boat skins, and levels. Additionally, communication channels through Discord were established, along with task management using Trello and GitHub integration.
- ***(March 13 - March 22):*** With a more refined concept, we implemented the requirement diagram, risk management, role assignment, and future planning. Several meetings were conducted via Discord to oversee the progress of these tasks and their respective reviews.
- ***(March 23 – May 5):*** Refinements for the first submission as requirements and the complete diagrams of the project.
- ***(May 6 – May 17):*** Beta versions of the game were archived, featuring boat movement, obstacles, and basic skins for boats, obstacles, and levels.
- ***(May 18 - May 24):*** Improved versions of the game incorporating physics collisions with obstacles and boats, diverse attributes for boats such as handling or speed, even the increase of difficulty across levels, random object appearances, and mini-games. Graphic developers expanded the repertoire of boat and level skins.
- ***(May 20 - May 24):*** Dedicated to testing the final game, with testers assuming responsibility for this phase. Programmers focused on bug fixes in the code, without introducing additional mechanics. Delivering the project to the client and asking for their feedback.
- ***(May 25 – June 7):*** Final versions from developers with the latest implementations and improvements, potentially including the integration of power-ups during gameplay or enhanced developer-friendly controls. Graphic designers introduced a degraded version of boats to depict damage accumulation, upgrading previous iterations, as necessary.

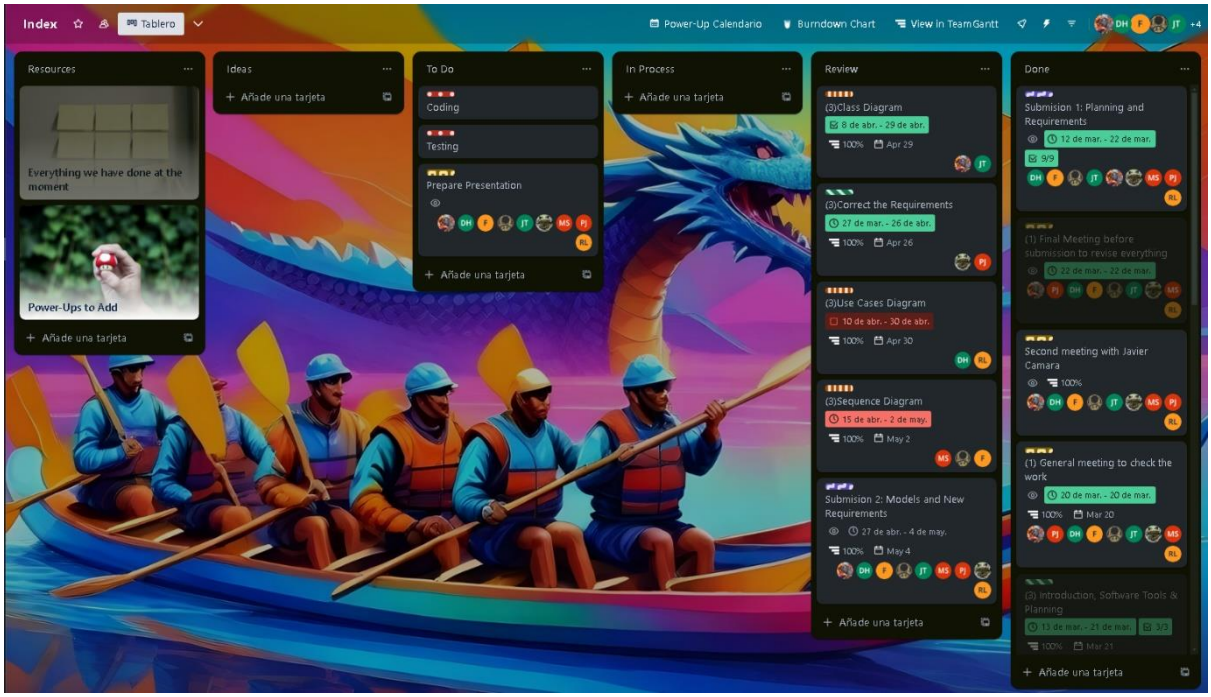

Figure 1: Trello Board

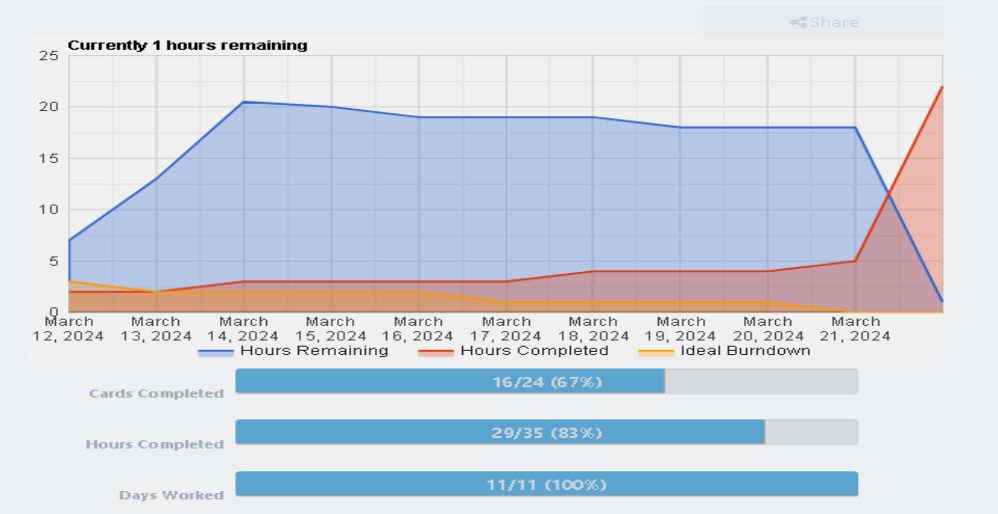

Figure 2: Burnt down chart

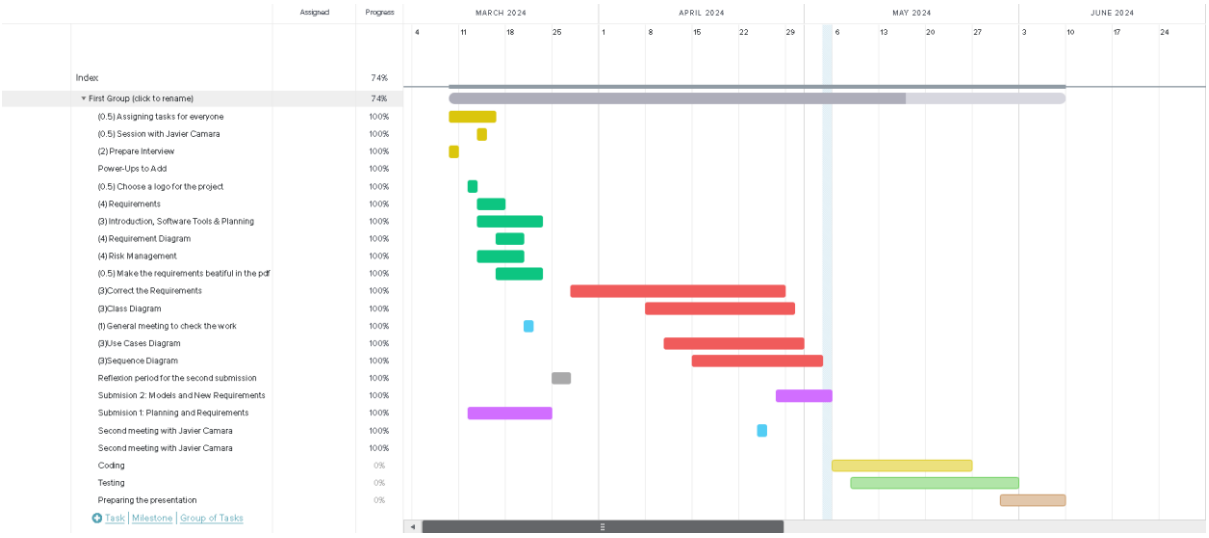

Figure 3: Gantt Chart for the whole project.

Section 5: Software tools

Every project and team need different tools. Some tools are more suitable for the job than others. Here we will list the tools that we have chosen for our Dragon Boat Racing project after carefully looking at all the options, consulting with different team members and of course making sure that they are the best tools for the job.

5.1 Communication

This section outlines the tools we utilize for seamless communication within the team. Effective communication software is crucial for real-time collaboration, swift issue resolution, and maintaining a cohesive team environment. These tools facilitate instant messaging, and videoconferencing ensuring that team members stay connected regardless of geographical locations.

- **Discord** will serve as our main communication platform. The app is great for videoconferencing between our team members, has a modern interface and mobile applications to make sure our team is always connected.

- **WhatsApp** will serve as another communication platform for its convenience, and its instant messaging capabilities as anything critical can be discussed via WhatsApp between our team members for instant response.

5.2 Project planning & productivity

This section enumerates the tools we employ for project planning, task management, and enhancing overall productivity. These tools are essential for organizing workflows, setting project milestones, allocating resources, tracking progress, and managing timelines efficiently. By utilizing such tools, we ensure transparency, accountability, and alignment with project objectives, optimizing team performance and project outcomes.

- **Trello** will be used exclusively as it is very well known in the industry and of course preferred by our project managers. It will make it easy for us to assign tasks and distribute work between our members thanks to its excellent, simple yet powerful web interface.

- **ChatGPT** is an excellent modern AI tool which will help boost the productivity of every one of our members. Be it helping our programmers write tests and similarly helping us document our code due to its great language capabilities.

5.3 Development related

This section delineates the software tools integral to the development process, including integrated development environments (IDEs), and version control systems. These tools facilitate code creation, debugging, version tracking, and testing. By leveraging these tools, we enhance code quality, accelerate development cycles, and foster collaboration among our team's programmers.

- **IntelliJ IDEA** is an IDE very well known in the java development space. It is modern, simple, powerful and above all our team of developers is comfortable using it. It has support for different build tools like ***gradle*** which we will use for our Dragon Boat Racing game.

- **Git** is an excellent version control system and is an industry leader. It is fully free and open source and will give us access to powerful version controlling. As for a git server, we will use ***GitHub*** simply because it does not require us self-hosting our git server, is easy to use and provides an excellent desktop client for us to use.

5.4 Design

This section highlights the tools utilized for graphical design, prototyping, and user interface (UI) development. These tools enable our graphic designers to conceptualize, iterate, and refine design elements, ensuring intuitive user experiences and visually appealing interfaces. From wireframing and mockup creation to asset management and prototyping, these tools play a pivotal role in translating design concepts into tangible product features, aligning with our client Mr. Cámara's expectations.

- **Visual Paradigm** is versatile tool for UML diagramming and requirements definition. It offers a user-friendly interface and a wide range of features for creating diagrams like use case, class, and sequence diagrams. Facilitating collaborative design and analysis ensures efficient communication and alignment of design concepts between us and our client, Mr. Cámara. It is simply put ideal in our case of agile development.

- **Canva** is an excellent modern graphic design tool which serves to create simple mock-ups as well as different marketing and mock-up material when working in our team.

- **Adobe Creative Cloud** is a comprehensive suite of creative tools for graphic design, photo editing, video production, and web development. Adobe Creative Cloud includes industry-standard software such as **Photoshop** and **Illustrator** which will be used by our professional

graphic designers to design different graphical elements of the game be it the simple game menu or complex boat designs.

- **Freepik Pikaso** is an AI-powered art generator that revolutionizes the creative process. Leveraging advanced algorithms, Pikaso transforms concepts into stunning visual artworks effortlessly. With its intuitive interface and vast library of styles and elements, Pikaso will empower our graphic designers to reach their maximum potential be it generating photos of different game elements or simply using it to get inspired when they are out of ideas.

5.5 Documentation

This section identifies the tools employed for documenting project requirements, specifications, codebase, and user documentation. Effective documentation tools facilitate knowledge sharing, maintain project transparency, and serve as invaluable references for our team and are important for our client Mr. Cámara.

- **Google Docs** is a cloud-based document collaboration platform offering real-time editing and sharing capabilities. Google Docs enables our team members to work concurrently on documents, facilitating seamless collaboration regardless of geographical locations.

- **Microsoft Office** is a suite of productivity tools including Word, Excel, and PowerPoint, providing robust capabilities for document creation, data analysis, and presentation. Microsoft Office offers a familiar interface to our team members and is an industry standard.

Figure 4: Best software tools for job

Section 6: Requirements

In this section, there will be explained all the requirements that this project must hold for its correct functionality, those whose color is green, are meant to be the functional requirements, those who explain what the project must do. Furthermore, those whose color is yellow will be denoted as the Non-Functional Requirements, that are explained as the qualities that the project holds itself.

FR001: PLAYER
As a player I want to control a boat to reach the end of the race.

FR002: BOATS
As a player I want to be able to choose between boats of different characteristics.

FR003: LEVELS
As a player I want to have different levels in order to get a sense of progress and feel an increasing difficulty as I advance.

FR004: OBSTACLES
As a player I want things that impede my goal including random dynamic obstacles and static obstacles that do not change.

FR005: MAIN MENU
As a player I want a main screen that allows me to navigate through other screens though buttons.

FR006: POWER-UPS
As a player I want to have pick-ups throughout the race that alter my speed, health and give invincibility.

FR007: LOADING SCREENS
As a player I want to see a decorated screen screen while things are loading, including a loading bar to keep the player informed of the progress.

FR008: TUTORIAL
As a player I want to have an explanation of how the game works before playing.

FR009: LEGS
As a player I want there to be 3 legs thoughout each game.

FR010: SCENERY
As a player I want to see a nice dragon-themed environment throughout the race that becomes more abundant every leg.

FR011: HEATH-BAR
As a player I want to know my boat's health throughout the race.

FR012: PAUSE SCREEN
As a player I want to be able to pause the game and resume it whenever I want.

FR013: RIVALS
As a player I want to compete against other boats.

FR014: LEG DURATION
As a player I want legs not to exceed 2 minutes of duration.

FR015: TIMER
As a player I want to know how long I am taking to finish the race.

FR016: BOAT SELECTION MENU
As a player I want to have a menu screen before the race that allows to choose between boats.

FR017: LEADERBOARD
As a player I want to know in which position of the race I am with respect to rivals.

FR018: CONTROLS
As a player I want to be able to change the controls however I want.

FR019: MINI-GAME
As a player I want a secondary minigame different in style from the main game. This game gives a second chance to the player to continue the game.

FR020: LANES

As a player I want boats to have their own lane to try staying in, and to slow down whenever they go out.

FR021: VISUAL EFFECTS
As a player I want simple animations after the trigger or certain events, such as the crash of the boat with an obstacle or the end of the race.

FR022: CREDITS
As a player I want to know the developers of the game.

NFR001: JAVA
Using Java language to develop the game as it is a technical constraint imposed by the client.

NFR002: FRAME RATE
As a player I want the game running always at a constant 30fps to give a smooth gameplay and avoid penalties external to player skill.

NFR003: LOW LATENCY RESPONSES
As a player I want game responses not to take more than 50ms.

NFR004: COMAPCT
As a user I want the executable files to be as light as possible (less than 1Gb).

NFR005: RESOLUTION
As a player I want the game to be playable in resolutions 1920x1080, 1280x720, 1024x768, 800x600.

Section 7: Requirements diagram

For every project which has requirements to classify, there must be a diagram that shows all the relations between Functional Requirements and Non-Functional ones, this part consists of the definitive version of the diagram with all the requirements of this projects and how everyone affects others.

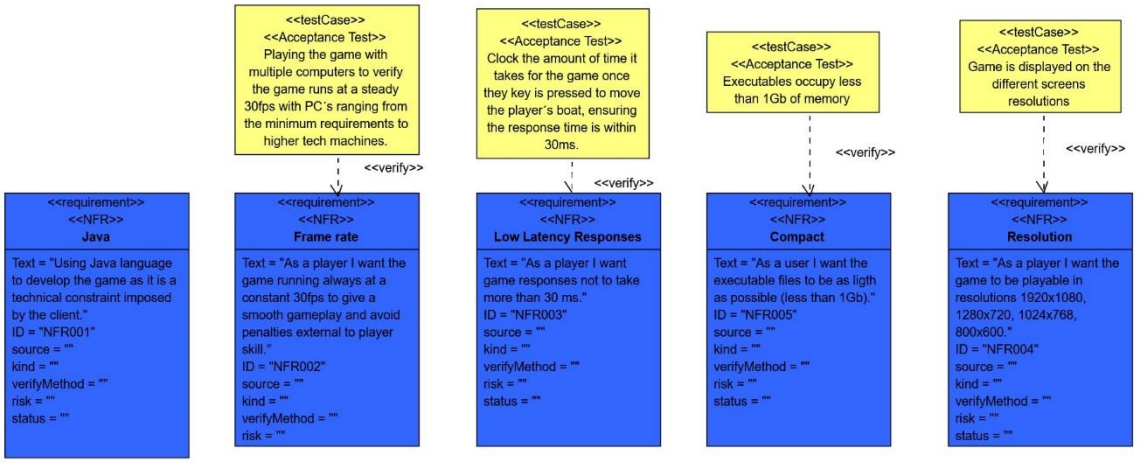

Figure 5: Requirements diagram: NFR part

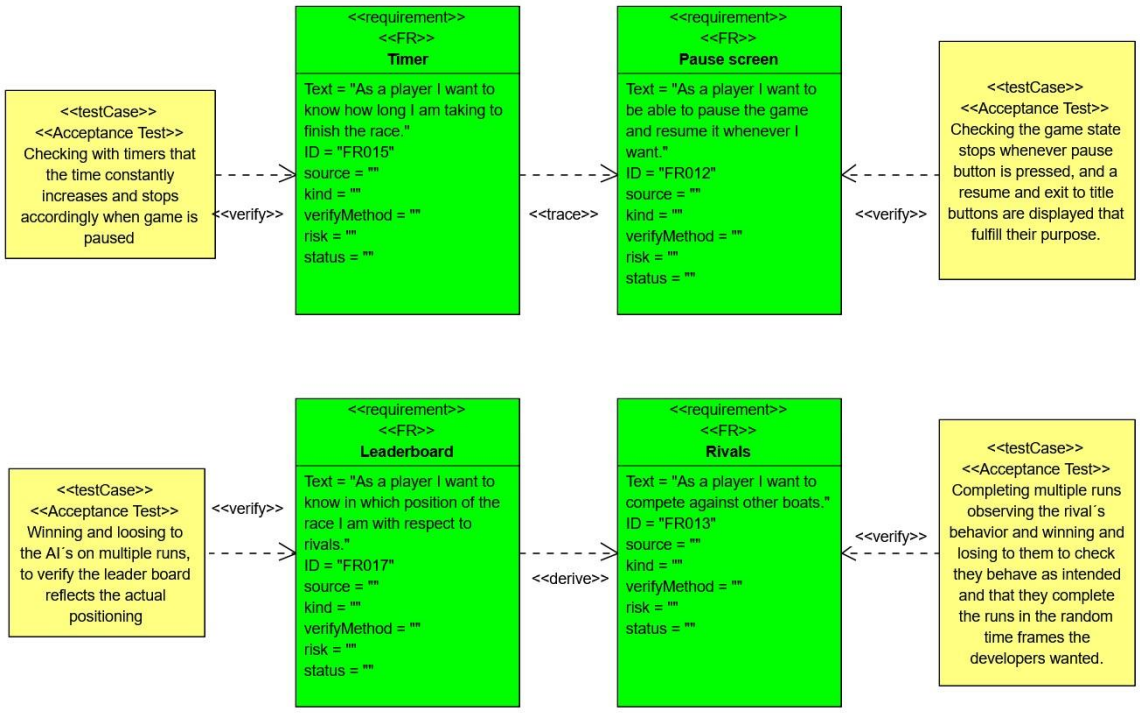

Figure 6: Requirements diagram: FR part 1

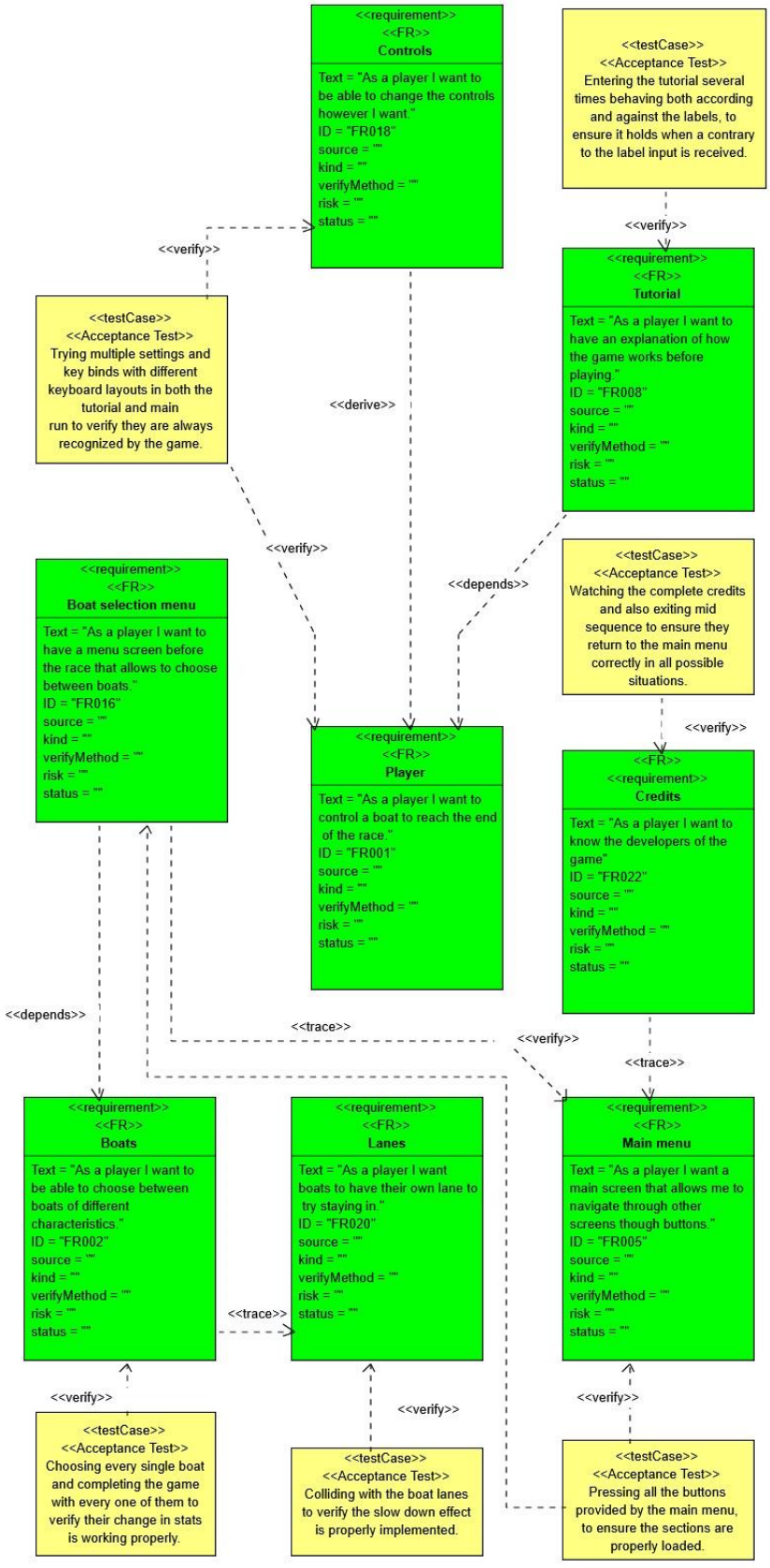

Figure 7: Requirements diagram: NFR part 2

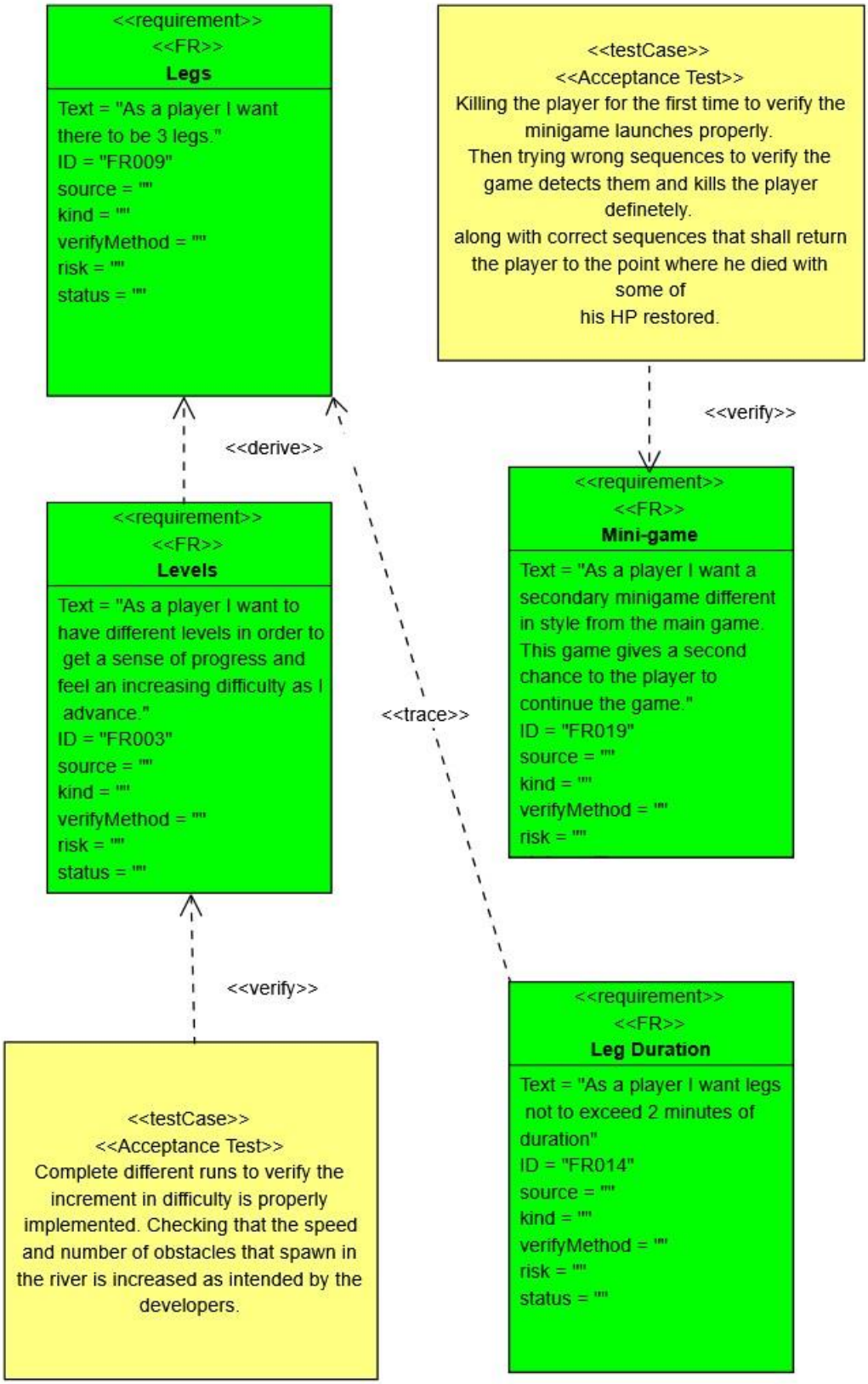

Figure 8: Requirements diagram: NFR part 3

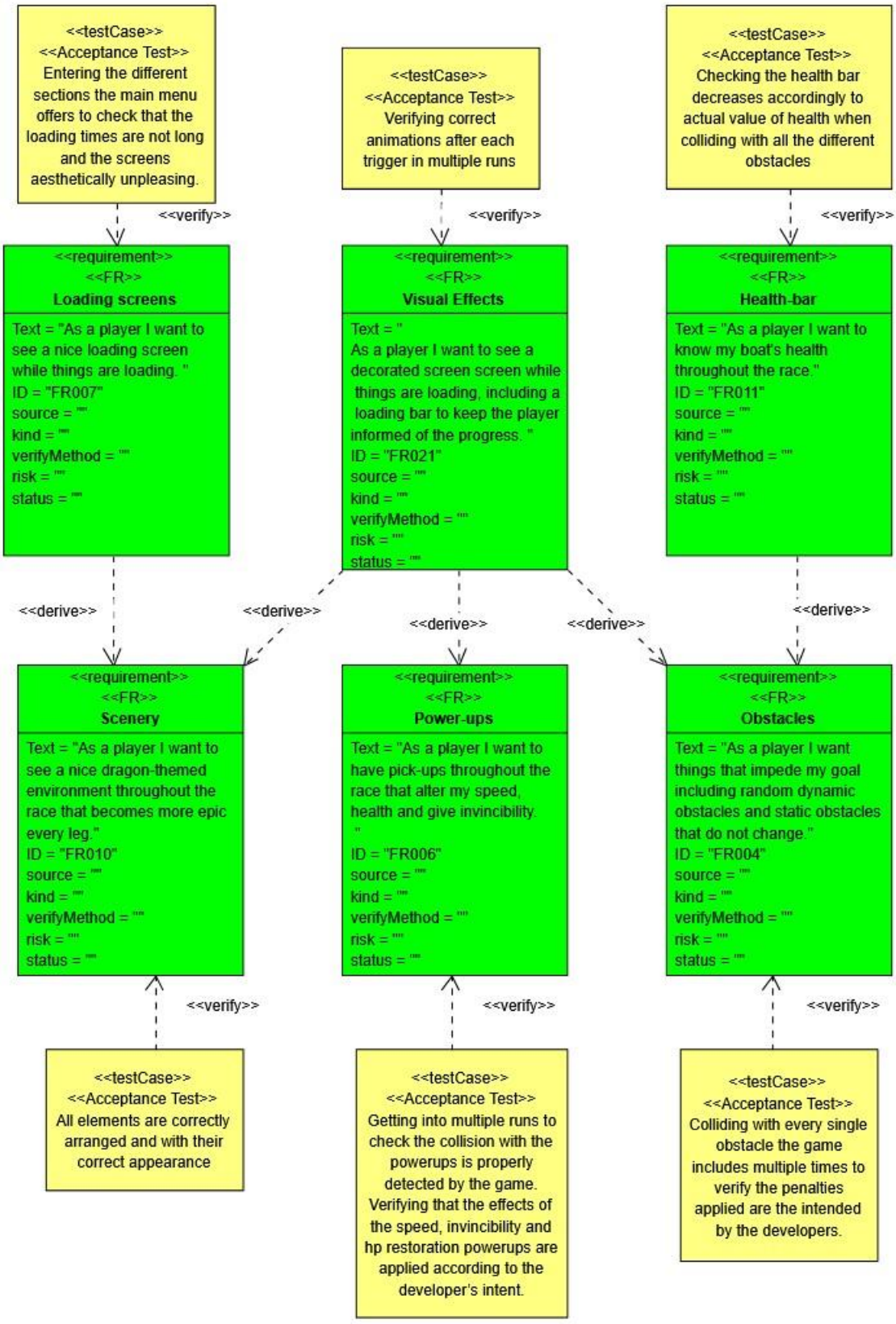

Figure 9: Requirements diagram: NFR part 4

Section 8: Use Case Diagram

Diagram

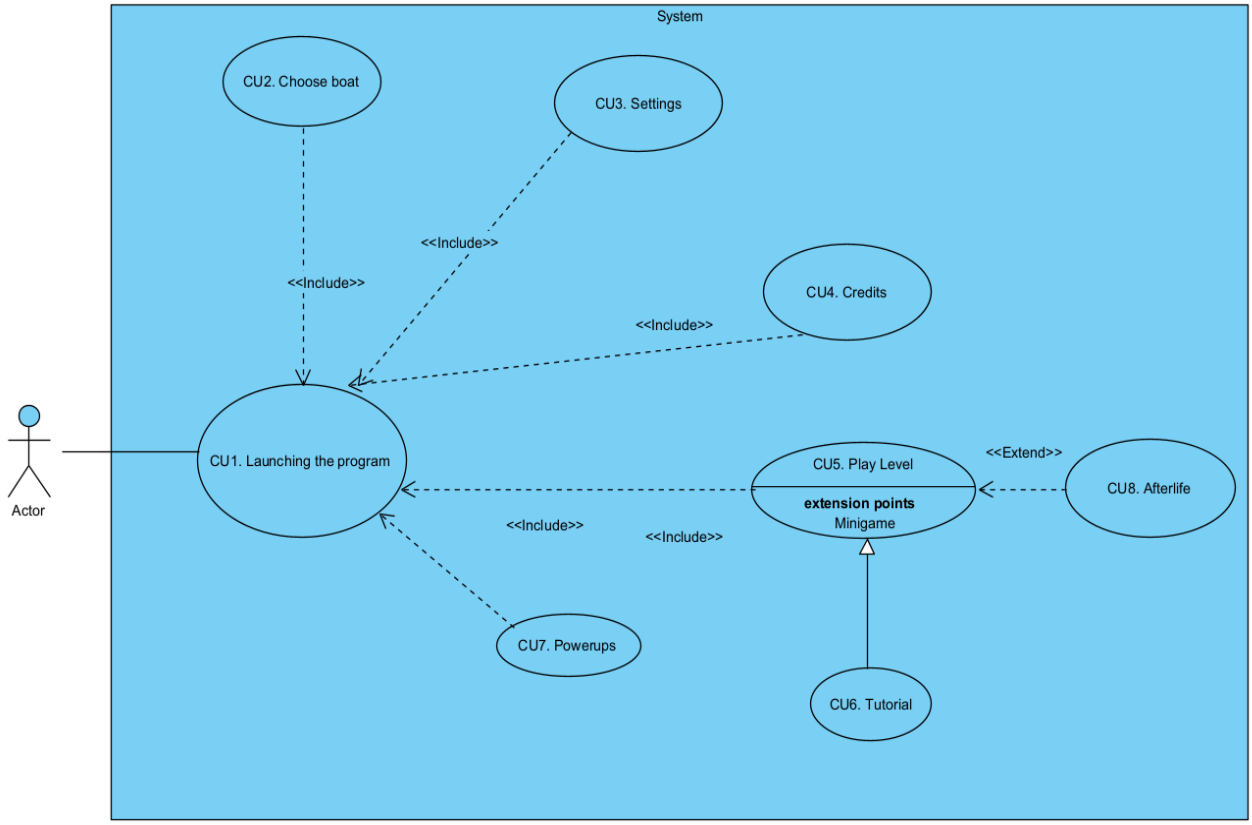

Figure 10: Use case diagram

Description

ID: UC1. Launching the program

Context of use

• The user clicks on the executable to launch the game. Once this is done a menu is displayed in which the player can select between "play level", "choose boat", "settings", "tutorial", "credits" and "exit to desktop".

Preconditions and activation

• The program is properly installed, and the process launches once the user runs the executable.

• The user fulfills the minimum requirements in his machine.

Guarantees of success or post-conditions

• The program properly displays the main menu.

Main scenario

• The user has installed the game.

• The user launches the program properly.

• The menu is loaded and displayed on the user´s screen.

• The user can navigate through the main menu.

Alternative scenarios

• The user´s computer does not have the necessary resources to run the game appropriately.

• There has been a problem with the installation and the executable behaves oddly.

ID: UC2. Choose boat

Context of use

• When the user clicks on the "choose boat" button, he will be able to see all the boats and their trade-offs, and he will be able to navigate through them to choose the one he prefers the most.

Preconditions and activation

• The user is already running the game and in the main menu

Guarantees of success or post-conditions

• The boat selection menu is properly displayed.

• The user can navigate through the menu and see the boats.

• Once the user selects a boat, changes are applied and shown is his next run.

Main scenario

• The user is on the main menu.

• He clicks on the "choose boat" button.

• The user can see all the boats.

• The user selects a boat, and that boat is loaded for the next run.

• The user goes back to the main menu with the new changes applied.

Alternative scenarios

• The user hasn´t played the game before and in case he decides to play the level before selecting a boat, there is already a default one selected for him.

• In case the user enters the boat selection menu and exits it without selecting a new one, the previous selection still holds.

ID: UC3. Setting

Context of use

• When the user clicks on the "settings" option, a menu in which he can configure various aspects of the game is displayed.

Preconditions and activation

• The user is already running the game and in the main menu.

Guarantees of success or post-conditions

• The user can configure the settings to his liking bounded by the options he is given.

Main scenario

• The user is on the main menu.

• The user wants to change some aspect of the game´s settings.

• The user clicks on the "settings" button and key binds and volume options are displayed, including a mute option.

• The user modifies either the key binds, the volume or both and saves the changes. The game´s behavior is modified accordingly.

• The user goes back to the main menu with the changes applied.

Alternative scenarios

• The user does not change any settings and the game keeps behaving in the same way it did before.

• The user makes some changes but does not save them. Not altering the way, the game behaves.

ID: UC4. Credits

Context of use

• The user is on the main menu, and he wants to see the credits of the game.

Preconditions and activation

• The user is already running the game and in the main menu.

Guarantees of success or post-conditions

• The user can see all the people that participated in creating the game once he clicks on the "credits" button.

Main scenario

• The user is on the main menu.

• The user wants to see who created the game.

• The user clicks on the "credits" button.

• A video is displayed in which the player can see all the people that were involved on the game development.

• Once the video ends, the player returns to the main menu.

Alternative scenarios

• The player decides to finish the video abruptly by pressing "Esc" and he is returned to the main menu without issues.

ID: UC5. Play Level

Context of use

• The user wants to play the game, so once he is on the main menu he clicks on the "play level" button, initializing the run with the default settings and boat or the ones he had previously selected.

Preconditions and activation

• The user is already running the game and in the main menu.

• The user´s computer is powerful enough to play the level, as it is more demanding than launching the game and main menu navigation.

Guarantees of success or post-conditions

• The run will launch.

• The boats, obstacles and scenery will be loaded on screen.

• The user will be able to complete the run and control the boat throughout it.

Main scenario

• The user is already on the main menu and decides to click on the "play level" button.

• Once the button is pressed the user will enter the run with the selected boat and settings.

• Inside the run the player can control the boat and avoid obstacles with a response time of 30ms.

• The user can pause the game and a menu with the options "resume "and "exit to title" is displayed.

• If the player crashes with enough obstacles he will lose all his hp and, the first time, an after-life Simon says like mini-game will be launched.

• If the player completes the mini game successfully his hp will be restored and he will come back to the point in which he died.

• Once the run is complete, he will advance to the next level or if found in the last one he will win the game.

• After finishing the run, either after winning or losing, he will have the opportunity to play the level again or to return to the main menu.

Alternative scenarios

• The user decides to exit mid-game returning to the main menu or desktop without issues.

ID: UC6. Tutorial

Context of use

• The user wants to play the tutorial, so once he is on the main menu he clicks on the "tutorial" button, initializing the run with the default settings and boat or the ones he had previously selected.

Preconditions and activation

• The user is already running the game and in the main menu.

• The user´s computer is powerful enough to play the tutorial, as it is more demanding than launching the game and main menu navigation.

Guarantees of success or post-conditions

• The tutorial will launch.

• The boats, obstacles and scenery will be loaded on screen.

• The user will be able to complete the tutorial and control the boat throughout it while the labels are displayed.

Main scenario

• The user is already on the main menu and decides to click on the "tutorial" button.

• Once the button is pressed the user will enter the tutorial with the selected boat and settings.

• Inside the tutorial the player can control the boat and avoid obstacles.

• The user can pause the tutorial and a menu with the options "resume "and "exit to title" is displayed.

• During the tutorial the user is introduced to the game mechanics by pop up labels that explain the basic aspects of the game, including a movement practice, where he is told to move right and left, and then he is asked to avoid some obstacles.

• He will also be introduced to the afterlife minigame. He will be killed and shown the mechanics of it.

• And he will be shown the mechanics of the powerup system, by colliding with one powerup and then activating it. While the labels explain the behavior.

• Once the tutorial is completed the user will be returned to the main menu.

Alternative scenarios

• The user decides to exit mid-tutorial returning to the main menu or desktop without issues.

ID: UC7. Powerups

Context of use

• While the user is found in the game, some powerups will be placed along the river.

• The user can, by colliding with them, activate one of the three powerups the game offers: higher speed, invincibility, and healing.

Preconditions and activation

• The user is already running the level.

Guarantees of success or post-conditions

• The user has collided with the powerup, and the according effect is applied.

Main scenario

• The user is already in the run.

• While he plays the level some powerups will be displayed along the river.

• If he collides with the powerup, the effects of it will be applied.

• If he collides with the healing powerup, some of his health will be restored.

• If he collides with the speed powerup, his velocity will be increased for some period.

• If he collides with the invincibility powerup, his will not receive damage for some period.

• After the period comes to an end, and in case the powerups with which he collided with where speed or invincibility, his stats will return to normal values.

Alternative scenarios

• The user does not collide with any powerups.

• The user collides with the healing powerup having more HP remaining than the amount received by it.

ID: CU8. Simon says minigame

Context of use

• When the user dies for the first time a minigame with Simon says like mechanics is loaded on screen.

Preconditions and activation

• The user has lost all his HP for the first time in the run.

Guarantees of success or post-conditions

• The user can play a game in which he is given the opportunity to return to life.

• In case he ends up winning, he will be able to return to the river where he left it.

Main scenario

• The user is already in the run.

• The user collides with enough obstacles to lose all his HP.

• Once he does so, a minigame is loaded on screen.

• In this game he is shown an increasingly difficult sequence he must replicate.

• In case he fails reproducing the sequence he will die, having the opportunity to play the main level again or to return to the main menu.

• In case he succeeds, he will return to the point where he died, with some of his HP restored.

Alternative scenarios

• The user decides to leave mid-game, returning to the main menu of exiting to the desktop.

Section 9: Domain Model

As with every programming project, a diagram which includes all the classes that are going to be developed among the programmers is necessary. This is the final version of the complete diagram. There is a briefly discuss of every class after the image:

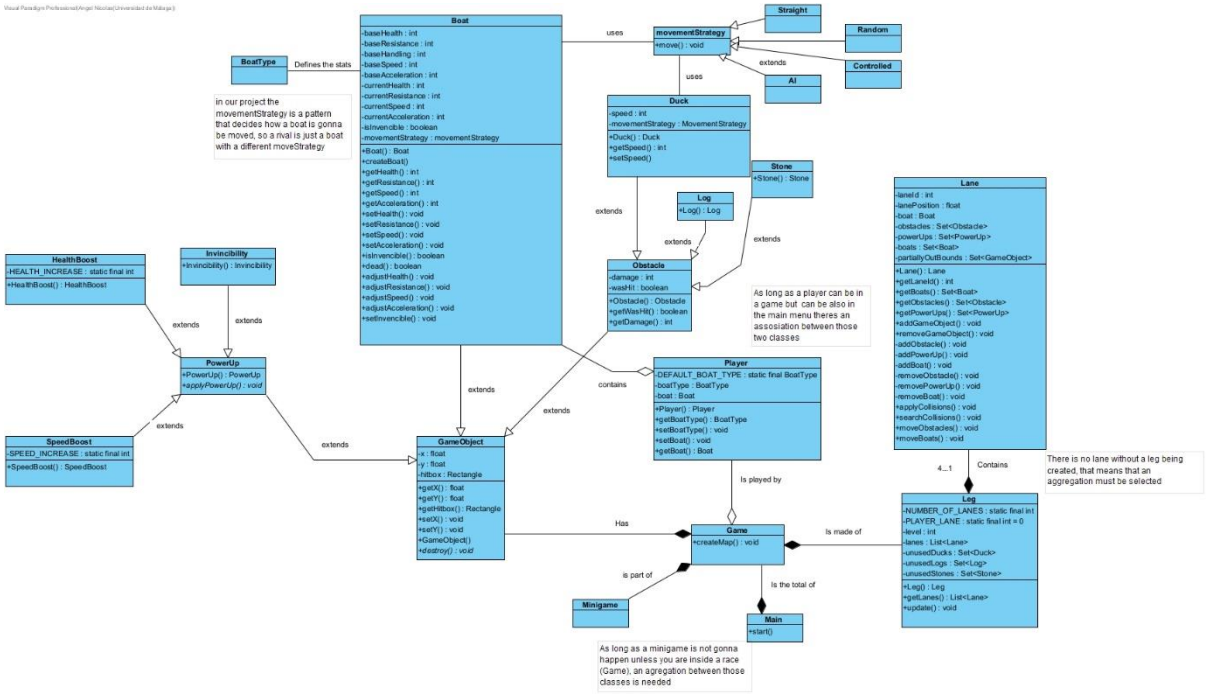

Figure 11: Class diagram

Section 10: Sequence Diagram

The sequence diagram illustrates the interaction between objects in the system and how they behave dynamically. It's important to understand the flow of messages and the communication patterns throughout the game.

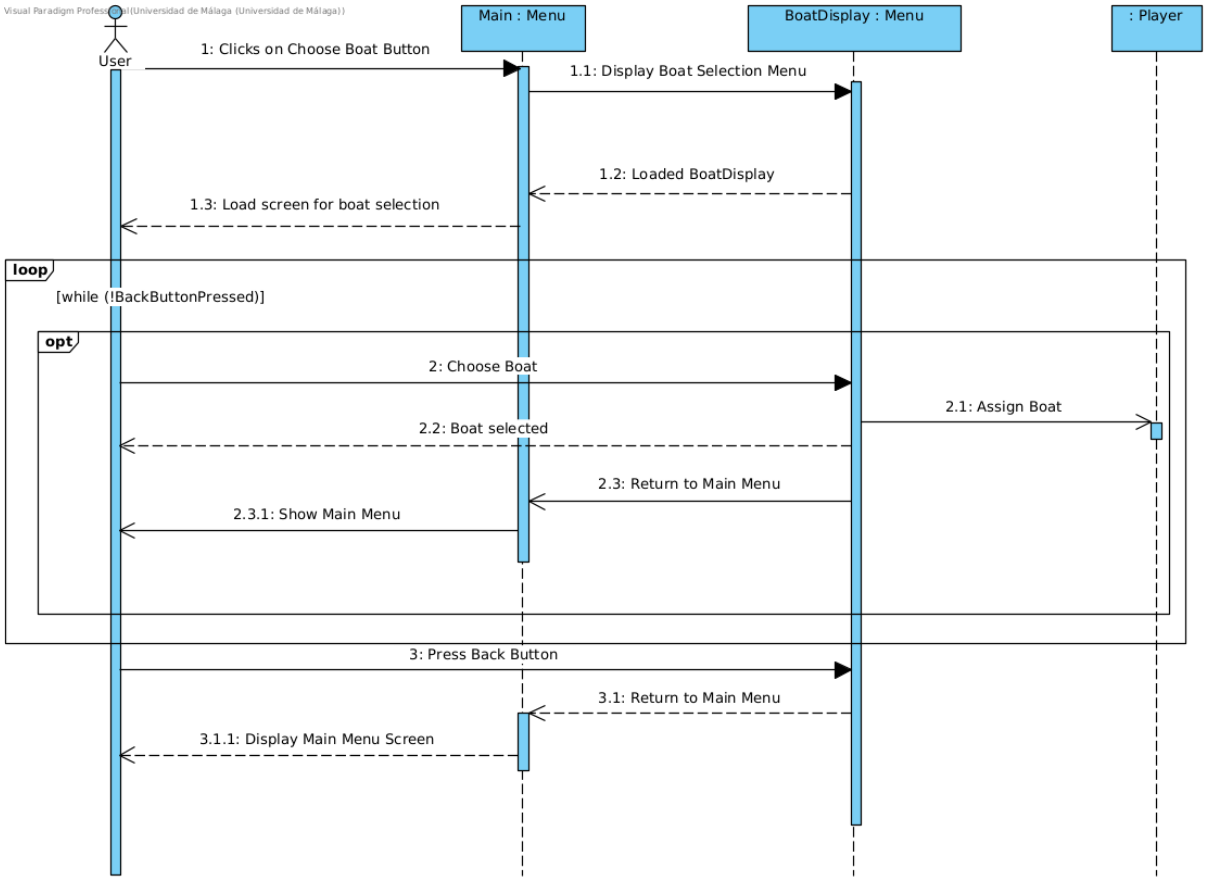

Figure 13: Sequence diagram UC2 Choose boat

The lifelines involved in this sequence diagram are:

- User
- Main Menu
- BoatDisplay Menu
- Player

In Use Case 2 (CU2), "Choose Boat", depicts the interactions between various objects when a user selects a boat. The process starts with the user clicking the "Choose Boat" button on the main menu. The main menu responds by creating and displaying the BoatDisplay. The BoatDisplay Menu then displays the list of boats to the user. The user selects a boat, which is then communicated back to the BoatDisplay Menu. The selected boat is assigned to the player. The player can change the boat as many times as he wants. Finally, the BoatDisplay Menu returns the user to the main menu when the "Back" button is pressed.

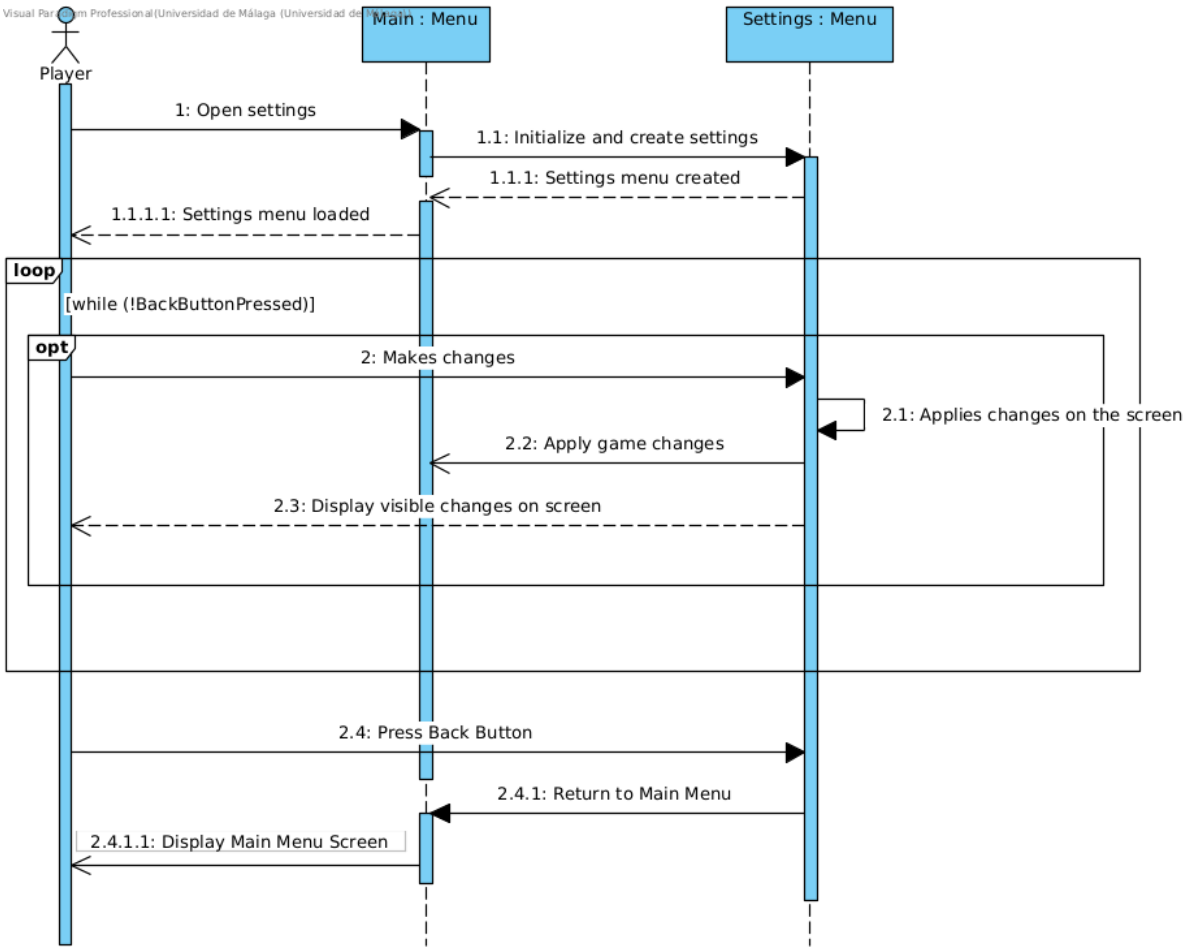

Figure 14: Sequence diagram UC3 Settings

The lifelines involved in this sequence diagram are:

- User
- Main Menu
- Settings Menu

In Use Case 3 (CU3), "Settings," the sequence begins with the user interacting with the main menu to open the settings. Upon clicking the "settings" button, the main menu initializes and creates the settings menu, which is then displayed to the user. The user makes changes within the settings menu, such as adjusting key binds or volume settings. When the user makes changes, they are saved automatically, and he can continue to make more changes until Back button is pressed. Once pressed, the settings menu is closed, and the user is taken to the main menu.

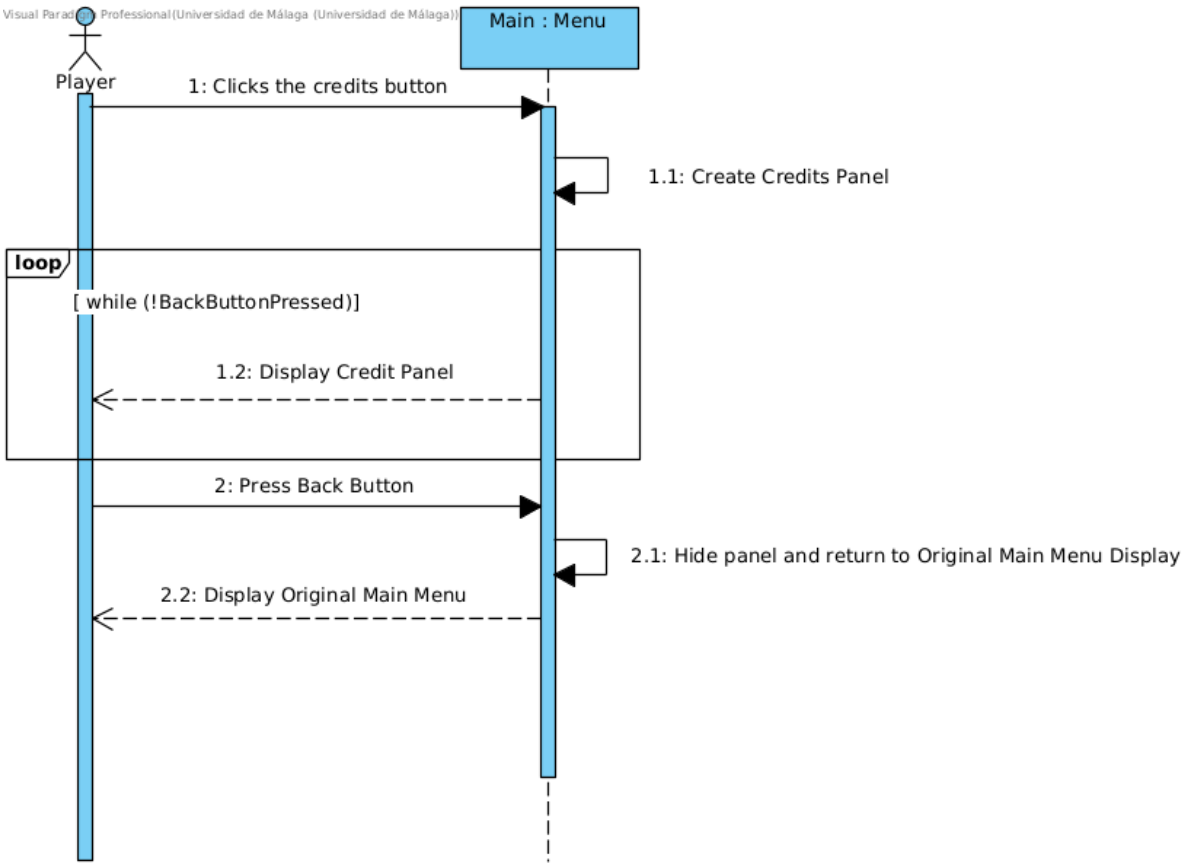

Figure 15: Sequence diagram UC4 Credits

The lifelines involved in this sequence diagram are:

- User
- Main Menu

In Use Case 4 (CU4), "Credits", the sequence begins with the user interacting with the main menu to see the credits. Upon clicking the "credits" button, the main menu displays a panel with the complete name of all the collaborators of the project. The user must press the Back button to hide the credits panel and return to the main selection menu.

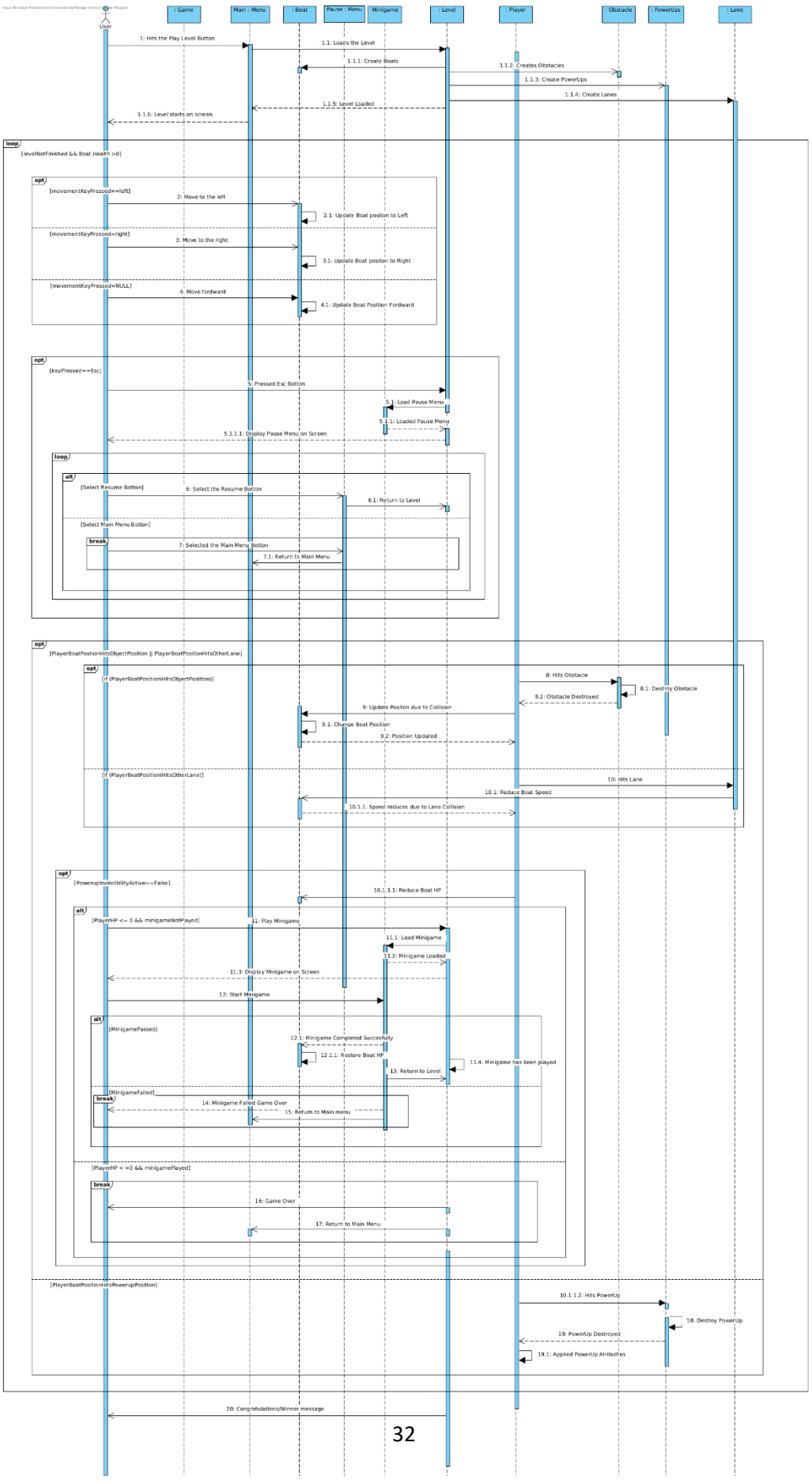
: Game
Main : Menu
: Boat
Pause : Menu
Minigame
: Level
: Player
: Obstacle
: PowerUps
: Lane
User
1: Hits the Play Level Button
1.1: Loads the Level
1.1.1: Create Boats
1.1.2: Creates Obstacles
1.1.3: Create PowerUps
1.1.4: Create Lanes
1.1.5: Level Loaded
1.1.6: Level starts on screen
loop
[levelNotFinished && Boat.Health >0]
opt
[movementKeyPressed=left]
2: Move to the left
2.1: Update Boat positon to Left
[movementKeyPressed=right]
3: Move to the right
3.1: Update Boat positon to Right
[movementKeyPressed=NULL]
4: Move fordward
4.1: Update Boat Position Fordward
opt
[keyPressed==Esc]
5: Pressed Esc Botton
5.1: Load Pause Menu
5.1.1: Loaded Pause Menu
5.1.1.1: Display Pause Menu on Screen
loop
alt
[Select Resume Botton]
6: Select the Resume Botton
6.1: Return to Level
[Select Main Menu Botton]
break
7: Selected the Main Menu Botton
7.1: Return to Main Menu
opt
[PlayerBoatPostionHitsObjectPosition || PlayerBoatPositonHitsOtherLane]
opt
[if (PlayerBoatPositioniHitsObjectPosition)]
8: Hits Obstacle
8.1: Destroy Obstacle
8.2: Obstacle Destroyed
9: Update Positon due to Collision
9.1: Change Boat Position
9.2: Position Updated
[if (PlayerBoatPositioniHitsOtherLane)]
10: Hits Lane
10.1: Reduce Boat Speed
10.1.1: Speed reduces due to Lane Collision
opt
[PowerupInvincibilityActive==False]
10.1.1.1: Reduce Boat HP
alt
[PlayerHP <= 0 && minigameNotPlayed]
11: Play Minigame
11.1: Load Minigame
11.2: Minigame Loaded
11.3: Display Minigame on Screen
12: Start Minigame
alt
[MinigamePassed]
12.1: Minigame Completed Succesfully
12.1.1: Restore Boat HP
11.4: Minigame has been played
13: Return to Level
[MinigameFailed]
break
14: Minigame Failed Game Over
15: Return to Main menu
[PlayerHP < =0 && minigamePlayed]
break
16: Game Over
17: Return to Main Menu
[PlayerBoatPositionHitsPowerupPosition]
10.1.1.2: Hits PowerUp
18: Destroy PowerUp
19: PowerUp Destroyed
19.1: Applied PowerUp Atributtes
20: Congratulations/Winner message

Figure 16: Sequence Diagram UC5 Play Level

The lifelines involved in this sequence diagram are:

- User
- Game
- Main Menu
- Boat
- Pause Menu
- Minigame
- Level
- Player
- Obstacle
- PowerUp
- Lane

In Use Case 5 (CU5), "Play level" shows the different interaction between the player and various components during the race. Initially, the player hits the play level button, prompting the game to load the level, which includes creating the assets (player, obstacles, boats, power-ups and lanes).

Once the level is loaded, in every iteration we check if the player presses a movement key (left or right) or does not (move forward), and the game updates the boat's position accordingly. If the player presses the escape button ("Esc"), the pause menu is loaded and displayed, allowing the player to either resume the game or return to the main menu. Selecting the resume button returns the player to the level, while selecting the main menu button sends them back to the main menu screen.

Throughout the gameplay, the player interacts with obstacles and power-ups. If the player's boat collides with an obstacle, the game updates the boat's position due to the collision, destroying the object, and reducing the boat's HP only if the Invincibility Powerup is not active (disable). Other source of collision is if the Player goes to another lane other than his own, also reducing boat's speed and HP. If the boat's HP reaches zero, a minigame is triggered, which is created and displayed on the minigame screen.

The self-message of "minigame has been played" will be a value saved by the level for future checks when the player dies. If the value of "minigame has been played" is equal to true, on the next player death the game will not trigger the minigame level again and end the current run. Otherwise, the minigame starts on player death. The player must successfully complete the minigame to restore the boat's HP and return to the level on the same position of death. In the case they fail, it results in a game over, returning them to the main menu.

Additionally, the player can interact with power-ups scattered randomly throughout the level. When the boat hits a power-up, its effect is applied to the boat enhancing gameplay. If the player manages to avoid obstacles and effectively use power-ups, they eventually reach the goal, triggering a victory message after completing all lanes.

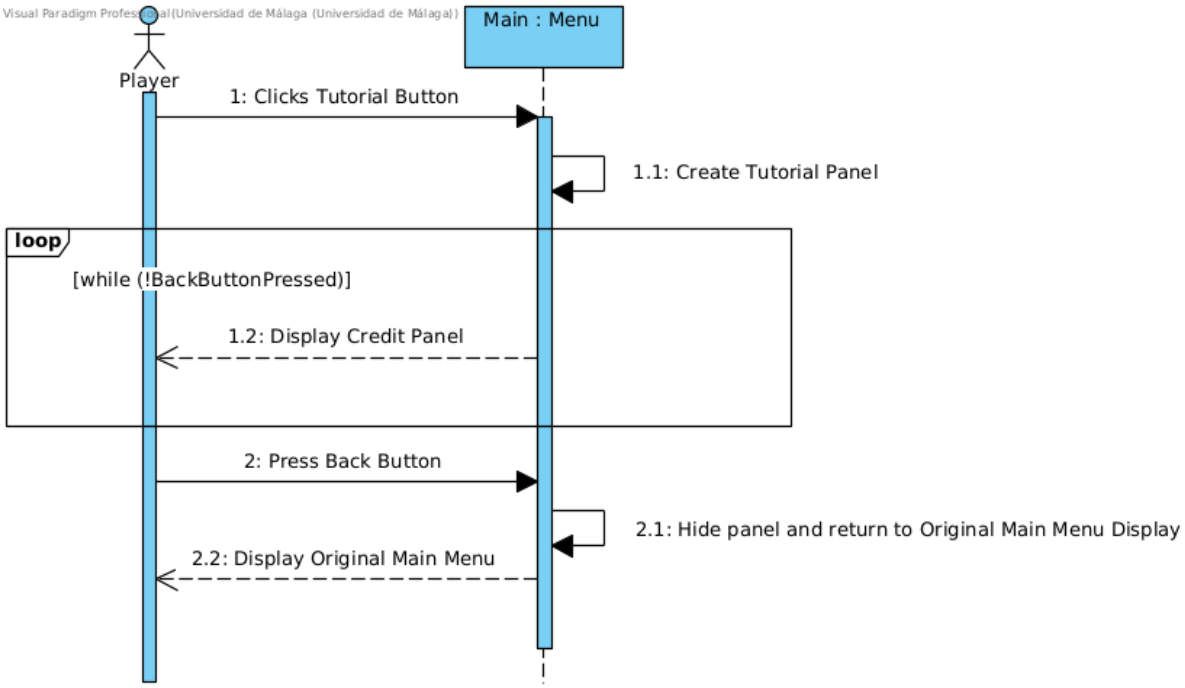

Figure 17: Sequence Diagram UC6 Tutorial

The lifelines involved in this sequence diagram are:

- User
- Main Menu

In Use Case 6 (CU6), "Tutorial", the sequence begins with the user interacting with the main menu to see the tutorial of the game. Upon clicking the "tutorial" button, the main menu displays a panel with a brief description of the game mechanics the player should be aware of during gameplay and introduction to the minigame. The user must press the Back button to hide the tutorial panel and return to the main selection menu.

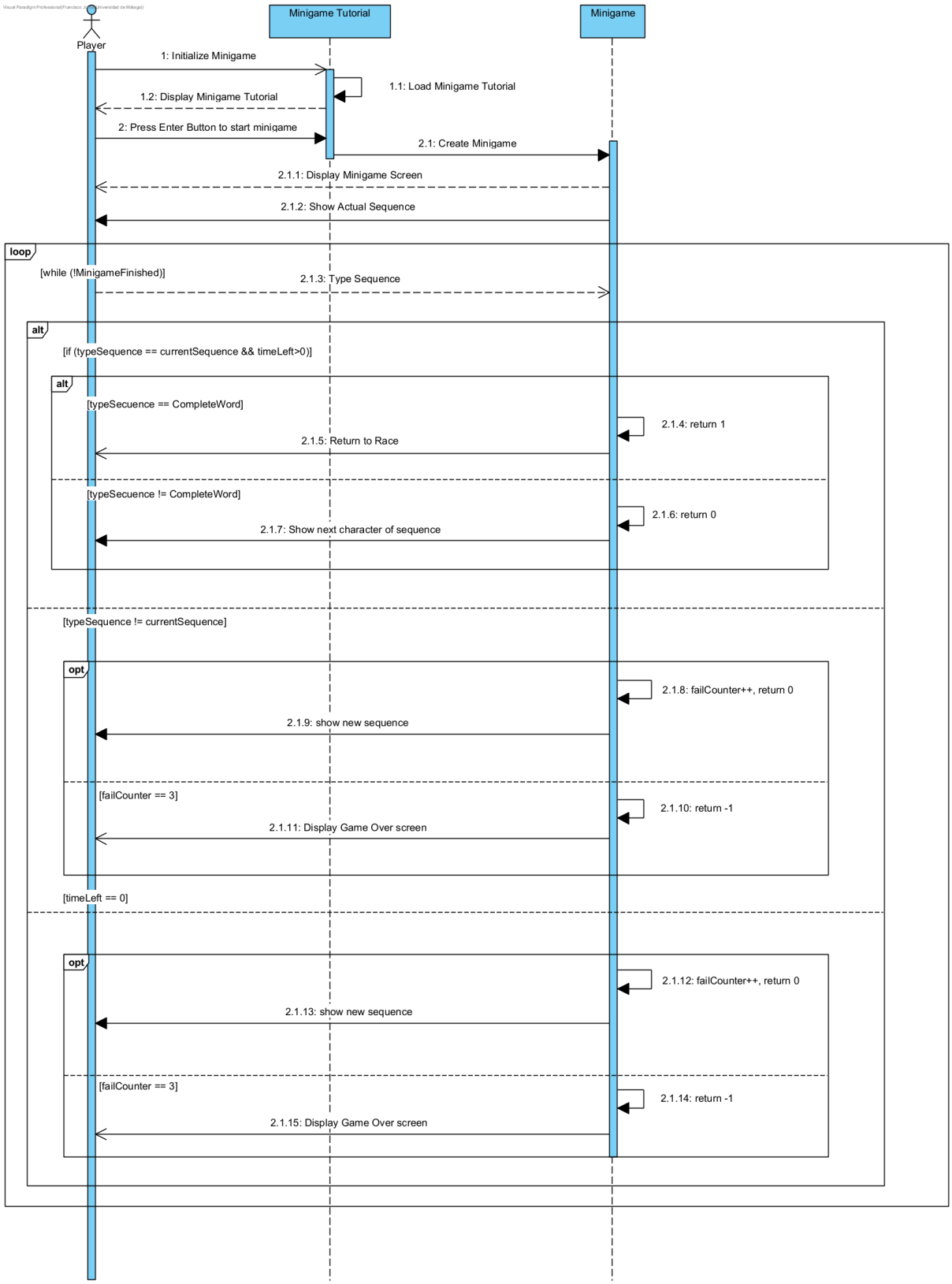

Figure 19: Sequence Diagram UC8 Simon Says Minigame

The lifelines involved in this sequence diagram are:

- Player
- Minigame
- Minigame Tutorial

In Use Case 8 (CU8), "Simon says" minigame, the sequence diagram illustrates the interaction between the player and the minigame system when the player's boat HP is reduced to 0. First, it is introduced to a short tutorial of how the minigame should be played; the player must follow the sequence presented on the screen.

One character will be shown (cAseSenTitiVe), the reply must be the same character. On each successful character type, a new character of the same word will be displayed, the player must retype all the previous characters in order. Once the word is completed, the player is returned to the race at the start of the leg where he first died, with the boat's HP replenished. In the case a character is mistyped, a new word will have to be completed, resetting any previous progress with a sequence; this decreases one of the three attempts available to complete the minigame.

The minigame also has a timeout fail system, meaning the player must type the sequence before the given time runs out. On every successful character type, time is restored to the maximum amount given. If the player fails to complete the minigame in the given attempts, he will be taken to a Game Over screen ending the current run; by pressing 'Enter' the player is returned to the main menu to start a new game.

Section 11: Test Cases

For any long term project, it is essential to maintain a level of security where everything implemented works correctly as specified in the documentation, this is where the tests of the most important entities of the project come into play, here we detail the tests that will involve the game and its predictions as well as its implementation:

Index:

Gameplay:

- TC_Boat
- TC_Lane
- TC_Obstacle
- TC_ObstacleVisitorTest
- TC_CollidableTest

Client Specifications:

- TC_Powerups

Utilities:

- TC_KeyBindings

We decided on ordering the tests by what we felt has higher priority. Firstly, we test all JUnit possible tests essential for the game, Boat, lanes, legs and obstacles. Then proceed to test the client requirements as our main goal is satisfying all the requirements of our client. And finally, we have a test for the assignation of Key Binds that we decided on giving its own category.

Testing Plan

Project Name:

UMA-ISE24-E1

Authors:

Ricardo Juan Allitt López, Manuel Barrios Moreno, Ángel Bayón Pazos, Ángel Nicolás Escaño López, Pablo Hormigo Jiménez, Francisco Javier Jordá Garay, Diego Sicre Cortizo, Muhammad Abdullah Sultán Sultán, Juan Torres Gómez.

Gameplay:

This section includes the most significant tests for the correct execution of the program, in case you want to test functions, these should be the main ones.

Test Case Overview

Test Case ID:

TC_Boat

Purpose:

Verify that boats can be created properly, ensure correct movement and test for inconsistencies when modifying their attributes.

This test cases involves FR002(Boats).

Test Case Description:

This test case checks the correct functioning of the class boats and its checkable methods.

JUnit tests related to this test case will check methods related to the creation, attributes, movement and destruction of the boat.

Pre-Conditions

Prerequisites:

Class Boat and its methods are properly implemented.

Test Data:

Requires boat type and their map positions for initial setUp function.

Test Steps

Step Description:

1. *SetUp function is called before every test to create a boat of each type.*
2. *The first test method tests the createBoat method from class Boat to ensure correct functioning of the createBoat method.*
3. *The next seven methods test all methods related to modifying a boat´s attributes such as speed and health.*

4. *The eight method testIsInvincible checks if the boat properly turns invincible if it has been in contact with an invincibility power up*
5. *The night method tests to see If the boat has been destroyed meaning all the hit points have been taken away from the boat*
6. *The final method tests the ability to reset all boats stats to default.*

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning the boat's main functions work correctly.

Cleanup:

Boat objects created for the test are destroyed.

Test Case Overview

Test Case ID:

TC_Lane

Purpose:

Verify that lane can incorporate and remove objects from the screen.

This test cases involves FR002(Boats), FR006(Power-Ups), FR004(Obstacles).

Test Case Description:

This test evaluates the correct functioning of the class Lane and its checkable methods.

JUnit tests related to this test case will check the correct functioning of the creation of a lane and the ability to create and remove objects in a lane(obstacles,boats,powerups).

Pre-Conditions

Prerequisites:

Class Lane and its methods are properly implemented.

Test Data:

Lane Id, a set of obstacles, a set of powerups, boat type and boat positioning.

Test Steps

Step Description:

1. *SetUp function is called before every test to create a boat, a set of powerUps, a set of obstacles and a lane.*
2. *The first test method tests the creation of a lane ensuring the correct attributes have been passed.*
3. *The second method checks the method testGetBoat to check if a boat is added to the lane correctly*
4. *The third method checks the method testGetBoat Obstacles to check if an obstacle is added to the lane correctly.*
5. *The third method checks the method testGetPowerUps to check if power-ups are added to the lane correctly*

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning the lane´s main functions work correctly.

Cleanup:

Destruction of constructor objects.

Test Case Overview

Test Case ID:

TC_Obstacle

Purpose:

Verify that obstacles spawn/de-spawn correctly and that they move to the desired position.

This test case involves FR004(Obstacles).

Test Case Description:

This test case evaluates the creation of obstacles and their ability to move in the lanes.

Related JUnit tests will check methods relative to the creation of the obstacles and their movement.

Pre-Conditions

Prerequisites:

Class Obstacle and its methods are properly implemented.

Test Data:

Three different objects of type Obstacle all three being of different types.

Test Steps

Step Description:

1. *The method setUp creates the hitboxes of our three obstacles and then creates them to be used in the later tests.*
2. *The second method tests the creation of all three obstacles.*
3. *The third, fourth and fifth methods test the creation obstacles creating all three different types with their correct stats.*

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning Obstacle´s main functions work properly.

Cleanup:

Destruction of constructor objects.

Test Case Overview

Test Case ID:

*TC*_ObstacleVisitorTest

Purpose:

Verify that obstacles of all three types register the act of collision with another object.

This test cases involves FR004(Obstacle).

Test Case Description:

This test case checks the correct function of the method visitObstacle of the interface ObstacleVisitor.

JUnit tests related to this test case will check methods related to the detection of collision of obstacles whether it´s a duck, log or stone.

Pre-Conditions

Prerequisites:

Class Obstacle and its methods are properly implemented.

Test Data:

Three different objects of type Obstacle all three being of different types and an object of type Visitor.

Test Steps

Step Description:

1. The first method setUp tests sets the state of the previously created obstacles to hit and initializes them.
2. The three following methods check if the obstacles detect correctly the fact that a collision has been caused using the getwasHit method.

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning Obstacles are capable of detecting collisions and therefore respond with the correct changes in attributes.

Cleanup:

Destruction of constructor objects.

Test Case Overview

Test Case ID:

TC_CollidableTest

Purpose:

Ensure that collisions between boats (player and AI) and other objects (obstacles and powerUps) in the game environment are detected and handled correctly.

This test cases involves FR002(Boats), FR006(Power-Ups), FR004(Obstacles).

Test Case Description:

This test evaluates the correct functioning of the class CollidableTest and its checkable methods.

JUnit tests related to this test case will check the correct functioning of the collision detection and its corresponding response.

Pre-Conditions

Prerequisites:

Classes Boat, Obstacle, PowerUp and their methods are properly implemented.

Test Data:

The creation of the boat (independent of its type).

Test Steps

Step Description:

1. First, we initialize Boat with the setUp method to use it in the following test.
2. The second function tests to see if the boat changes collision status when function setWasHit is used.
3. The second function tests to see if the boat changes collision status when function getWasHit is used.
4. Finally, the last method ensures that the boat has a non-null rectangular hitbox, so a collision can be caused.

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning Collision´s main functions work correctly.

Cleanup:

Destruction of constructor objects.

Client Specifications:

All the tests belonging to this section are those requirements defined by the clients, they have a lower rank of importance compared to the gameplay because they are not essential for the correct execution of the game.

Test Case Overview

Test Case ID:

TC_Powerups

Purpose:

Verify that powerups are modifying a boat´s attribute correctly.

This test cases involves FR006(Power-ups), FR002(Boats)

Test Case Description:

This test case evaluates the functionality of Speed, HP and Invincibility Power-ups within the game interface.

Related JUnit tests will check the creation and appliance of the three different Power-ups.

Pre-Conditions

Prerequisites:

Class Boat and its methods are properly implemented.

Class HealthBoost and its methods are properly implemented.

Class SpeedBoost and its methods are properly implemented.

Class InvincibilityBoost and its methods are properly implemented.

Test Data:

Height and width of hitbox for the power up creation.

Test Steps

Step Description:

1. *The method setUp is used to create all three power ups for them to be used in the other tests.*
2. *For all three power ups, there is a method Apply that tests the ability of each power up to modify the attributes of a boat.*

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning all three different Power-Up´s main functions work properly.

Cleanup:*Destruction of constructor objects.*

Utilities:

The utilities of the program are those specifications that make the program more complete and give the user a wider range of possibilities when navigating through the system.

Test Case Overview

Test Case ID:

TC_KeyBindings

Purpose:

Verify that the user can customize the controls.

This test cases involves FR018(Controls), FR001(User).

Test Case Description:

This test case checks whether the control related tests can be performed with different control settings.

JUnit tests related to this test case will check the methods that update the variables related to keyboard listeners.

Pre-Conditions

Prerequisites:

Class KeyBindings and its methods are properly implemented.

Test Data:

Object of class KeyBindings

Test Steps

Step Description:

1. *The function setUp is executed in order to create an object of type KeyBindings for the other tests.*
2. *The second function tests to see that there are default key bind values and that they are set as soon as the object KeyBindings is created.*
3. *The third, fourth and fifth methods check the ability to change the value of a specific keybind to another key on the keyboard.*

Post-Conditions

Expected Outcome:

JUnit passes every one of the previous tests meaning KeyBinding´s main functions work properly.

Cleanup:

Destruction of constructor objects.

Section 12 *Extra*: Patterns

In order to carry out the work of implementing the code of any project, it is an essential tool to use all possible methods to make the code pleasant and readable. Patterns are crucial to be able to meet all these requirements. Here are the most common patterns. that we have implemented in our project and what work they take within it

1.Visitor Pattern and Strategy Pattern for movement and collisions

In our project we have multiple dynamic objects that implement different types of movement (random, linear, by input) in order to simplify everything and to ensure that the objects access only the movement that they are able to do, we use the pattern Strategy and Visitor.

In order to carry out all types of movements, we have used a strategy pattern called Movement Strategy through which dynamic objects can access the move method to perform their corresponding movement, so that the objects do not depend on this pattern, both this and the visitor pattern are connected so that the movement depends on the visitors and not the objects themselves.

The different strategies implemented are:

- AI Controlled Strategy for rivals
- Straight Movement Strategy for obstacles
- Random Movement Strategy for obstacles
- Player Controlled Strategy for the boat related to the player

Every move type is totally independent from another.

2. Singleton Pattern for the different screens of the project

Another characteristic pattern is the Singleton, used to give a global view of the General Controller class but accessing its instance.

The General Controller acts as a central point of control for various screens and controllers in the project. By ensuring there's only one instance of General Controller, we guarantee that all parts of the game access and modify the game state consistently. This avoids issues related to having multiple instances modifying the state independently, which can lead to bugs and inconsistent behavior.

3. Observer Pattern for inputs of the user

The observer pattern is a crucial part of the project, it is visible in the Input Manager class.

The Input Manager class is responsible for handling user input events (such as key presses). When using the Observer pattern, it does not need to know the details of the actions to be performed in response to these events. Instead, it simply notifies subscribed listeners. This decouples the input handling logic from the specific actions to be performed, making the code more modular and easier to maintain.

It can notify multiple listeners about an input event. This is useful if multiple parts of the game need to react to the same input event. It also adheres to the Single Responsibility Principle by focusing solely on handling input and notifying listeners, rather than containing the logic for handling specific input responses.