GROUP 1

ÁNGEL NICOLÁS ESCAÑO LÓPEZ	ANGELESC04@UMA.ES
RICARDO JUAN ALLITT LÓPEZ	RICARDOALLITT@UMA.ES
MANUEL BARRIOS MORENO	MANUELBARRIOS@UMA.ES
MUHAMMAD ABDULLAH SULTAN	MASU@UMA.ES
PABLO HORMIGO JIMÉNEZ	PABLOHORJIM@UMA.ES
DIEGO SICRE CORTIZO	0611146788@UMA.ES
ÁNGEL BAYÓN PAZOS	ABAYPAZ@UMA.ES
FRANCISCO JAVIER JORDÁ GARAY	FRANCISCO.JORDA@UMA.ES
JUAN TORRES GÓMEZ	JTORRESGOMEZ26UMA@UMA.ES

GitHub Repository
https://github.com/ProyectoXMA/UMA-ISE24-E1

Index:

We decided ordering the tests by what we felt has higher priority. Firstly, we test the elements essential for a playable game, player, levels, lanes, legs, obstacles and rivals are the essentials for there to be an actual game to play. Frames and resolution are also a priority test because we think a playable game must be at least 30fps and we test the resolution with it. Later all the additions the client requires are tested as our main goal is satisfying all the requirements of our client. Finally, all the menus as utilities and other additions as accessories implemented by us are tested to have a polished final product.

Testing Plan

Project Name:
UMA-ISE24-E1

Authors:
Ricardo Juan Allit 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_Player

Purpose:
Verify that boats have the correct properties and if properties are updated appropriately when adding user inputs. This test cases involves FR016(Boat selection menu), FR018(Controls), FR002(Boats) and FR001(Player).

Test Case Description:
This test case checks whether the selected boat by the player exists and is accurate with the one selected by the boat selection menu and that the player can move perfectly.
JUnit tests related to this test case will check the numbers related to the player are equal to each property of the boat selected and with the test data, in case selection menu is incorrect. After this, the test will ensure the player only controls a unique player boat and that it follows all restrictions provided for the player, for example altering its x and y values appropriately with its speed property.

Pre-Conditions

Prerequisites:
The game is installed and running successfully.
Game is in the selection menu.

Test Data:
Structure with n number of boats with their properties.

Test Steps

Step Description:
1. *Selecting the desired boat from the selection menu.*
2. *Movement of the player is tested.*
3. *Quitting the race.*
4. *Selecting the next desired boat.*
5. *Steps are repeated until all boats are tested.*

Post-Conditions

Expected Outcome:
JUnit test verifies and returns if all boat properties were the same or not inside player, selection menu and test data.
JUnit test verifies player can alter with input uniquely the properties it is allowed to change.

Cleanup:
Close the game.

Notes

This test depends on the navigation, player and boat selection tests.

Test Case Overview

Test Case ID:
TC_Levels

Purpose:
Verify that the user experiences a difference in difficulty through the race.
This test case involves FR009 (Legs), FR003 (Levels), FR014(Leg Duration).

Test Case Description:
This test case evaluates the level of difficulty of each leg which entails.
the level of the opposing boats, quantity of obstacles and the speed at which the obstacles move.
It aims to ensure that players can acknowledge the changes in difficulty between the 3 legs.
Related JUnit tests will check methods relative to the legs.

Pre-Conditions

Prerequisites:
The game is installed and running successfully.
User is in main menu.
User initiates the race by pressing the play button in main menu.

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Register time of the run produced by ai in leg 1.*
2. *Time speed of objects movements in leg 1.*
3. *Register number of obstacles in leg.*
4. *Register time of the run produced by ai in leg 2.*
5. *Time speed of objects movements in leg 2.*
6. *Register number of obstacles in leg.*
7. *Register time of the run produced by ai in leg 3.*
8. *Time speed of objects movements in leg 3.*
9. *Register number of obstacles in leg.*
10. *Compare the results obtained with each other to ensure a visible change in difficulty.*

Post-Conditions

Expected Outcome:
Object movements speed increase, obstacle amount increases and ai leg times decrease as you advance in the legs.

Cleanup:
Not applicable.

Notes

Must be run with all 3 boats to check that all 3 boats can beat these difficulties.
The collision of the ai boats will vary from run to run as they have a level of randomness.

Test Case Overview

Test Case ID:
TC_Lanes

Purpose:
Verify that every player has its own space inside the race to clarify where the user must stay during it.
This test cases involves FR020(Lanes).

Test Case Description:

This test evaluates the player's ability against the division of the entire map into lanes to check all illegal moves that the user may cause.
This ensures that players can stand their ground and that if they do not comply with this rule, we can penalize them for the illegal action they have committed.

Pre-Conditions

Prerequisites:
Game is installed and running.
User has already selected a boat and is inside a race.

Test Data:
Not applicable

Test Steps

Step Description:
1. *Enter a race with the user selected boat.*
2. *Check if the collision between the boat and the division lane is well implemented.*
3. *Check that the lane collision penalty is correctly applied to the player.*

Post-Conditions

Expected Outcome:
In case the player collides with a division line, then the player must die, and a Game Lost screen must pop up to tell the user that the race has ended because of an illegal movement.

Cleanup:
Close the game.

Notes

This test can be performed with any controls and it's just to verify that the lanes between competitors are not crossable.

Test Case Overview

Test Case ID:
TC_Legs

Purpose:
Verify that the user can play all three legs in the race if the previous legs have been completed.
This test case involves FR009 (Legs), FR003 (Levels), FR014(Leg Duration).

Test Case Description:
This test case evaluates the functionality of the 3-leg system inside the level including the change in difficulty and scenery. It aims to ensure the correct transitioning in the race for the player.
Related JUnit tests will check methods relative to the clock, the control listeners and the objects.

Pre-Conditions

Prerequisites:
The game is installed and running successfully.
User is in main menu.
User initiates the race by pressing the button in main menu.

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Complete the first leg in the race.*
2. *Verify the boat moves correctly.*
3. *Check the difficulty has been increased by confirming the change in speed.*
4. *Check the scenery has changed.*

5. *Check the quick menu is still accessible.*
6. *Repeat process for all three legs and repeat several times to ensure correct functioning.*

Post-Conditions

Expected Outcome:
User changes legs without encountering any glitches or errors like not having a change in scenery.
Difficulty is modified.
The level is beatable.

Cleanup:
Not applicable.

Notes

The test must be repeated several times and must be run with the three different boats.
To ensure all three boats can beat the levels and that none of them cause the race to crash.

Test Case Overview

Test Case ID:
TC_Obstacles

Purpose:
Verify that randomly spawned along the lane obstacles work correctly, attributing slowing properties and severe damage to the boat hull. This test cases involves FR011 (Health-bar), FR020 (Lanes), FR001 (Player) and FR009 (Legs).

Test Case Description:
This test case evaluates the functionality of obstacles such as ducks, logs, and stones within the game interface. It aims to ensure the player sees his navigation along the river negatively affected due to the collision with the obstacles' hitbox and to prevent any issue when appearing randomly down the lane.

Pre-Conditions

Prerequisites:
Game is installed and running successfully.
User has chosen a boat and a race scenery.
User is already participating in a race (the difficulty level does not affect)

Test Data:
Not applicable.

Test Steps

Step Description:
1. *The duck obstacle appears randomly down the lane and keeps moving side to side (horizontally) with the purpose of hitting the boat.*
2. *When the player's boat surpasses a duck obstacle's hitbox, the ship loses speed and the remaining health attribute indicated with the health bar decreases.*
3. *The log obstacle appears randomly down the lane and remains static.*
4. *When the player's boat surpasses a log obstacle's hitbox, the ship loses speed and the remaining health attribute indicated with the health bar decreases.*
5. *The stone obstacle appears randomly down the lane and remains static.*
6. *When the player's boat surpasses a stone obstacle's hitbox, the ship loses speed and the remaining health attribute indicated with the health bar decreases.*
7. *If no obstacles' hitbox is surpassed by the boat, the player will not have any disadvantage regarding the rivals.*

Post-Conditions

Expected Outcome:
The ship continues racing against the rivals, seeing the navigation penalized in the event of a collision with any of the obstacles and not in any other case.

Cleanup:
Close the game.

Notes

Each obstacle will cause a different amount of damage regarding his size, being the log the heavier one.
Regarding the slowing down effect, they all act in the same way.

Test Case Overview

Test Case ID:
TC_Rivals

Purpose:
Verify that the AI-controlled rivals (FR013) behave appropriately and have correct properties.

Test Case Description:
This test case ensures that AI-controlled rivals during game execution exhibit a consistent behaviour with their defined properties and characteristics. It involves testing various aspects of rival behaviour such as movement, decision-making, and interaction with the player/obstacles. Additionally, it verifies that the rivals possess the correct attributes and statistics as per the game specifications.

Pre-Conditions

Prerequisites:
Randow boats are assigned to the rivals.
Game is running where the boats of the AI rivals are present.

Test Data:
Structure with 3 number of AI rivals with their properties.

Test Steps

Step Description:
1. *Check if the attributes and statistics of AI rivals match the defined properties in the game specifications.*
2. *Observe the movement patterns of AI rivals during the level and ensure that the rivals stay in their respective lane.*
3. *Evaluate the decision-making process of AI rivals in response to player actions and environmental factors (obstacles and lane limits).*
4. *Repeat steps 1-3 for multiple race sessions to ensure consistency and reliability of AI behaviour.*

Post-Conditions

Expected Outcome:
AI rivals demonstrate realistic and diverse behaviour patterns.
Attributes and statistics of AI rivals align with predefined properties.
JUnit tests verify the accuracy of AI behaviour and properties based on the provided test data.

Cleanup:
Close the game.

Notes

This test relies on the effective implementation of decision-making algorithms for the AI rivals and their corresponding behavior within the game environment. A well-designed AI decision-making system is crucial to provide the players a challenging experience.

Test Case Overview

Test Case ID:
TC_FramesAndResolution

Purpose:
Testing that the game is playable at 30fps for smooth user experience (NFR002) and it's within the boundaries of expected resolution (1920x1080, 1280x720, 1024x768, 800x600) (NFR004).

Test Case Description:
To test the game's performance across different hardware configurations to ensure that it achieves a frame rate of at least 30 frames per second (fps) with the option to run on different screen sizes.

Pre-Conditions

Prerequisites:
Access to multiple computer models and OS with varying specifications to broad our testing ground.

Test Data:
Minimum and maximum specifications of the systems.

Test Steps

Step Description:
1. *Test on minimum requirements.*
2. *Play the game on a computer that meets the minimum hardware requirements to validate our lower bound.*
3. *Test on higher specifications.*
4. *Verify that the frame rate remains stable at 30fps or higher across different hardware configurations.*
5. *Test resolution settings on each system.*
6. *Playing the game on different resolutions may modify the way the game runs so the fps could depend on the resolution based on what configuration the player imposes for the game.*

Post-Conditions

Expected Outcome:
The game consistently achieves a frame rate of 30fps or higher on all tested hardware configurations and screen resolutions.

Cleanup:
Not applicable.

Notes

Regular monitoring and testing of frame rates may be required, after implementing new features like the minigame or adding different components to the screen such as more obstacle types or powerups and if the resolution of the screen changes the way the game behaves in a critical way it may be needed to modify the resolution options available to the player further limiting the environments the game can execute.

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_MiniGame

Purpose:
We're testing that mini-game (FR019) works as intended while following non-functional requirements of Resolution (NFR004), Low Latency Responses (NFR003) since it is another instance inside the main game.

Test Case Description:
Verifying the correct behavior of the mini-game first by launching the mini-game upon player death, checking functionality in response to different player actions such as detecting correct/wrong sequences, and correctly restoring player's HP and respawn point upon successful completion of the mini-game or displaying the Game Over message on failure.

Pre-Conditions

Prerequisites:
The player collides in main game with enough obstacles that boat's HP drops to zero, meaning its destroyed (player dies), then the mini-game is triggered upon death.

Test Data:
Initial player state (position of death and level progress to restore gameplay if successful) and boat's HP.
Rival boats and obstacles states (current position).

Test Steps

Step Description:
1. *Launch mini-game only on player's death within the main game due to boat's HP is zero.*
2. *Verify that the mini-game launches properly (for instance, the commands the player must execute are visible and keystrokes register accordingly).*
3. *While the mini-game is being played, rival boats should not continue the race and only resume movement when the player re-enters the race.*
4. *Test player's input sequences. If enough incorrect keystrokes punish the player by ending current run triggering a Game Over. Otherwise, input of sufficient correct sequences will trigger a win condition thus, ending the mini-game on a successful scenario.*
5. *Player respawns after completing successfully the mini-game, the boat regains the correct amount of HP and the location of boat's revival is exactly the same as the death point.*
6. *When the player is respawned, the game continues as it was, meaning existing obstacles stay on their original track except for the obstacle that triggered the last collision; that one is removed to avoid problems on return.*

Post-Conditions

Expected Outcome:
The mini-game works as intended by initializing on player's death, the gameplay is smooth due to synchronization with NFRs; the response time of player's actions (NFR003) is the expected and that the mini-game displays correctly as stated by NFR004. Success scenario gives the player to a second chance on the current run or on fail scenario finally ends the game displaying a Game Over message.

Cleanup:
Return boat to main game and last collided obstacle is removed.

Test Case Overview

Test Case ID:
TC_LegDuration

Purpose:
Verify the length of the leg is approximately 60 seconds.
This test case involves FR009 (Legs), FR003 (Levels), FR014(Leg Duration).

Test Case Description:
This test case evaluates the duration of each leg of the race.
It aims to ensure the requirements set by the client are met.
Related JUnit tests will check methods relative to clock.

Pre-Conditions

Prerequisites:
The game is installed and running successfully.
User is in main menu.
User initiates the race by pressing the button in main menu.

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Start a timer.*
2. *Complete leg.*
3. *Stop timer.*
4. *repeat the process several times to ensure the nonexistence of any anomalies.*

Post-Conditions

Expected Outcome:

Leg duration is at maximum at the 60 second mark.

Cleanup:
Not applicable.

Notes

The test must be performed with all three boats to ensure none of them are above the threshold.

Test Case Overview

Test Case ID:
TC_Powerups

Purpose:
Verify that user can correctly grabs the collectibles during the race and receive the corresponding advantage regarding the AI-controlled rivals.
This test cases involves FR011 (Health-bar), FR020 (Lanes), FR001 (Player) and FR009 (Legs).

Test Case Description:
This test case evaluates the functionality of Speed, HP and Invincibility Power-ups within the game interface. It aims to ensure the player benefits during his journey along the river in an easy and intuitive way and without encountering any issues when appearing randomly down the lane.

Pre-Conditions

Prerequisites:
Game is installed and running successfully.
User has chosen a boat and a race scenery.
User is already participating in a race (the difficulty level does not affect)

Test Data:
Not applicable.

Test Steps

Step Description:
1. *When the player's boat surpasses the Speed Power-up's hitbox, the ship doubles movement speed.*
2. *When the player's boat surpasses the HP Power-up's hitbox, 25% of the health is automatically added to the health-bar.*
3. *When the player's boat surpasses the Invincibility Power-up's hitbox, the ship becomes oblivious to the obstacles and cannot be negatively affected by these in a time Interval of 10 seconds.*
4. *If no Power-ups hitbox is surpassed by the boat, the player will not obtain any advantage regarding the rivals.*

Post-Conditions

Expected Outcome:
The corresponding booster is correctly applied to the player's boat for each Power-up collected meanwhile, verifying that no advantages are gained if no Power-up is collected maintaining fairness throughout the race.

Cleanup:
Close the game.

Notes

This specific test can be performed with any of the boat models available on the "Boat selection" screen.

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_Navigation

Purpose:
Verify that user can navigate through the menus using the selected controls in expected response time.
This test cases involves FR018 (Controls), FR005 (Main Menu), FR016 (Boat Selection Menu) and NFR003 (Low Latency Responses).

Test Case Description:
This test case evaluates the functionality of controls specifically for menu navigation within the game interface. It aims to ensure that players can traverse through main menu and boat selection screens and select options without encountering any issues or delays.
Related JUnit tests will check methods relative to the control's listeners and event handlers of the keyboard.

Pre-Conditions

Prerequisites:
Game is installed and running successfully.
User is in main menu.
Controls settings are set to default (Arrows).

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Press up and down arrow keys and verify that the different menu options are highlighted.*
2. *Press enter key on "Boat selection" and verify that it is open.*
3. *Press esc key and verify that the main menu is open.*
4. *Press enter key on "Settings" and verify that the settings menu is opened.*
5. *Press esc key and verify that the main menu is open.*
6. *Press enter key on "Tutorial" and verify that the tutorial is opened.*
7. *Press esc key and verify that the main menu is open.*
8. *Press enter key on "Credits" and verify that the credits are opened.*
9. *Press esc key and verify that the main menu is open.*
10. *Repeat navigation process multiple times to check for consistency and reliability.*
11. *Press enter key on "Exit" and verify that the game is closed.*

Post-Conditions

Expected Outcome:
Response time of each operation is less than 30ms.

Cleanup:
Not applicable.

Notes

This specific test can be performed with different control settings after the control's customization test is completed.

Test Case Overview

Test Case ID:
TC_BoatSelection

Purpose:
Verify that user can select a boat from the boat selection menu.
This test cases involves FR018(Controls), FR016(Boat Selection Menu) and NFR003(Low Latency Responses).

Test Case Description:

This test case ensures that the user can select a boat from the boat selection menu. It verifies that the variable that stores the selected boat is updated correctly and that the game can load the selected boat.
JUnit tests will check methods related to the controls listeners and event handlers related to the keyboard and the change of playerBoat variable.

Pre-Conditions

Prerequisites:
The game is installed and running successfully.
User is in boat selection menu.
Controls settings are set to default (Arrows).

Test Data:
playerBoat

Test Steps

Step Description:
1. *Press left and right arrow keys and verify that the different boats are highlighted.*
2. *Press enter key on a boat and verify that it is selected.*
3. *Press enter key on other boats and verify that the selected boat is updated correctly.*

Post-Conditions

Expected Outcome:
playerBoat variable is updated correctly.
Response time of each operation is less than 30ms.

Cleanup:
Set playerBoat to default value.
Close the game.

Notes

This test can be performed with different control settings after the control customization test is completed.

Test Case Overview

Test Case ID:
TC_ControlCustomization

Purpose:
Verify that the user can customize the controls.
This test cases involves FR018(Controls).

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:
The game is installed and running successfully.
User is in settings.
Controls settings are set to default (Arrows).

Test Data:
Key binding variables.

Test Steps

Step Description:
1. *Press "Customize Controls" on the settings screen.*
2. *Desired new key binding for UP, DOWN, LEFT, RIGHT, ENTER and ESC keys are selected whenever it is informed on screen.*
3. *Press "Save" and verify that the new key bindings are saved.*
4. *Perform the above tests with the new key bindings.*

Post-Conditions

Expected Outcome:
Key binding variables are updated correctly.
Tests related to navigation and boat selection are performed successfully with the new key bindings.

Cleanup:
Set key bindings to default values.
Close the game.

Notes

This test depends on the navigation and boat selection tests.

Test Case Overview

Test Case ID:
TC_Pause

Purpose:
Ensure that the pause screen activates and functions correctly.

Test Case Description:
This test case checks that the pause screen activates and functions correctly when the player pauses the game, stopping the race and providing options to resume, exit to main menu, and view basic race statistics.

Pre-Conditions

Prerequisites:
Game has been installed successfully on the user's computer.
The user can start and run the game.
The user has started a race.
The user has default controls.

Test Data:
Not applicable.

Test Steps

Step Description:
1. *During the race, press the designated button pause.*
2. *Verify that the game immediately pauses, halting all race actions.*
3. *Confirm that the pause screen appears with options to resume the game, exit to the main menu.*
4. *Ensure there is a "Resume" button on the pause screen and select it.*
5. *Verify that the game resumes from the exact point it was paused.*
6. *Pause the game again and select the "Exit to Main Menu" option.*
7. *Verify that the game exits the current race and returns to the main menu confirming that no unintended behaviours occur, such as the game freezing or crashing.*
8. *Start a new race to ensure functionality after returning to the main menu.*
9. *During the race, pause and resume the game multiple times using the pause button.*
10. *Verify that each pause and resume operation work correctly without any issues.*

Post-Conditions

Expected Outcome:
The pause screen functionality works as intended, providing a seamless and accurate pausing experience for the player.

Cleanup:
Exit to the main menu and quit the game.

Notes

None.

Accessories:

All the less important requirements are defined as accessories, whose main function is to make the project more efficient and less consuming, as well as to make the player's experience more complete within the video game, their tests are the least significant, as they are secondary agents.

Test Case Overview

Test Case ID:
TC_FileSize

Purpose:
Verifies that the size of the game executables is less than 1GB (NFR005).

Test Case Description:
Check the size of the game during implementation of each class and implementations to calculate that the final product is no more than 1GB is size.

Pre-Conditions

Prerequisites:
Game executables and assets weight (such as size of images used for boats, obstacles, etc. and any music if added).

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Locate the game executables on the system as well as visual assets in source code and see it does not pass the 1GB threshold.*

Post-Conditions

Expected Outcome:
The size of the game is within the desired scope.

Cleanup:
Not applicable.

Notes

Executable size can be affected by factors such as included assets, libraries, and code so further optimization may be needed if this requirement is not met.

Test Case Overview

Test Case ID:
TC_SceneryAndEffects

Purpose:
Verify the proper display of the map scenery and all the assets the game includes.
This test case involves FR008 (Tutorial), FR003 (Levels), FR009 (Legs), FR004 (Obstacles), FR010 (Scenery), FR021 (Visual Effects).

Test case description:
The test ensures that all the sprites are properly displayed during the leg, including obstacles: ducks, logs, and stones; and the scenery: lanes, river, and surroundings. Verifying the scenery changes in every leg according to requirements.
Related Junit tests will check methods relative to the loading screen that appears during the transitions from the different sections the game.

Pre-Conditions

Prerequisites:
Game is installed and running successfully on the different available resolutions.
The user is in the level.
Control settings are set to default.
The assets associated with the obstacles and scenery have already been designed.

Test data:
All the asset`s PNGs must be included in the project´s assets folder.

Test Steps

Steps description:
1. *Complete both the tutorial and the three legs the game includes.*
2. *While doing so load all the available obstacles, verifying they are displayed in the proper resolution, depending on the user´s screen (“1920x1080, 1280x720, 1024x768 or 800x600”).*
3. *Ensure that after loading the scenery, the dragon-themed surrounding fills the screen on the various available resolutions, producing no frame rate drops (frame rate shall not go under 30 fps).*
4. *Collide with all the different obstacles to ensure the PNG fits their hitbox.*
5. *Complete all the legs and the tutorial making sure the scenery is updated every time we move up legs.*

Post-Conditions

Expected outcome:
All the asset’s sprites are properly loaded and displayed, fitting their hitboxes, and not destabilizing the game’s frame rate (staying at constant 30 fps). Scenery is updated every time we complete a leg, and it is displayed properly, fitting the screen in the according resolution.

Cleanup:

Reset the main game’s state.

Notes

This test case assumes the game launches correctly and that the play level section of the game is already implemented and tested individually.

Once this test is concluded, we will assume that the visual effects and scenery of the game is working properly.

Test Case Overview

Test Case ID:
TC_HealthBar

Purpose:
Verify that the item increases and decreases the level shown by the graphical interface regarding the Power-ups or damage caused by collision with obstacles reflects changes in the boat accordingly (FR002). This test cases involves FR004 (Obstacles) and FR006 (Power-ups).

Test Case Description:
This test case evaluates the functionality of the boat’s remaining health bar within the game interface.

Pre-Conditions

Prerequisites:
Game is installed and running successfully.
User has chosen a boat and a race scenery.
User is already participating in a race (the difficulty level does not affect)

Test Data:
Not applicable.

Test Steps

Step Description:

1. *When the race starts, the health bar is completely full.*
2. *When the player's boat surpasses the HP Power-up's hitbox, 25% of the health is automatically added to the health bar.*
3. *When the player's boat collides with an obstacle (Stone, Duck or Log), the amount of ship's remaining health decreases a 25% and its current speed is reduced for a short amount of time as penalty.*
4. *If no Power-ups' hitbox is surpassed or the boat does not crash with any of the randomly spawned obstacles, health bar remains the same.*
5. *If the remaining health becomes 0 or lower, the boat becomes unable to continue racing and the game finishes.*

Post-Conditions

Expected Outcome:
Health indicator is correctly increased/decreased by 25% depending on colliding object and remains unchanged if no Power-up is collected or obstacle is hit. Also, the boat is unable to continue racing, and the game ends when the health-bar reaches 0 or lower.

Cleanup:
Close the game.

Notes

This specific test can be performed with any of the boat models available on the "Boat selection" screen. However, depending on the qualities of each of them, the starting health level and the percentage of health increased/decreased will be different for testing purposes.

Test Case Overview

Test Case ID:
TC_Timer

Purpose:
Ensure that the game timer is functioning correctly, i.e. the time is recorded accurately during active gameplay and pauses when the game is paused (FR012).

Test Case Description:
This test case evaluates that the Timer functionality in active game is working perfectly without any sort of problems. The timer should record the time during the race, so the player has an idea of how long he has been in a particular race. Furthermore, the timer should be paused when the player has paused the game or in case of any other interruptions.

Pre-Conditions

Prerequisites:
Game has been installed successfully on the user's computer.
The user can start, run, and pause the game.
The user has loaded and started a race.

Test Data:
Not applicable.

Test Steps

Step Description:

1. *Verify that the timer starts from 0:00 when the race begins.*
2. *Observe the timer for a specific duration (e.g., 1 minute).*
3. *Ensure that the timer increments correctly (e.g., seconds and minutes are counted correctly).*
4. *During the race, press the pause button to bring up the pause menu and verify that the timer stops incrementing immediately upon pausing.*
5. *Resume the game from the pause menu and confirm that the timer resumes from the exact time it was paused at and continues incrementing correctly.*
6. *Complete the first leg and note the final time displayed on the timer when the leg ends.*
7. *Verify timer pauses after the race completion.*

8. *Verify that the timer resets after leg:*
9. *Verify Timer Does Not Increment Post-Race in any other screen.*
10. *Make sure that the timer functions as intended in all three legs.*

Post-Conditions

Expected Outcome:
The game accurately tracks and displays race time, pausing and resuming correctly in all tested scenarios.

Cleanup:
Exit to the main menu and quit the game.

Notes

None.

Test Case Overview

Test Case ID:
TC_Leaderboard

Purpose:
Verify that the Leaderboard (FR017) accurately displays the positions of the Player (FR001) and Rivals (FR013) during the race.

Test Case Description:
This test case ensures that the leaderboard provide real-time information about the positions of the player and rivals throughout the level. It involves ensuring that the leaderboard dynamically updates as a player or rival overtakes another player or rival.

Pre-Conditions

Prerequisites:
Game is running where the boats of the rivals and the player are present.
Leaderboard visible with an initial order.

Test Data:
Player and AI rival's initial position.

Test Steps

Step Description:
1. *Begin the race session and ensure that the leaderboard is displayed.*
2. *Monitor the leaderboard throughout the race to observe real-time updates of player and rival positions.*
3. *Confirm that the positions and names of the player and rivals are accurately reflected on the leaderboard.*
4. *Ensure that the leaderboard remains legible and in the same place throughout the level.*

Post-Conditions

Expected Outcome:
The leaderboard updates for providing accurate names and positions of both player and AI rivals.
Leaderboard enhances the player understanding of their position in the race relative to rivals.
JUnit tests validate the functionality and reliability of the leaderboard feature.

Cleanup:
Close the game.

Notes

For this test is essential that the updates to the leaderboard accurately correspond to real ones. This provides the player a competitive experience.

Test Case Overview

Test Case ID:
TC_Tutorial

Purpose:
Verify that all users can take a training run to familiarize themselves with the game methodology and learn the basic controls.
This test cases involves FR008(Tutorial).

Test Case Description:
With this test we will check the full functionality of the test run as well as the readiness it gives users to get involved in a real race.

Pre-Conditions

Prerequisites:
Game is installed and running.
User selects the tutorial section in the main menu.
Control settings have been already modified (default as well if no modification is needed).

Test Data:
Not applicable.

Test Steps

Step Description:
1. *Enter the tutorial and check if it starts correctly.*
2. *If the tutorial gives the user the instructions to move, check whether the controls are well specified or not (if user has changed its control inputs).*
3. *verify that all tutorial purposes can be completed without any system failures.*
4. *Finish the tutorial and check if the user has successfully retreated to the main menu.*
5. *Pressing the "Exit" button in the middle of the tutorial to check if the user can end the tutorial whenever he wants.*

Post-Conditions

Expected Outcome:
After the tutorial is completed, the user must see a screen telling him that the tutorial has been successfully completed and can return to the main menu.
If the user decides to close the tutorial, he must go back to the main menu screen.

Cleanup:
Close the game.

Notes

This test may have multiple outcomes because the user has the privilege of exiting the tutorial whenever he wants.

Test Case Overview

Test Case ID:
TC_Credits

Purpose:
Verify that the user can see a final with all the people involved in the creation of the project in the main menu section.
This test cases involves FR022(Credits).

Test Case Description:
This test ensures that the user can see the final credits screen from the main menu.

Pre-Conditions

Prerequisites:
The game is installed and running.
Users are in the main menu section.

Test Data:

Not applicable.

Test Steps

Step Description:
1. *Press the "Credits" button inside the main menu.*
2. *After the execution of the credits, press the return to main menu button.*

Post-Conditions

Expected Outcome:
The user, after pressing the credits button, must see the credits screen and when it finishes, then he must be redirected to the main menu section.

Cleanup:
Close the game.

Notes

This test depends on the main menu section and the controls of the user in the menus.
The user can quit this section whenever he wants by pressing the pause button and selecting the return to main menu option.