|  |
| --- |
| HDSSD |
| Requirements Specification (RS) |
| Cargo Race |

|  |
| --- |
| Sean Trant x13332576  11/13/2016 |

Requirements Specification (RS)

Document Control

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Scope of Activity** | **Prepared** | **Reviewed** | **Approved** |
| 14/10/2005 | 1 | Create | AB | X | X |
| 21/10/2005 | 2 | Update | CD |  |  |

Distribution List

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Version** |
| Eamon Nolan | Lecturer |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Related Documents

|  |  |
| --- | --- |
| **Title** | **Comments** |
| Title of Use Case Model |  |
| Title of Use Case Description |  |

**Table of Contents**

[Requirements Specification (RS) 1](#_Toc467181922)

[1 Introduction 3](#_Toc467181927)

[1.1 Purpose 3](#_Toc467181928)

[1.2 Project Scope 3](#_Toc467181929)

[2 User Requirements Definition 3](#_Toc467181930)

[3 Requirements Specification 4](#_Toc467181931)

[3.1 Functional requirements 4](#_Toc467181932)

[3.1.1 Use Case Diagram 5](#_Toc467181933)

[3.1.2 Use Case 001 – Launch game 5](#_Toc467181934)

[3.1.3 Use Case 002 – Select Level 6](#_Toc467181935)

[3.1.4 Use Case 003 – Choose character 7](#_Toc467181936)

[3.1.5 Use Case 004 – Load cargo 7](#_Toc467181937)

[3.1.6 Use Case 005 – Set trap 8](#_Toc467181938)

[3.1.7 Use Case 006 – Remove cargo 8](#_Toc467181939)

[3.1.8 Use Case 007 – Attack 9](#_Toc467181940)

[3.1.9 Use Case 008 – Remove cargo (AI) 10](#_Toc467181941)

[3.1.10 Use Case 009 – Attack (AI) 11](#_Toc467181942)

[3.2 Non-Functional Requirements 12](#_Toc467181943)

[4 GUI 12](#_Toc467181944)

[5 System Architecture 15](#_Toc467181945)

[6 System Evolution 16](#_Toc467181946)

# Introduction

## Purpose

The purpose of this document is to set out the requirements for the development of a racing game called ***Cargo Race***, its contents will illustrate the complete purpose and declaration of this system.The predetermined users of the software system are people with an avid interest in gaming aged anywhere from 8 years up. In the document the systems constraints, interface and interactions are explained in detail. This document is designed to be proposed to a customer for its approval and to serve as a reference for developing the first version of the system for future development.

## Project Scope

The scope of this project is to develop a 3D, multiplayer racing game made using unity and C# that is aimed at pushing the boundaries of the current racing genre by adding new gameplay elements and twists. The primary objective in this game will be to race other players to the finish line, while keeping as much cargo inside the vehicle as possible, this will contribute to a player’s score, which is not just reliant on who finishes first, but also who has the most valuable cargo remaining. There will be an AI or player controlled “Chaser” that will pursue players through the race and attempt to eliminate them from it. The game will play on a user’s level of greed, as if the chaser gets too close to a player, one can escape by throwing away some of their cargo, which consequently deducts from their score. Also planned is a pre-race or “setup” phase, which will have players load cargo of different value and weight into their vehicle with a 60 second time limit before the chaser arrives and the race begins, during this time, the chaser will have the chance to alter obstacles and set traps on the track.

# User Requirements Definition

In this instance, a 3D racing game is being designed in which users are tasked with loading cargo into their vehicle based on the preferences of weight and value during a 60 second time window which has grown into the “setup phase”. After which, they must attempt to transport these objects from the start to finish of the track as fast as possible while avoiding crashing and being caught by the chaser. As mentioned previously, an objective of this project is to tempt players’ greed, thus cargo being removed from the vehicle will be entirely by player choice, not via physics.

Users will find they need to think about the order in which they stack their cargo. This is because when a player presses the throw cargo key, the cargo will be removed via a first in last out protocol. This again plays on greed, as heavier objects will be worth more, and thus stacking them first will slow a user down for a longer period.

Users must have a warning for when the chaser is near, this can be done using multiple camera angles but ultimately this should be achieved through visual and audio signaling, e.g. music gets louder as the chaser gets closer, a character on the vehicle can visibly begin to panic when the chaser is closing distance. The user will be provided with the ability to play both single-player (AI controlled opponents and chaser) or multi-player (player controlled opponents and chaser).

The objective of the chaser is to eliminate as many players from the race as possible. The chaser will function as such: The longer a player has gone without eliminating one of the racers, the “hungrier” he becomes, when hungry, the chaser will have improved speed, vice versa, the chaser will be slower if he has just recently caught a racer. During the setup phase, a player will have the chance to establish a perimeter of traps and obstacles throughout the race within the same 60 second timeframe the racers get to load cargo.

It is planned to showcase 2 levels, the first will show users a jungle race with makeshift cars being chased by a gorilla. The second, will be underwater in mini submarines with a shark as the chaser, this also must show a form of progression, i.e. the game becoming more difficult or introducing a new challenge, which may come in the form of inverting the movement controls or having to find and collect cargo during the race.

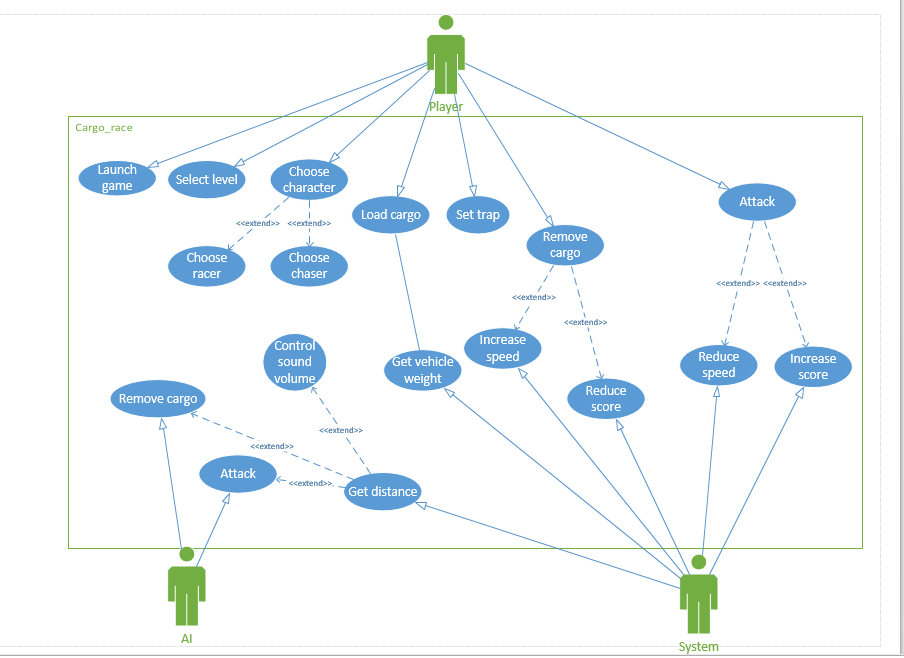
# Requirements Specification

## Functional requirements

**Functional**

* The game must run when launched.
* A player should be able to select a level.
* A player should be able choose between a racer or chaser.
* A player should be able to load their vehicle with cargo based on preferences like value and weight.
* A player’s vehicle speed should be affected by the loaded cargo.
* A player should be able to remove cargo from their vehicle with the press of a key.
* The system should be able to detect and record the distance between a chaser and a player.
* The system should be able to detect and present the speed of the player’s vehicle.
* The second level should unlock after the first level.
* A computer controlled racer should remove cargo from the vehicle when the chaser is within a certain distance.
* A player’s score should be negatively impacted every time cargo is removed from the vehicle.
* Computer controlled chasers should be able to eliminate(attack) players from the race when they are within a certain distance.
* Player controlled chasers should be able to eliminate(attack) players from the race.
* Player controlled chasers should be able to set traps during the setup phase.
* There should be sound to indicate how close a chaser is to a player.
* This sound should get louder the closer the distance is.

### Use Case Diagram



### Use Case 001 – Launch game

**Scope -** This case entails that the game should run after the executable is launched by the user.

**Precondition –** The user has the game installed on their machine.

**Activation –** This use case begins when the user launches the game.

**Normal Path**

1. <Player> opens the executable.
2. The game runs as expected.

**Alternate Path**

1. <Player> opens the executable.
2. An unexpected error occurs.
3. The game does not run as expected.

**Termination –** The user is directed to the menu.

**Post Condition –** The system waits for new entry.

### Use Case 002 – Select Level

**Scope –** This use case entails that a user should be able to choose from the available levels.

**Precondition –** The game is running and the user is on the menu.

**Activation –** This use case begins when the user clicks on the level select button of the main menu.

**Normal Path**

1. <Player> enters the level select screen.
2. <Player> selects the first level.
3. The first level is unlocked.
4. The system continues to the next screen.

**Alternate Path**

1. <Player> enters the level select screen.
2. <Player> selects the second level.
3. The second level is not unlocked.
4. The system prompts the user that the level is not unlocked.

**Alternate Path 2**

1. <Player> enters the level select screen.
2. <Player> selects the second level.
3. The second level is unlocked.
4. The system continues to the next screen.

**Termination –** The system runs the selected level.

**Post Condition –** The system waits for new entry.

### Use Case 003 – Choose character

**Scope –** This use case entails that a user should be able to choose between playing as a racer/chaser.

**Precondition –** The game is running and the user has selected a level.

**Activation –** This use case begins after the user has selected a level.

**Normal Path**

1. <Player> enters the character select screen.
2. <Player> selects a racer.
3. The system directs <Player> to the next screen.

**Alternate Path**

1. <Player> enters the character select screen.
2. <Player> selects a chaser.
3. The system directs <Player> to the next screen.

**Termination –** The system continues to the playing screen.

**Post Condition –** The system waits for new entry.

### Use Case 004 – Load cargo

**Scope –** This use case entails that a racer should be able to choose which cargo to carry in their vehicle during the setup phase.

**Precondition –** The game is running and a level has been loaded.

**Activation –** This use case begins when the setup phase timer begins to countdown.

**Normal Path**

1. <Player> picks up a block of cargo.
2. <Player> puts the cargo in the designated position beside the vehicle.
3. The cargo’s weight and value are stored by the system.
4. The system adjusts the vehicles speed.

**Alternate Path**

1. <Player> picks up a block of cargo.
2. <Player> attempts to pick up another block of cargo.
3. The system notifies <Player> that only one block can be carried at a time.

**Alternate Path 2**

1. <Player> picks up a block of cargo.
2. <Player> attempts to put the cargo in the designated position.
3. The vehicle has no more space.
4. The system notifies <Player> that no more cargo can be loaded into the vehicle.

**Termination –** The timer ends and the system continues to the next screen.

**Post Condition –** The system waits for new entry.

### Use Case 005 – Set trap

**Scope –** This use case entails that a chaser should be able to set traps on the track during the setup phase.

**Precondition –** The game is running and a level has been loaded.

**Activation –** This use case begins when the setup phase timer begins to countdown.

**Normal Path**

1. <Player> picks up a trap.
2. <Player> attempts to set the trap.
3. The location is valid, the system saved the trap position.

**Alternate Path**

1. <Player> attempts to pick up a trap.
2. <Player> is already carrying the maximum number of traps.
3. The system prompts the user that they are carrying too many traps.

**Alternate Path 2**

1. <Player> attempts to set a trap.
2. The location is invalid (e.g. on an uneven surface, beyond the starting line).
3. The system prompts the user that the location for the trap is invalid.

**Termination –** The timer ends and the system continues to the next screen.

**Post condition –** The system waits for new entry.

### Use Case 006 – Remove cargo

**Scope –** This use case entails that a racer should be able to remove cargo from their vehicle during the racing phase.

**Precondition –** The game is running; a level has been loaded and has left the setup phase.

**Activation –** This use case begins when the racing phase does.

**Normal Path**

1. <Player> presses the remove cargo key.
2. The system removes a piece of cargo.

**Alternate Path**

1. <Player> presses the remove cargo key.
2. Less than 3 seconds have occurred since the last removal.
3. The system notifies the user that this ability is not ready yet.

**Alternate Path 2**

1. <Player> presses the remove cargo key.
2. There is no cargo left in the vehicle.
3. The system notifies the user that there is no cargo left.

**Termination –** The action completes and the system reduces the player score and increases speed accordingly.

**Post condition –** The system waits for new entry.

### Use Case 007 – Attack

**Scope –** This use case entails that a chaser should be able to attack a player.

**Precondition –** The game is running; a level has been loaded and has left the setup phase.

**Activation –** This use case begins when the racing phase does.

**Normal Path**

1. <Player> presses the attack key.
2. <Player> hits a racer.
3. The system records the hit and eliminates the player.

**Alternate Path**

1. <Player> presses the attack key.
2. <Player> does not hit anything.

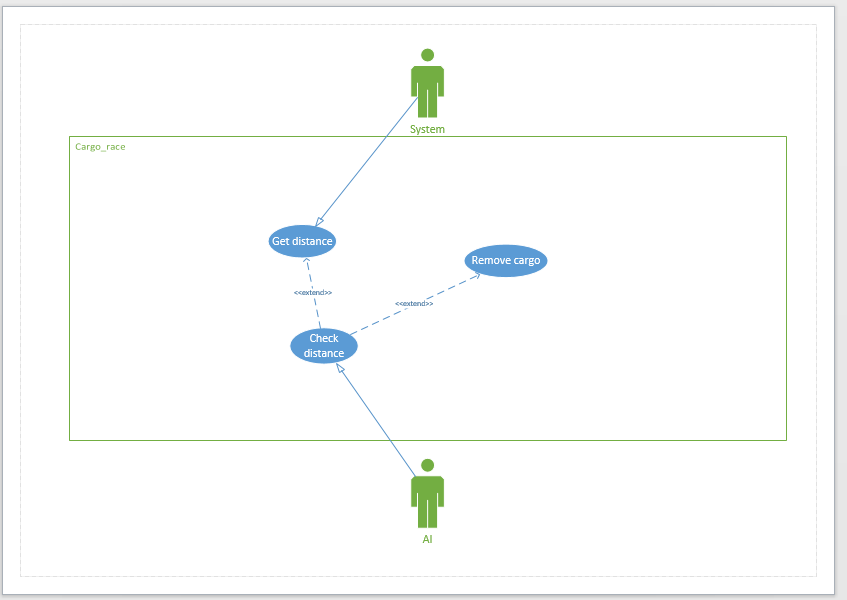
**Alternate Path 2**

1. <Player> presses the attack key.
2. Less than 5 seconds have occurred since the last attack.
3. The system notifies the player that this ability is not ready yet.

**Termination –** The action completes and the system increases the player score and reduces speed accordingly.

**Post condition –** The system waits for new entry.

### Use Case 008 – Remove cargo (AI)



**Scope –** This use case entails that a computer controlled racer should remove cargo when the vehicle and chaser are within a certain distance.

**Precondition –** The game is running; a level has been loaded and has left the setup phase.

**Activation –** This use case begins when the racing phase does.

**Normal Path**

1. <AI> checks if the chaser is >3 (distance) away.
2. The chaser is within 3, <AI> performs the remove cargo action.

**Alternate Path**

1. <AI> checks if the chaser is >3 (distance) away.
2. The chaser is not within 3.
3. <AI> continues to check once per frame.

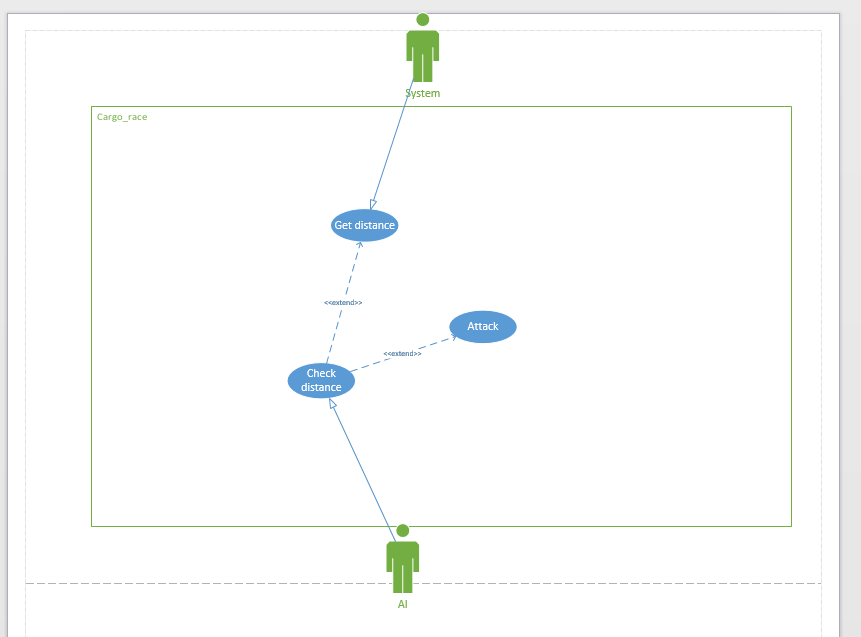
**Alternate Path 2**

1. <AI> checks if the chaser is >3(distance) away.
2. The chaser is within 3, but less than 5 seconds have passed since the last usage.
3. <AI> continues to attempt the remove cargo action once per frame.

**Termination –** The action completes and the system reduces the AI agent’s score and increases speed accordingly.

**Post condition –** The system waits for new entry.

### Use Case 009 – Attack (AI)



**Scope –** This use case entails that a computer controlled chaser should be able to attack a racer when within a certain distance.

**Pre-condition –** The game is running; a level has been loaded and has left the setup phase.

**Activation –** This use case begins when the racing phase does.

**Normal Path**

1. <AI> checks if the racer is >3(distance) away.
2. The racer is within 3, <AI> performs the attack action.
3. There was a hit, the system records such and eliminates the player.

**Alternate Path**

1. <AI> checks if the racer is >3(distance) away.
2. The racer is not within 3, <AI> continues to check once per frame.

**Alternate Path 2**

1. <AI> checks if the racer is >3(distance) away.
2. The racer is within 3, but less than 5 seconds have passed since the attack action.
3. <AI> continues to attempt the attack action once per frame, if the >3 distance remains valid.

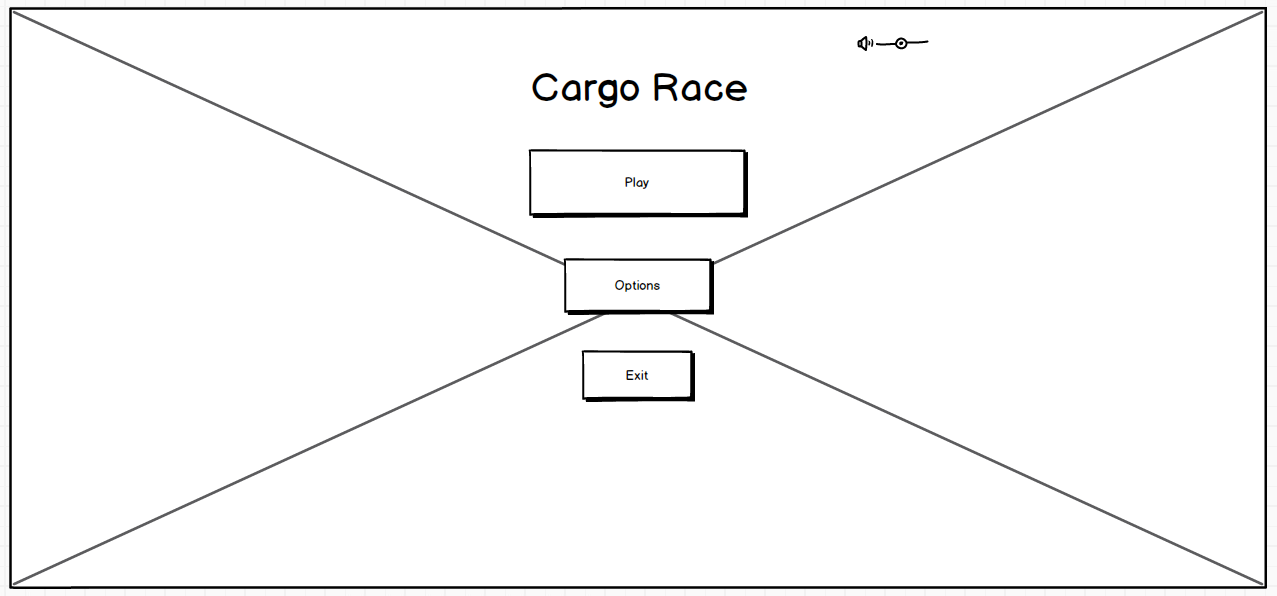
**Termination –** The action completes and the system increases the AI agent’s score and decreases speed accordingly.

**Post-condition –** The system waits for new entry.

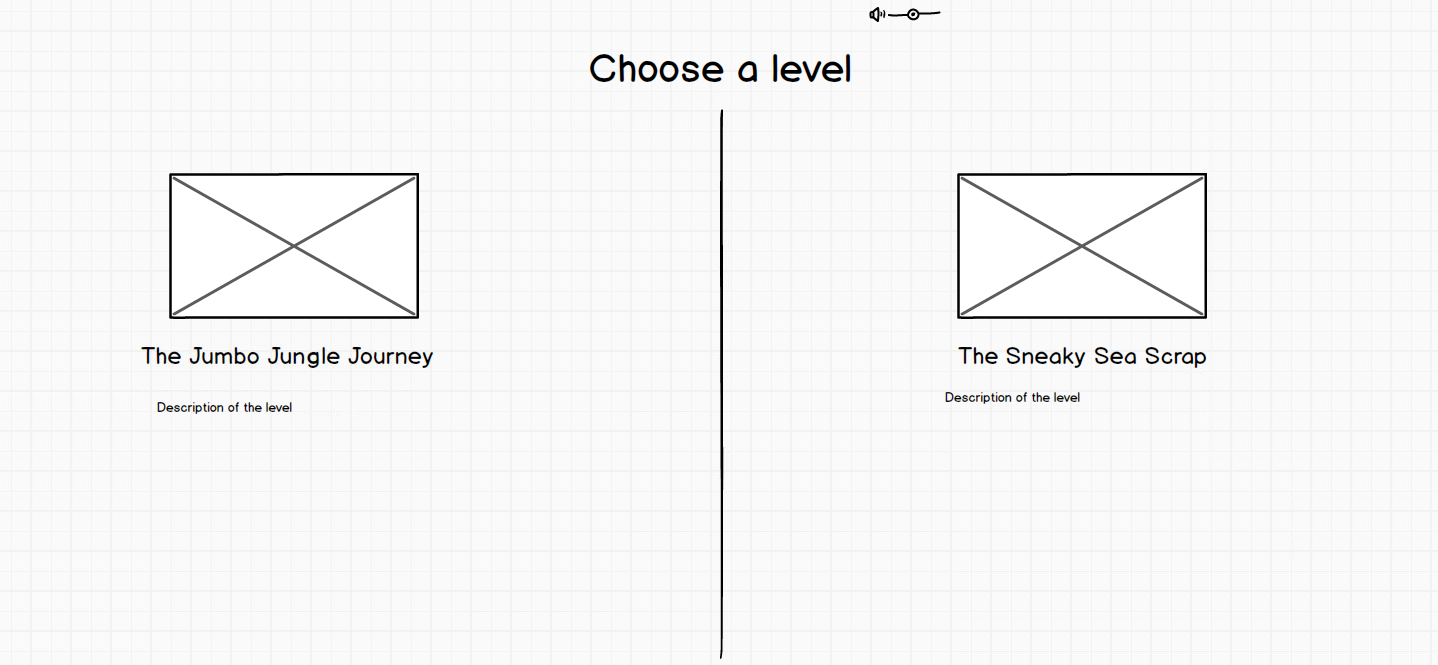
## Non-Functional Requirements

* The game should be implemented and deployed using unity.
* Functionality should be coded using C#.
* The minimum frame rate must be greater than 20.
* The average frame rate must be greater than 30, but ideally should be 60.
* The game must run on windows 7 and windows 10.
* Response time for user input must be less than 0.2 seconds.
* The game’s menu should be easy to navigate.
* All GUIs should be aesthetically pleasing.

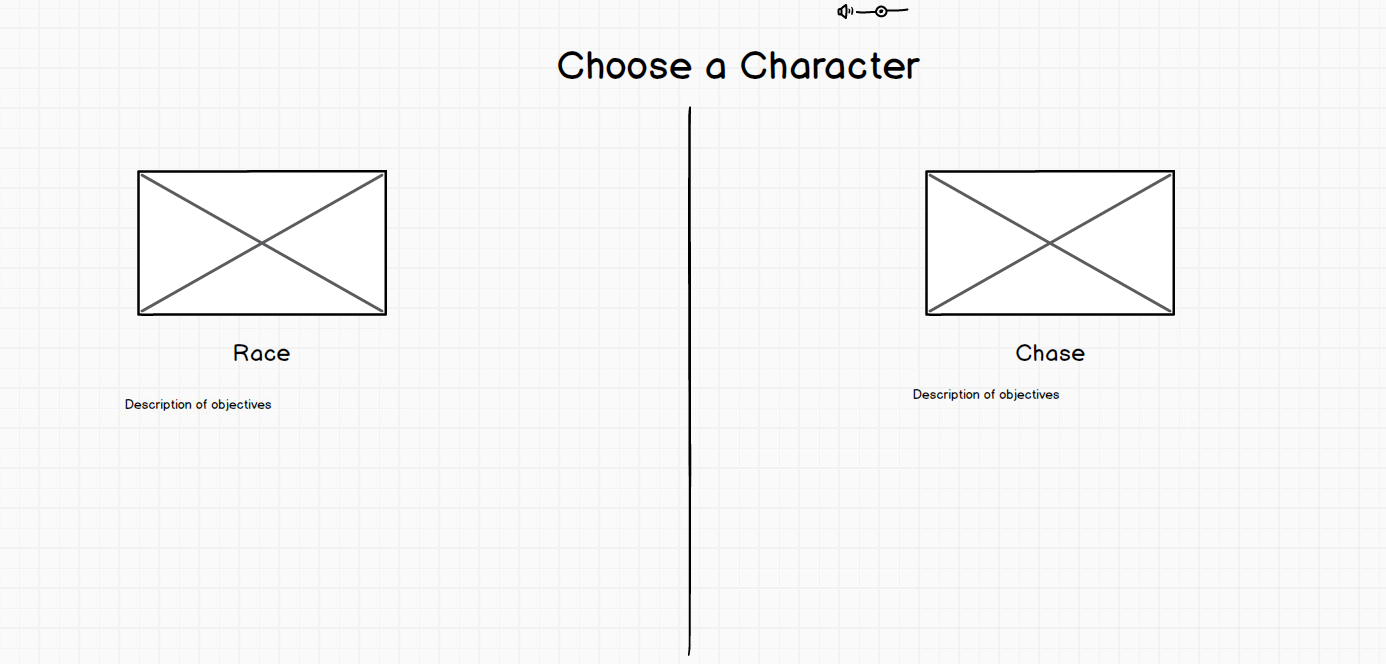
# GUI



This is an early mockup of how the main menu may look. The design is simple as it is expected that users will not want to spend long periods of time on this screen.



This is a mockup that accommodates the functional requirement of choosing a level. The squares above the level names will either be screenshots of the level or short videos of it. Beneath the level names will be a description for users of what they can expect.



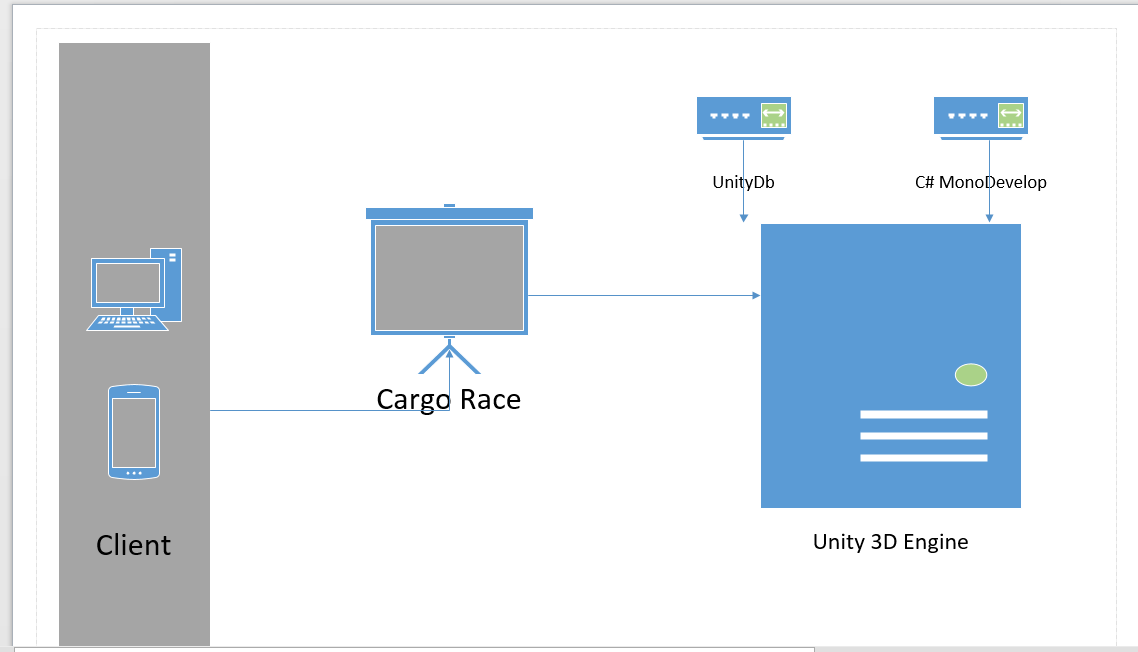
Like the previous screenshot, this mockup deals with the requirement of choosing a character. The squares again will show either pictures of the characters or short videos detailing the actions they can do. Underneath the character title will be a description of a user’s objectives if selecting the character.



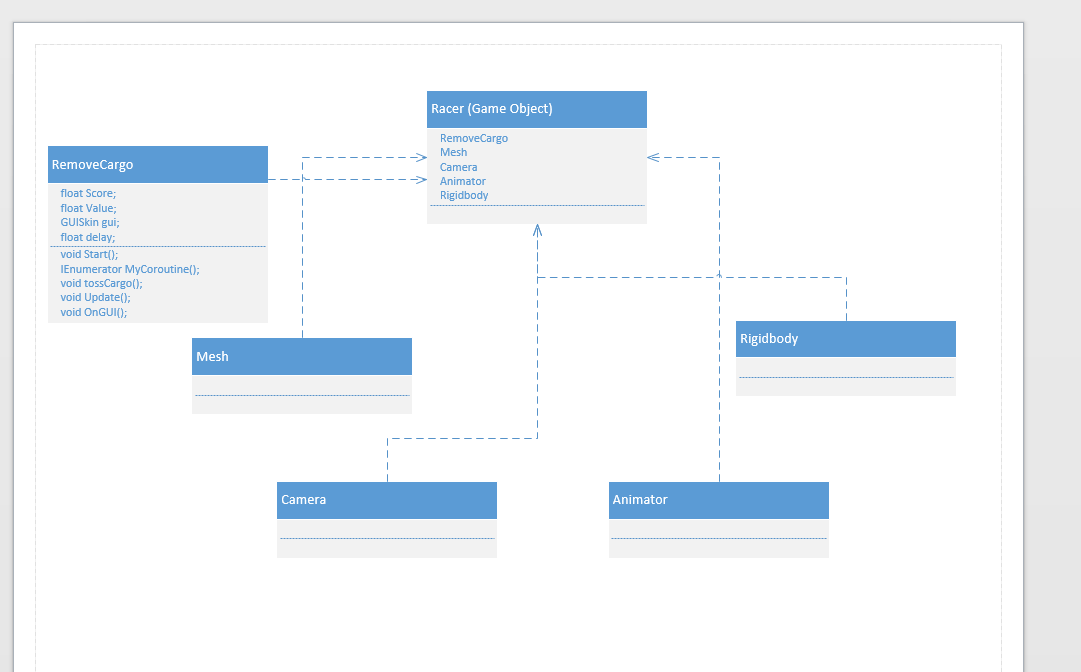


The two pictures above show an early attempt at how the score and cargo removal elements of the UI will be dealt with. When a user presses the remove cargo key, the text in the “Top Cargo: “and “Next Cargo: “sections will change to show which cargo in the vehicle is next, as is demonstrated between the first and second picture. The score will also be reduced per the removed cargo’s value.

# System Architecture



This is the system architecture diagram for Cargo Race showing how it will interact with the client on a physical level. It was chosen because it presents how the game will be run from the Unity 3D engine, which is supported by its own database and the C# MonoDevelop program for scripting purposes.



This diagram details the Racer class at a conceptual level. Not all the details have been worked out, but the Racer will be a “Game Object” within unity and will depend on other attributes (rigidbody, camera, animator etc.) to work as intended.

# System Evolution

The most striking way this system could evolve is through implementing forms of progression or adding new challenges. As per the details of this document, there is no real progression to speak of. Time permitting, it would be fitting to implement progression in the form(s) of:

* Money: If the objective for racing players is to retain as much cargo as possible, there should be a reward for doing so. The system could accommodate this if it is made possible to use players’ score from a race to buy upgrades for their vehicle (improve speed, improve weight capacity, cosmetic changes etc.) and inversely upgrade the chaser.
* Maze mode: The second level, instead of having a setup phase, could present players with multiple paths to the finish line and have them collect cargo on the way there, while avoiding the chaser.
* Last man standing (winner takes all): Instead of having a finish line, the jungle race could loop until there is only 1 racer left. The chaser would have to constantly gain speed, to ensure racers are eliminated. This synergizes well with the money aspect discussed previously.