**REQUIREMENTS SPECIFICATION**

**<ARMAMENT>**

****

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Author | Revision Date | Comments |
| 1.0 | Alex Cohn | Feb. 6, 2019 | initiated |
| 1.1 | Andrew Pitt | Feb. 6, 2019 | System diagrams |
| 1.2 | Alex Cohn | Feb 7, 2019 | Features |
| 1.3 | Andrew Pitt | Feb 7, 2019 | Doc overview, General requirements |
| 1.4 | Jeremy Scott | Feb 8, 2019 | Use cases |
| 1.5 | Keith Bosworth | Feb 8, 2019 | System reqs., minor revisions |
| 1.6 | Ben Mankin | Feb 8, 2019 | Gameplay Diagrams |
| 1.7 | Andrew Pitt | Feb 8, 2019 | System overview, system block diagram |
| 1.8 | Ben Mankin | Feb 15, 2019 | AI Functionality, figure cross-referencing, references |
| 2.0 | Andrew Pitt | April 23, 2019 | System overview revisions |
| 2.1 | Keith Bosworth | April 24, 2019 | Ensure outlined feature set is accurate / up-to-date |
| 2.2 | Keith Bosworth | April 24, 2019 | Add / Update use cases |
| 2.3 | Alex Cohn | April 25, 2019 | AI related updates |
| 2.4 | Alex Cohn | April 28, 2019 | Revisions |
| 2.5 | Keith Bosworth | April 28, 2019 | Finalize |

**Table of Contents**

System Overview 4

System Block Diagram 8

Glossary 9

Document Overview 9

General Requirements 9

Features and Requirements 10

Use Cases 15

References 24

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## System Overview

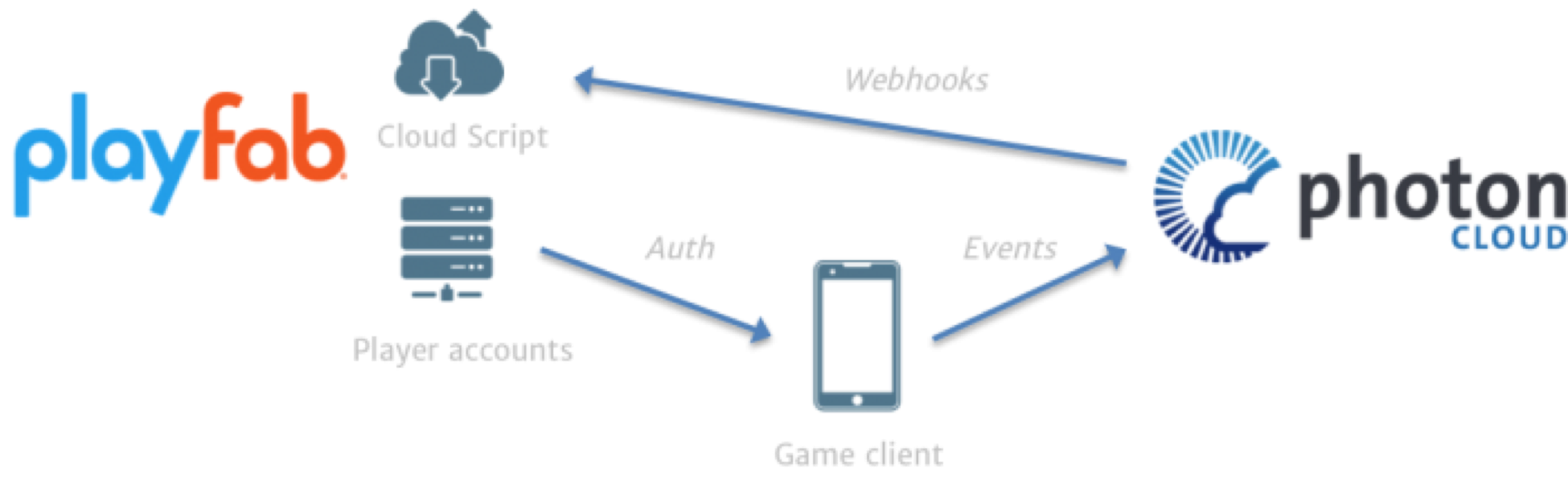
This section provides an overview of the Armament project: an original, networked, multiplayer, team-based video game built for PC, Mac, iOS, and Android platforms. Armament is built with the Unity game engine and API, Photon Networking servers and API, and PlayFab backend database and API.

Gameplay will consist of two stages played in succession: first, an *Armament* stage where players gather weapons and resources, and subsequently, a *Battle* stage where combatants fight for control of the arena using the resources they’ve acquired. Original sounds and art will be created for the project in addition to existing assets.

Armament will be designed and driven with the Unity engine and API, which contains numerous scripts and libraries that provide abstraction for the low-level details of physics rendering, graphics processing, animation, A.I, platform-specific builds, and system analytics.,

Player information is stored in a database provided by PlayFab. Users can register accounts and authenticate from both PC and mobile devices. Once logged in, players will see the Launcher where they can choose to play a game, or they can check the statistics stored in the database, which may be accessed through the leaderboard. If they choose to play, their statistics during that game will be updated to the leaderboard. Players can also add friends through the Launcher, which will allow them to invite those friends to private games. This feature gives players the ability to stay in touch with players they enjoyed playing with.

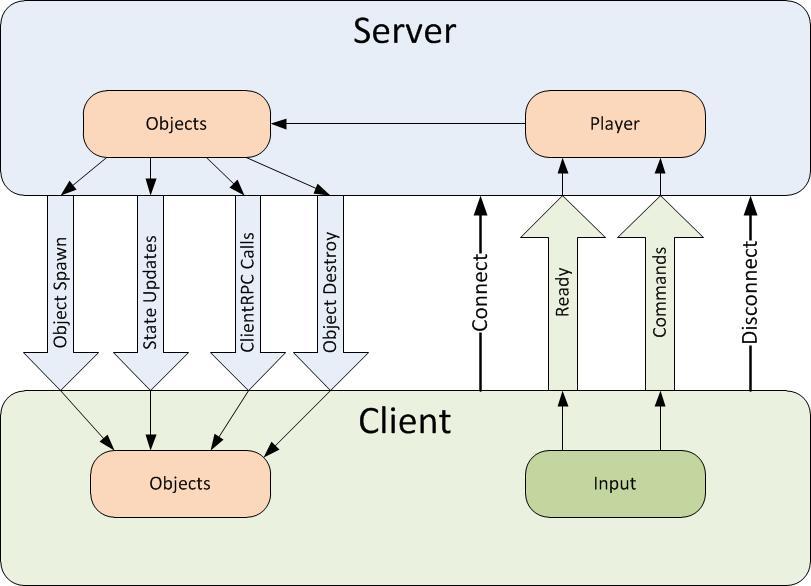
The interaction between Unity clients, the Photon cloud, and PlayFab can be seen below in the abstraction provided by **Figure 1.**

****

**Figure 1.** General overview of Unity client integration with Photon and PlayFab from a mobile perspective (similar for PC).

To play the role of the serving or host device, one client will assume the role of master client, which will synchronize remote clients by broadcasting RPC through Photon servers. A master client is a specialized type of local client that synchronizes remote clients using Photon servers as a means of communication, which sends and receives changes in specific game state based on client input. This implementation is headless in that it only passes RPC calls and choice data through Photon servers, as opposed to processing graphics and physics data within the Photon server space.

In order for Armament to synchronize across every connected device, copies of each object created in the game will exist in memory on both the master and remote clients. The master client device assumes the role of the authority to keep track of changes made to various GameObjects, as well as communicating those changes to the clients. The client/server relationship, RPC calls, data flow, GameObject storage, and state changes can be seen in **figure 2** below:



**Figure 2.** Overall data flow, object storage, and state changes from server (master client) to remote and local clients.

Data will be broadcasted from client to client using RCP and other modes of communication and data synchronization via the Photon server, and through the internet via UDP and TCP. The Photon networking framework provides a robust networking API built specifically for Unity projects in order to meet networking requirements. Clients send input, which is received by listening to various RPC events (for example OnMouseClick() called from within an RPC wrapper to listen to mouse clicks supplied from user input). The Photon Networking API provides abstraction of low-level socket code. **Figure 3** describes the user flow to begin playing online with other players.



**Figure 3.** User flow to begin playing Armament

When joining a game, players must use the in-game menu to connect to a Photon *name server*, which gives them access to a *master server*. Master servers are geographically located around the globe to provide low ping times to all clients, regardless of their location. Master clients will then place the clients in a *master server for matchmaking purposes.* . When a player finds a match they will communicate with other clients via a Game Server that is responsible for hosting the game room they are in.

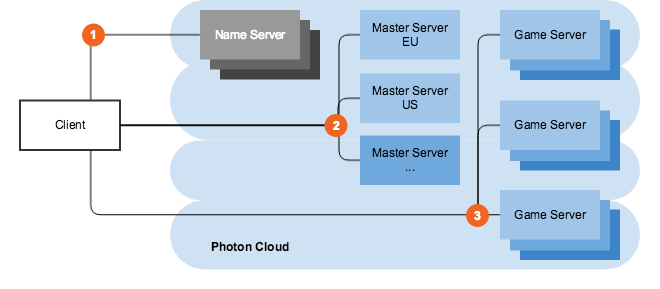
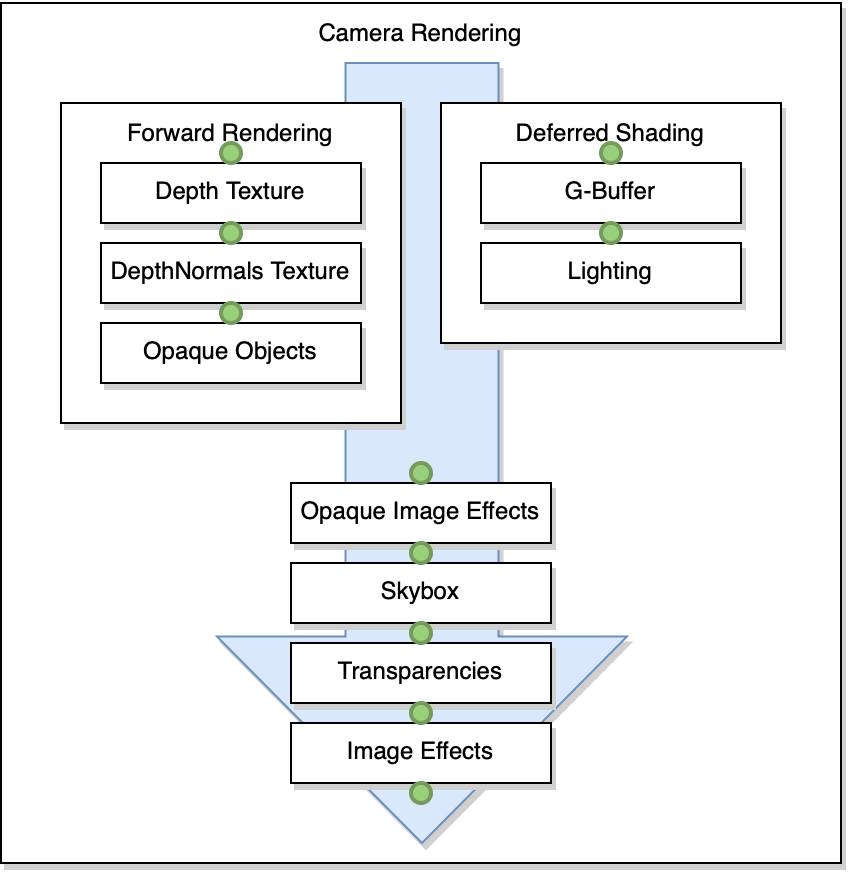


Figure 4.Master server and game server layout..

In the case that the host connection quality terminates or becomes suboptimal, the Photon API offers a host migration service which is called in order to move server identity to the next available device. Sending RPC through the Photon server adds one layer of security for clients in that the IPs of remote clients are managed by the Photon server instead of seen directly by the master.

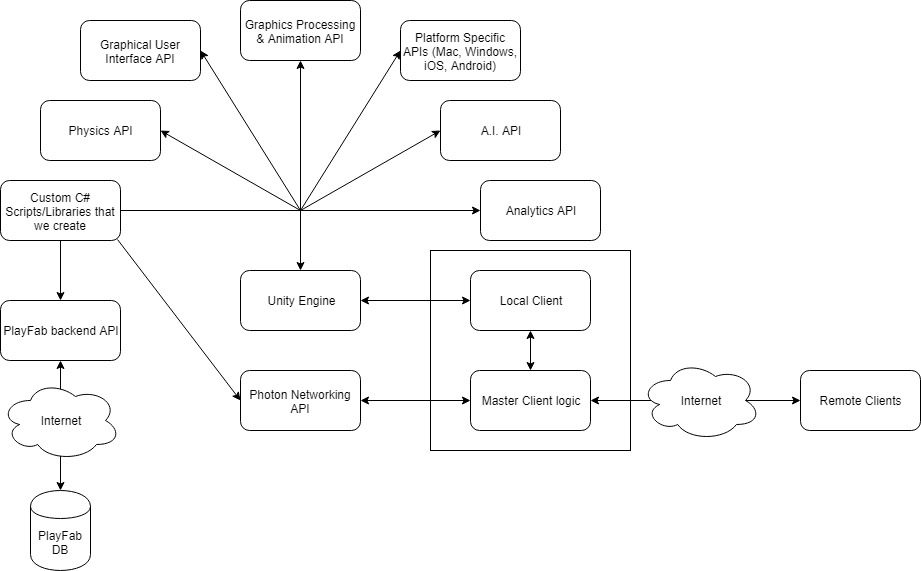
The onus of graphics processing is placed on the client devices, which may flex the master client’s processing capabilities somewhat, but overall is not an impediment considering how picky our master client is in terms of choosing what to broadcast to clients through Photon servers. On the client side, the Unity engine will render a camera placed in the virtual environment, resulting in the data flow represented in figure 5 below. After graphics processing, certain state changes to the GUI will be communicated as attributes via various synchronization mechanisms provided by photon to all clients over the Internet.



**Figure 5**. Camera rendering data flow

During gameplay, players have the option to toggle an A.I. controller. When this controller activated the game automatically takes control over the player avatar’s movement and actions. If the A.I. controller is activated during the *Armament* stage, the A.I. player will target (i.e., go to) known gun spawn points in order to pick up guns. Along the way, the A.I. may recognize that a gun, which it is not currently targeting for pickup and not yet picked up by another player, has come into view. . When it sees a valid gun target, it will run towards it and attempt to pick it up. During the Battle stage, the A.I. will wait for opponents (i.e., players on the other team) to come into view or shoot at it. Either event will trigger the A.I. player to target the opponent. The A.I. player will pursue its targeted opponent even if the player tries to run away. If the A.I. player gets the target in its crosshairs, it will immediately shoot..The A.I. player always calculates shortest path to its destination whether or not its target’s position changes. A high-level representation of the interlocking systems is described below in figure 4.

## System Block Diagram



**Figure 4.** A high-level view of the components in Armament.

## 

## 

## Glossary

* **Master client**: the client that is designated to act as a pseudo server for all other clients. The master client becomes responsible for making decisions and coordinating actions that would typically be the responsibility of a server in a server-client model. Any client that joins a game room can potentially become the master client at some point. By default, the master client is chosen in the order of who entered the game room first.
* **Remote client**: all clients that are not currently the master client.
* Name server: the first server that every client contacts, which provides the list of available regions.
* **Master server**: every region has a completely separate master server for matchmaking.Game server: hosts game rooms
* **Launcher**: the first scene presented to the user upon starting the game. In this scene, a user has the ability to log in to their account, choose their gameplay options, and enter a game room to begin playing the game.

## Document Overview

The requirements document defines features and requirements for Armament. It includes the following sections:

* General requirements
* Features and Requirements
* Use Cases

## General Requirements

Armament will work on both computer and mobile platforms. Multiplayer will be enabled on every platform provided. In addition, players on one platform will be able to join matches and play with players on other platforms. Below is the minimum system requirements that will be targeted throughout development of Armament.

Minimum computer requirements target:

* CPU: Dual-core 2.2GHz (Intel Core i5-5200U or better)
* RAM: 4 GB
* OS: Windows 7 or higher, Linux, macOS High Sierra
* VIDEO CARD: Integrated with at least 1.7GB of shared memory (Intel HD 5500 or better), or dedicated with at least 512MB VRAM (GeForce 240 GT or better)

Minimum Android requirements target:

* OS: Android 5.1.1 or higher
* RAM: 2GB

Mobile-based Apple product compatibility target:

* iPhone X or newer

## Features and Requirements

**Player Setup Features**

* User/Player login
  + Uses PlayFab Database and API
  + Information is used to track game statistics including kills, deaths, and round wins
  + User friendly in-game account registration and login is available via custom menu interface
* Game-play settings
  + Settings are stored locally
    - See Unity documentation (PlayerPrefs) for storage of player preferences
  + User can edit:
    - Player name (default = “Player”)
    - Screen resolution
    - Graphical Fidelity

**Game Initialization Features**

* New Game Creator
  + User chooses a name for the game (a unique name helps other users identify the game)
  + User selects the maximum number of players (multiple of two)

**Game Play Features**

* Two-Stage Round-based Gameplay
  + First stage: Armament
    - Starts at the beginning of the round and continues until the opaque barrier is removed
  + Second stage: Battle
    - Starts when the first stage ends
    - Ends when all players on one team are dead
    - When a team wins, a HUD overlay will notify players of the winning team
* Game Arena
  + Game will have multiple maps
  + Every map contains an opaque barrier that separates players of opposing teams
  + Every map contains a custom Jumbotron object that displays a countdown until the current stage ends, and a label describing the current stage
* Weapons and Items
  + Will be spawned at the beginning of rounds
  + Players can pick up multiple guns during the first stage
* Player Avatars
  + Users will be able to select a team preference prior to a game starting
  + Teams will auto-balance as players join, and this will override player team preference if one team has 2 more players than the opposing team
  + Will be spawned at semi-random spawn points determined by team association at the beginning of round 1
* Every player is represented by and controls an avatar with keyboard and mouse
  + - If user is on mobile device, control buttons will appear on screen
* Players will have the ability to toggle an AI at will to control their avatar
* Two types of avatars to differentiate teams
* If a player death does not result in round completion, then the player will be moved to spectator mode until the next round begins
* Player avatars will have custom animation sets to reflect player actions. These animations will sync across the network
* Player Status Display
  + Custom HUD elements notify the current player of their own status including:
    - Health level
    - Shield level
    - Weapons and Items in inventory
  + During gameplay, each player can open a secondary overlay menu to see a display of the current status of all other players including:
    - Whether a given player is master client
    - The ping of a given player
    - The amount of kills and deaths of a given player
    - Whether or not a given player is currently dead
* AI Player Properties
  + AI players will have fields of view
  + If an object appears in an AI’s field of view, the AI will perform object detection on it and determine the appropriate course of action based on its programmed behavior.
* AI Player Behavior
  + Players are able to manually toggle an AI player to take control of their player avatar
  + The AI will abide by the rules of Armament’s two stage gameplay model
    - During stage 1, the AI player will actively search known spawn locations for guns. If the AI encounters another guns while searching, it will instead pursue that guns first
    - During stage 2, the AI player will pursue players in the opposing team, and attempt to kill them

**Latency/Response-time Requirements**

* In order to prevent noticeable lag, existing games with a ping-time below 150 milliseconds must be available for the user to find. If all suitable games are full, the user can create a new game.
* If the activity on or connectivity with the user’s network is the cause of high ping times, the game system cannot be held responsible for laggy gameplay.
* The GUI update must remain within 30-60 FPS.
* All GameObjects relevant to gameplay will be synced across the network via Photon API

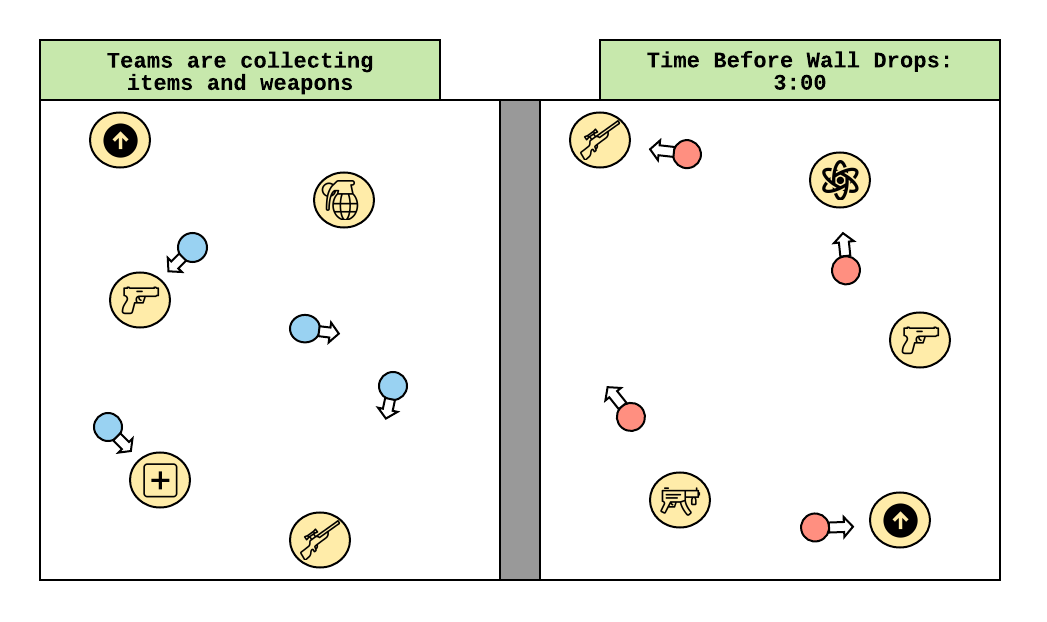
**Downtime Requirements**

* No downtime: if a host goes down, migrate it.

**Player Interaction Requirements**

* Players will be able to establish a friend connection via in-game menu
* Players will be able to interact with a chat box during game matches. The contents of the chat box will indicate the player who sent a particular message. All messages will sync across all active clients in a match.
* Global player statistics / leaderboard for kills, deaths, and match wins will be available to view via in-game menu

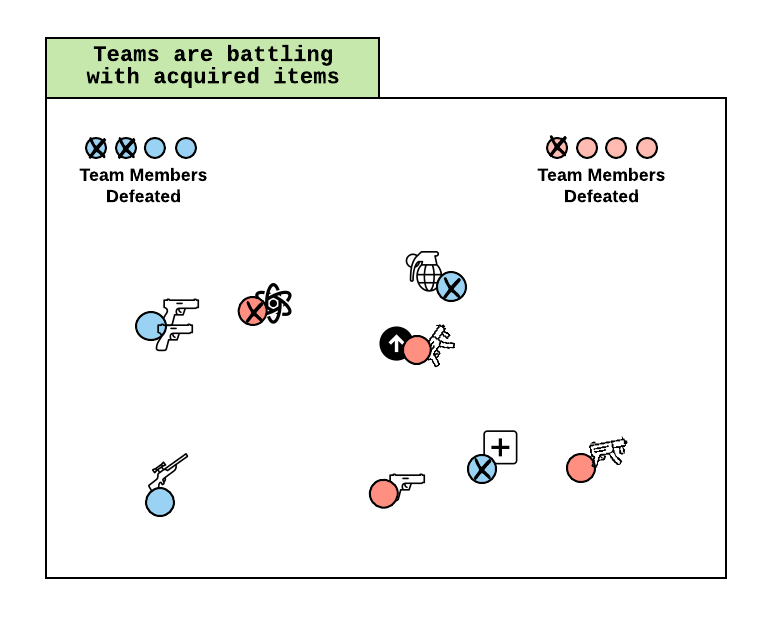
**Gameplay Diagrams (Top-Down View)**

****

**Figure 7**. Armament Stage

*Armament Stage (Figure 7):*

Players are separated by a barrier, and items are scattered throughout the arena. Once the game starts, players can collect the best/as many weapons and items they can before time runs out or until one team decides to drop the barrier early. Weapon/item placement is calculated with an algorithm which uses the value (usefulness) of the item in conjunction with a random factor. As such, better items are more likely to spawn in more remote areas (and thus, have a higher Risk-Of-Acquisition (ROA) number). Players can only hold one item of a category at a time (one machine gun, one melee, etc.).

**

**Figure 8:** Battle Stage

*Battle Stage (Figure 8):*

The barrier is removed from the arena, and the battle begins between teams. All weapons and items that haven’t been acquired yet are removed from game. Players engage in combat using their items and weapons until all members of the opposing team are defeated. Once a team is victorious, the round ends and the players are placed back into a lobby to either play another game or exit.

## Use Cases

|  |  |
| --- | --- |
| **Name** | **UC-1: Random Matchmaking** |
| Summary | The user wants to find a live match along with other users who are also looking for a match. |
| Rationale | In a competitive team game, users often want to quickly be placed into a match without having to find enough other players to fill out a game themselves. Sometimes they may have a few friends to play with as well, but not enough to fill out an entire team. |
| Users | Solo players and small groups of players |
| Preconditions | The game is currently running on a device connected to the internet. |
| Basic Course of Events | 1. The user selects the “Join Random Room” option on the Launcher of the game. 2. The user waits on a temporary screen until enough players have been found to fill out a match. 3. The user is loaded into the game map and the match begins. |
| Postconditions | The user has started a match with other random players. |

|  |  |
| --- | --- |
| **Name** | **UC-2: Private Match** |
| Summary | A group of users choose to be placed into a live match with one another. |
| Rationale | Users with a lot of friends who also play the game will often want to set up a private match with one another. |
| Users | Large group of users |
| Preconditions | All users have the game currently running on a device connected to the internet. |
| Basic Course of Events | 1. One user in the group types a custom room name and clicks “Set Private Room”. 2. Users who are friends with the player can click “Ready Private Room” to grab the set value for the room name. 3. Users can join the private room via the same interface / input that they create a custom match. 4. The users wait in the lobby until the match begins. 5. The host users selects the “Start Match” option to start the match. |
| Postconditions | The group of users has started a match.. |

|  |  |
| --- | --- |
| **Name** | **UC-3: Leaving Game** |
| Summary | The user leaves the game. |
| Rationale | Users may need to leave a match that is already in progress. |
| Users | All users |
| Preconditions | The user is in a match. |
| Basic Course of Events | 1. The user selects the “Leave Game” option. 2. The user is brought out of the active game and back into the Launcher. If they were a host, a new host will be chosen and migrated to in order to avoid downtime. |
| Alternate Paths | 1. At any point after Step 1, the user can exit out of the game options menu and return to the live match. 2. During Step 5, the user can select “No” instead and return to the live match. |
| Postconditions | The user has left the match and is in the Launcher. |

|  |  |
| --- | --- |
| **Name** | **UC-4: AI Toggle** |
| Summary | The user enables the game to control their avatar. |
| Rationale | Users may need to step away from the game, and leaving their players inactive may inhibit gameplay |
| Users | All users |
| Preconditions | The user is in a match. |
| Basic Course of Events | 1. The user presses “T” to toggle the AI player    1. In mobile the player presses the “Toggle AI” button |
| Postconditions | The user has enabled an AI player. |

|  |  |
| --- | --- |
| **Name** | **UC-5: In-game Chat** |
| Summary | The user sends messages to players in the current match |
| Rationale | Users may need to communicate with other users that they are playing with |
| Users | All users |
| Preconditions | The user is in a match. |
| Basic Course of Events | 1. The user presses the “Enter” key to initiate the chat box view.    1. On mobile, the user presses “Open Chat” button 2. The user types a message into the chat box. 3. The user again presses “Enter” to send the message through the chat box to each player in the match. |
| Postconditions | The user has sent a message to all players in the match, and each player in the match can see the message. |

|  |  |
| --- | --- |
| **Name** | **UC-6: View Leaderboards** |
| Summary | The user views leaderboards comparing global stats of players |
| Rationale | Competitive users want to compare their performance with other players |
| Users | All users |
| Preconditions | The user has started the game and logged in |
| Basic Course of Events | 1. The user selects the “Friends/Leaderboard” button 2. The user selects which leaderboard they want to view from the Leaderboard dropdown |
| Postconditions | A dynamic overlay menu displays the selected leaderboard to the user, aggregating global player stats from PlayFab Database |

|  |  |
| --- | --- |
| **Name** | **UC-7: Select Arena** |
| Summary | The user selects an arena to play a match in from a list of multiple available arenas |
| Rationale | Industry standards result in users having an expectation that modern games contain a variety of levels to play in |
| Users | All users |
| Preconditions | The user has started the game, logged in, and navigated to the matchmaking screen |
| Basic Course of Events | 1. The user clicks the “Arena Filter” dropdown 2. The user selects an arenal from the list 3. The user clicks either “Join Random Room” or “Join Specific Room” |
| Postconditions | The user is loaded into a random match that takes place in the selected arena OR they create a new match that takes place in the selected arena |

|  |  |
| --- | --- |
| **Name** | **UC-8: Add Friend** |
| Summary | The user adds a friend to their friend group |
| Rationale | If users connect with other players during gameplay, they want a means of maintaining contact to play with the same players again. Also, real-life friends want a means of coordinating gameplay with each other |
| Users | All users |
| Preconditions | The user has started the game and logged in. |
| Basic Course of Events | 1. The user clicks “Friends/Leaderboards” button. 2. The user enters a friends username in the “Enter a friend’s username” field 3. The user clicks “Add Friend” button |
| Postconditions | The specified username is added to the current user’s friend list |

|  |  |
| --- | --- |
| **Name** | **UC-9: View Player Info in Match** |
| Summary | The user is able to view info of other players during a match |
| Rationale | During a match, users want to keep track of what users are on what team, and how well they may be performing relative to other players. |
| Users | All users |
| Preconditions | The user has started the game, logged in, and joined a match |
| Basic Course of Events | 1. The user presses the “Tab” key to bring up player info overlay    1. On mobile, the user holds the “Player Info” button |
| Postconditions | The player info overlay is displayed, showing the kills, deaths, current status and ping of every player. It also displays which player is master client. |

|  |  |
| --- | --- |
| **Name** | **UC-10: Team Preference** |
| Summary | The user selects a team preference for when they join a match |
| Rationale | The way that matches are designed inherently requires two teams. Often users develop a preference for a given team in multiplayer games (even if just purely aesthetic). Giving the option to users to indicate their preference may increase their enjoyment of the game. |
| Users | All users |
| Preconditions | The user has started the game, logged in, and navigated to the matchmaking screen |
| Basic Course of Events | 1. The user uses the team preference slider to indicate preference between available teams |
| Postconditions | When the player loads into an arena, their team preference is taken into account when assigning a team. They will get into their preferred team unless their preferred team has more players than the other team already. |

## References

* Unity user Manual 2018.3 <https://docs.unity3d.com/Manual/index.html>
* Photon documentation <https://doc.photonengine.com/en-us/pun/v2/getting-started/pun-intro>
* Playfab documentation <https://api.playfab.com/docs/general-getting-started>