Table of Contents:

Installing

You can download the project from GitHub HERE. After downloaded, the Unity project will be under Project folder.
If some errors occurs, it is recommended to delete the folder Photon Unity Networking, and reimport the plugin PUN from Asset Store.

Play

The game **must** be played from the Landing scene. It will initialize all the needed references (Facebook, PlayFab, audio manager, etc), and start the login process, to be able to access all features, and properly connect to Photon and join a room.

Game Flow

Scenes

Landing

Landing screen handles all account management. Logs the player, by connecting him to facebook, and later, PlayFab. Also registers the device for push notification, and initializes the audio manager, which handles music and SFX on/off.

MainMenu

This scene allows the player to check options (to enable/disable music, SFX, and tutorial), go to the profile screen, go to store, or play the game.

Profile

This scene show the player his properties. Name, profile picture, number of wins, kills, level and account experience. Also shows the global leaderboard for the players' killcount.

Store

This scene enables the player to buy a second character to play with, via InApp Purchase.

ModeSelect

In this scene, the player will choose the game mode he will play (1x1,2x2 or 3x3). Based on this option, the player will join a different room. Photon connection is initialized on this scene.

MatchRoom

This is the scene the game will actually play; it contains two "sub scenes": Room, and Match

Room

This scene works as a lobby for the player, where he waits for other players to join. Has a list of characters he can choose to play with (two in total, the second one blocked by InApp Purchase), a list of the players on each team, and a place to chat with the other players.
When the room is full, or everyone on screen hit the "Ready" button, the game will start.
This room has heavy network sync, with messages being sent for every property a player change. More on that later. Also, Photon chat is used only in this scene.

Match

Gameplay scene. Every player is instantiated, all towers and panels are initialized, and, if necessary, bots are created to fill the room.
This is the most complicated scene, network-wise. Syncs everything for players, towers, panels, game states, etc. More on that later.
There's a sidebar, with some upgrades purchases. If the player has enough money, the purchase will be made. Also in this sidebar is the leave game button.

EndGame
Handles end game. Only shows the winner team, and gives the player the option to go to menu, store, or play again.

Tutorial
Has two screens, showing the player how to play. Shown only on the first time the player enters the game, and later if the player enables it on the options menu.

Loading
Helper for scene loading. Has a nice loading animation, empties all unused resources and collects GC, and then loads the next scene, asynchronously.

Prefabs

Avatars
Used on the Room part of MatchRoom. Is the avatar showing all the time in the scene. Also contains player attributes.

Players
Used on Match. Has all player properties and behaviours, shoot and movement system, and network layer.

Bots
Used on Match. Is the same as players, but also has a AI layer on it.

Towers
Used on Match. Has all towers behaviours, and a network layer.

GameLevel
Used on Match. Is a prefab of the game level, containing everything a level should have. If the game will have more than one level in the future, just duplicate and update it.

Bullets
Used on Match. Is the prefab of the player's bullets. On game start, players create a buffer from these prefabs, and use them.

Panels
Used on Match. Reference of the game's panels. Has panel's properties and behaviours, and a network layer.

Persistencies

AccountManager
Persistent to control all account properties getters and setters. Comunicates with Facebook and PlayFab, and holds every property any of these communications may return. Called

anywhere in the game, because account's properties can be changed on gameplay, on store, and will be shown in profile screen.

PhotonManager

Persistent to control connection. Connects to photon and finds a room on ModeSelect, handles all gameplay syncs and events on MatchRoom, and gets destroyed on EndGame. Done this to maintain connection between scenes.

AudioManager

Persistent to handle properties and play music. Has an AudioSource, that keeps playing menu (or gameplay) music in loop, and holds booleans for music enabled, and SFX enabled. Because there's SFX and music in every scene, the manager is persistent.

PlayFab

Login

PlayFab logs in every time the game starts. If new, creates a new user, and logs it in. Used on the script PlayFabManager.

Facebook Link

Gets Facebook's access token from FacebookManager, and links it with a new or existing account. Used on the script PlayFabManager.

Push Notification

PlayFab has a built-in push notification system. After you're able to register the player's device for push notifications, the token must be sent to PlayFab. When this is done, PlayFab will automatically send a new push to the player, whenever the game needs to. Used on the script PushNotificationManager.

News

PlayFab has a built-in news system. The script PlayFabManager gets a defined number of news from the server, and holds it. The NewsManager script asks for a random one from the news list, and shows it on screen.

Store

Catalog

PlayFab holds a catalog for purchasable items. On PlayFabManager script, there's a call to get all items from a catalog. This catalog is shown on the ingame sidebar, and on the item to purchase the second character.

Currency Purchase

The ingame sidebar handles the purchase with virtual currency. PlayFabManager script handles the call for purchase. If the purchase was successful, there's a callback to handle the effects of the purchase.

Real Money Purchase

The store screen handles purchase with real money. InAppPurchase manager handles the call to the stores, and if the purchase is successful, the same script will give PlayFab the purchase token and signature.

Player Data

In this game, PlayFab holds data for the account level, the account exp, the link for the facebook profile picture, and if the player has bought the second character. The script PlayFabManager handles the data save/load for all these parameters.

Player Statistics

In this game, PlayFab holds statistics for players' killcount, and number of wins. The script PlayFabManager handles the data load/update for these parameters.

Inventory

When a player buys an item, the item will be given to his inventory. Because the items are consumed on the moment they buy, this inventory is almost unused. PlayFabManager script handles this item consume, every time a player buys an item.

Photon

Connection Flow

The flow happens in three separate scenes: ModeSelect, MatchRoom, and EndGame. Basically, the player connects to photon, enters lobby, finds a room with the defined parameters, waits for the room to fill, plays the game inside the room, and then exits the room.

Server Connection

MultiplayerRoomsManager handles server connection. It just connects to photon, using the settings found on the Photon settings file.

Lobby

MultiplayerRoomsManager handles lobby. It just stays there for a while, when the user is on ModeSelect scene, waiting for the user to select an option of room size.

Room Join

MultiplayerRoomsManager handles room join. It gets all necessary parameters (player level and room size), and finds a random room within the parameters. If none is found, the system creates a new room, and joins it.

Chat

ChatManager handles chat. When the user joins a room, the script gets the room name, and starts listening to the chat channel with the same name as the room. It handles message send and receive, and populates the respective text fields.

Room Update

MultiplayerRoomsManager handles room update, along with MatchRoomScreen. There are some player properties that need to be synced (player team, player avatar, and ready status). They are all saved as PhotonPlayer properties, and each time some property is updated in a local player, an event is triggered, to update properties. It just gets all players data again, and updates the screen. When all players are ready, starts game.

Game Syncs

Player

Player syncs are handled in PlayerNetworkLayer. Every frame, position, rotation, velocity and actual state are synced. By events, are synced shoot, life, level, respawn, and player attributes

Tower

Tower syncs are handled in TowerNetworkLayer. No properties are synced by frame. By events, are synced tower state, shield state, shoot, and life.

Panel

Panel syncs are handled in PanelNetworkLayer. No properties are synced by frame. By events, is synced only the panel’s owner.

GameController

GameController sync is handled in PlayerNetworkLayer. It just sends the endgame event.

Facebook Login

Facebook login is handled by Facebook Unity SDK, and a class called FacebookManager. It asks for login; if successful, gets user properties (name and facebook picture); when successful, sends user properties and Facebook access token to PlayFab.

PlayerPrefs

PlayerPrefs stores some basic info for the player. Stores ints (converted in runtime to booleans) containing if music, SFX and tutorial are enabled, and if player already connected to facebook.

Input System

The input system was created by us, as general as possible. The class is called InputManager.
There is a main loop, that checks for all touches on screen, and its actual states (can be mocked by mouse, if a boolean is checked). Depending on the touch state, it handles touch start, touch move, and touch end, and sends to delegates, which are responsible for movement and shoot. It also has sprites for two analog sticks, which are shown and relocated on screen by this manager.

Player Attributes

Players have different attributes, which are updated on level up. Character 1 has different attributes from Character 2, and, each level up, these attributes are updated on a constant base, and also there is a method to give a stat bonus, based on player's choice.
This is handled in class PlayerProperties (HeProperties and SheProperties are children of PlayerProperties, and are used to make different bonus options for each character), and is shown in class GameplayUI.

Bots

Bots have a simple system. They are normal players, but have one more class, called PlayerAIManager, which gives a simple behaviour, to look like a player is controlling it.
The base bot behaviour is to walk to a random place in screen, and move the target depending on time and distance from the random endpoint chosen.
On every update, it looks for other elements on the game, and, if they are within a certain range, it will act accordingly. If there is some enemy around, starts attacking him. If there is an unclaimed panel, the AI will try to occupy it. It the tower is around, and has no shield, the AI will attack it. At any time, if the character is shot, it will flee to a random point, but continue to attack.