Geo Cell – Single Player and Multiplayer Game

This document provides detailed instructions on how to edit or extend on this Unity Package.

Contents

Networking Infrastructure

The networking component of this game is using Photon's Network System which is a free plugin for 20 connections (https://www.assetstore.unity3d.com/en/#!/content/1786). After this you might need to sign up to one of their plans but they are very cost effective, see here: https://www.photonengine.com/en-US/PUN/Pricing

This game template doesn't explain how Photon's Networking System works in general. You should read how Photon works to understanding networking in general. It is a simple concept and is covered in this link by Photon and how they suggest you use their infrastructure: http://doc.photonengine.com/en/pun/current/getting-started/pun-intro

IMPORTANT:

You will need to setup a Photon account for your project to use. In Unity you need to go to "Window" -> "Photon Unity Networking" and then click "Setup Project". If you have a Photon Account already you can enter the AppID here, otherwise enter an email address to setup a new account and AppID for your project. That's it, you're now setup and ready to start using their free service.

Currently multiplayer games are based on random selection. Once a user starts a multiplayer game, clients hit the join button and are connected to a random room. For this type of game it doesn't warrant a full lobby but future updates will include a game name for players to enter and join their friend's games instead of pure random connects.

Global Settings

There are a couple of things you will need to change at certain times (like when you build this project for different platforms). These global settings are found in the gameobject called "ConstantObj". This gameobject holds the ConstantObj.cs script which contains the following for you to change:

Control Type

IMPORTANT:

Current you have the options to select DESKTOP, MOBILE or TVOS. Each option sets up the correct input for each platform so DOUBLE CHECK what this setting is before running the game. If you don't set this up correct you will find that your controls won't work. In the Editor you will want to run DESKTOP.

Max Connected Players

This controls the number of players that can join your multiplayer game. Currently this is set to 6 but feel free to change it. Once the max players have joined no more people can join that particular room if it is full and a new room will be created.

Build Version

This is to determine the version of your game so specific versions can play with each other in multiplayer games. No point in having an updated version play with an old version as something will fail.

Game

This is the main gameplay class that manages most of the active game.

The Server controls all the gameplay actors (pickups, enemies and enemy lines currently) and the clients are told when to spawn them and the server updates the clients with positions, rotations and other variables through the PunRPC calls. The clients control their own Player actor and update everyone on their position and a few other variables via the Serialization method *(OnPhotonSerializeView)*. If a client leaves the game will continue but once the Server leaves the clients will be kicked out of their game.

These variables control the spawning of the pickups, enemies and enemy lines for the gane.

Players
Holds the list of spawned players. You don't need to do anything with this one. We have just shown it in the editor so you can easily see what is added to the game.cs script when running the game.

Actors
Holds the list of spawned actors. You don't need to do anything with this one. We have just shown it in the editor so you can easily see what is added to the game.cs script when running the game.

Prefab Player
Holds the player prefab.

Prefab Pickup Shrink
The pickup prefab to spawn on game init

Prefab Pickup Shrink Max
The amount of prefabs instances to spawn into the world on init

Pickup Shrink Spawn Delay Min
The minimum amount of time before enabling a pickup for the player to get

Pickup Shrink Spawn Delay Max
The maximum amount of time before enabling a pickup for the player to get

Prefab Enemy 1
The enemy prefab to spawn on game init

Prefab Enemy 1 Max
The amount of prefab instances to spawn into the world on init

Enemy 1 Spawn Delay Min
The minimum amount of time before enabling an enemy into the world

Enemy 1 Spawn Delay Max
The maximum amount of time before enabling an enemy into the world

Enemy 1 Increase Spawn Speed Count
The number of enemies enabled into the world before decreasing the spawn delay (*Enemy 1 Spawn*

Delay Max) by the decrease spawn speed value (*Enemy 1 Decrease Spawn Speed*). This means the enemies will start to spawn quicker and quicker every X Count.

Enemy 1 Decrease Spawn Speed
The amount of time to decrease the *Enemy 1 Spawn Delay Max* by when the enemy Spawn Count is greater than the Enemy 1 Increase Spawn Speed Count.

Prefab Enemy Line
The enemy line prefab to spawn on game init

Prefab Enemy Line Max
The amount of prefabs instances to spawn into the world on init

Enemy Line Spawn Delay Min
The minimum amount of time before enabling an enemy line object into the world

Enemy Line Spawn Delay Max
The maximum amount of time before enabling an enemy line object into the world

Enemy

An Enemy is made up of several Enemy Levels. Enemy.cs is the parent class that processes an enemy. It holds several EnemyLevel.cs objects in the Enemy Level list. These objects are basically individual enemies that we turn on and off as we wish to change the status of this Enemy.

Each Enemy (Enemy Level actually) has a detection radius *(EnemyDetectionRadius.cs)*. If the Player gets within this radius the Enemy has detected the player and the Enemy will change its status from PASSIVE to AGGRESSIVE.

Each Enemy (Enemy Level actually) has a hit radius *(EnemyHitRadius.cs)*. If the player gets within this radius then the player will be killed.

If the Enemy status is AGGRESSIVE then it will try and line itself up (direction wise) with the player at that very moment to try and hit the player.

If the Enemy goes outside the screen bounds then it will be turned OFF and put back into the pool of enemies to be reused later on. Once it is used again its Current Level will be increased so that the next Enemy Level in the list is used making it a little harder that before.

These variables control the Enemy.cs class

Enemy Levels
Holds all the EnemyLevels so it knows which ones to turn ON and OFF as the game continues. The Current Level is the index as to what EnemyLevel is used at the current time. If the Current Level is 0, then the Enemy Level that shows is the one in position 0 of the Enemy Level list and the rest are turned off.

Current Level
The index for the EnemyLevels

These variables control the EnemyLevel.cs class

Detection Sprite
Sprite that is used on the EnemyDetectionRadius. This gets turned off once an enemy detects the player.

Detection Speed
The speed that will be used to move the Enemy once it detects a player

Speed Min
The min speed that this Enemy will move at when it has not detected the player

Speed Max
The max speed that this Enemy will move at when it has not detected the player

Passive Effects
A container that holds all effects for the PASSIVE status of this Enemy. When this status becomes active, these effects are turned on and any children under this gameobject. The Aggressive Effects are switched off when the Passive Effects are active.

Aggressive Effects
A container that holds all effects for the AGGRESSIVE status of this Enemy. When this status becomes active, these effects are turned on and any children under this gameobject. The Passive Effects are switched off when the Aggressive Effects are active.

Turning Speed
The speed which the Enemy Level determines the Enemy should turn to face it's ideal direction

Bomb
If this Enemy Level has a bomb attached it here. An Enemy Bomb is deployed when an Enemy detection radius is hit. The Enemy Bomb is basically an Enemy that doesn't move. It starts off as a child of an Enemy and when that Enemy has detected the Player it drops (detaches) the Enemy Bomb into the world.

Player

A Player is spawned in the world for each user. If it is a single player game then the Player is the server. If they create a multiplayer game then the player is the server. If they join a multiplayer game then the player is the client.

Each player will be controlled differently depending on what platform is being used. DESKTOP controls the movement of the player by holding down the left mouse button and moving the mouse. MOBILE is holding down 1 finger on the touchscreen and moving that finger around. TVOS is holding down 1 finger on the touchpad (not pressing so it actually clicks, but just lightly touching the touchpad) and moving that finger around to move the player.

The users Player they are controlling has a Player Indicator gameobject that indicates which player they are. All other client Players have this removed on init.

Each Player will constantly grow while the game is active. This is to make it harder for the players to live longer within the game since this gameplay is focused on quick games.

These variables control the Player.cs class

Debug
Currently this enabled God Mode so you can debug easier.

Growth Speed
The speed that the player will scale up

Speed
The speed which the player moves around on DESKTOP

Speed Mobile
The speed which the player moves around on MOBILE

Speed TVOS
The speed which the player moves around on TVOS

Is Player Indicator
This is a container that is enabled only for the local player, all other remote clients will have this disabled so we can tell which player is ours. Customise this however you like.

Enemy Lines

An Enemy is made a line that swipes across the screen and wipes out any enemies it hits. If it hits a Player it will increase the scale of the player making them a bigger target to be detected and hit by the enemy.

These variables control the EnemyLine.cs class

Speed Min

The min speed that this enemy line could start with

Speed Max

The max speed that this enemy line could start with

Flash Time Max

The max time delay the enemy line waits until it moves to wipe out the enemies

On Hit Increase Player By

The value the player will be scaled up by... scaleX * m_onHitIncreasePlayerBy = new scaleX

Pickup

A Pickup is something the Player can collect to help them. In this case there is only one pickup which sets the players scale back to their original scale when the game first stared.

Note: there are 2 TYPES that the pickup can be set to but the “STOP SHRINK” it not implemented yet. It’s not hard to do but it was not implemented due to it not really being that much of a win for the player. If anyone cares for it we can add it in. Select “SHRINK” only for now.

User Interface

We will only be covering the hard stuff in this document. We presume you know how to use the Unity UI system and don't need us to run through that with you. There are also several online resource that explain the Unity UI system if you need help with that.

Each individual menu's parent is a UIPanelBase which is basically a class that holds a reference to a Canvas Group so that with the UIPanelBase.Show() function is called the Canvas Group will fade in smoothly giving a nice effect. It also enables itself and any of its children so certain menus are active at certain times and are disabled at other times - UIPanelBase.Hide(). This stops menus getting in the way of other menus also.

Menu Scene

In the menu scene (under the folder "Scenes") you can find all the main menu editor variables to assign panels, buttons and text to under the **UIMainMenu** gameobject. This gameobject has the **UIMainMenu.cs** script attached to it and the following variables:

These variables control the spawning of the background enemies for the menu.

Prefab Enemy 1
The enemy prefab to spawn on game init. There is only one prefab to the current game that holds all different enemy variations. This is explained later in the document.

Prefab Enemy 1 Max
The amount of prefab instances to spawn into the world on init.

Enemy 1 Spawn Delay Min
The minimum amount of time before enabling an enemy into the world

Enemy 1 Spawn Delay Max
The maximum amount of time before enabling an enemy into the world

Enemy 1 Increase Spawn Speed Count
The number of enemies enabled into the world before decreasing the spawn delay (*Enemy 1 Spawn Delay Max*) by the decrease spawn speed value (*Enemy 1 Decrease Spawn Speed*). This means the enemies will start to spawn quicker and quicker every X Count.

Enemy 1 Decrease Spawn Speed
The amount of time to decrease the *Enemy 1 Spawn Delay Max* by when the enemy Spawn Count is greater than the Enemy 1 Increase Spawn Speed Count.

These variables control the UI items for the menu scene

Panel Menu
Holds the main menu screen UI

Panel MP
Holds the multiplayer screen UI

Btn Single Player
Button to play single player game

Btn Multiplayer
Button to enter multiplayer options

Btn MP Create
Button to create a multiplayer game

Btn MP Join
Button to join a multiplayer game

Btn MP Back
Button to go back from the multiplayer options

Info
Progress text to show status in the multiplayer menu

Game Scene

In the game scene (under the folder "Scenes") you can find all the game editor variables to assign panels, buttons and text to under the **Game** gameobject. This gameobject has the **Game.cs** script attached to it and the following variables:

Note: Game.cs contains a lot more but this section only outlines the UI components of Game.cs. The document is written to allow you to easily find what you are looking for based on the categories and other parts of Game.cs are within this document.

These variables control the UI items for the game scene

Panel Game
Holds the game screens UI

Panel Results
Holds the results screens UI

Btn Start Battle
Button to start the battle

Btn Play Again
Button to play again

Btn Exit
Button to exit the game

Btn Disconnect
Button to disconnect from a multiplayer game

Time To Live
Label to show the BattleTimer

Winner
Label to show text when the game is over

Network Manager
This is a wrapper class that encompasses a few of the PhotoNetwork calls. Take a look in the .cs file as it is all documented and self-explanatory with code and comments.

This script is needed to run a multiplayer game as it handles all the communication between Photon, the server and clients.

World Camera
Holds a reference to an instance of our world camera so we can call it in code easily. Call WorldCamera.Instance.[whatever] to use this.

Support
If you need support on this package or any of our other packages please email support@wireddevelopments.com and we will help you as quickly as possible.

Thanks for your purchase and support on our Unity Assets. If at all possible please leave a review on the Asset Store to help others decide if this package is right for them!

REVIEW LINK: http://u3d.as/ptG