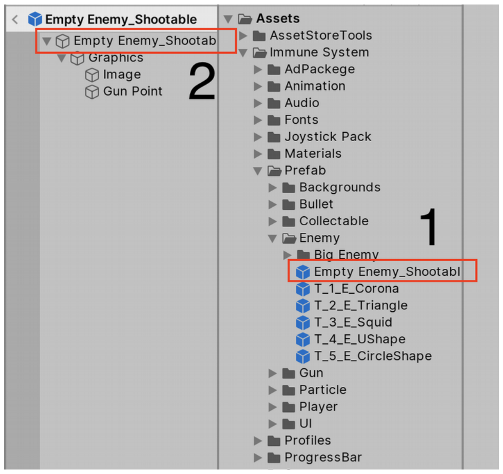
2D Arcade Shoot 'em up Game Template

Introduction: It is a 2d shooting arcade game template. Using this template You can create shoot em up type games without any coding. Every core game mechanics settings are handled by scriptable objects and prefabs. So you can change your game mechanics and graphics from the Unity editor. Everything is very flexible and anyone can customize this template even if you do not have an idea about writing codes. Even if you are not satisfied you can dive into coding and every piece of code is accessible and easily understandable. Not only that, but we are also always open to giving any kind of support.

Enemy: After importing the asset, you can find an enemy prefab in the "Assets/Immune System/Prefab/Enemy" folder. Let's Create a new enemy.



1. First double click on "Empty Enemy_Shootable" prefab(Number 1 in image).
2. All enemy graphics are placed inside Graphics Gameobject. Set enemy graphics on the image component inside the graphics Gameobject. If you want to add multiple images then add that inside the Graphics Object.
3. Place (Positioning) your gunpoint Gameobject properly.
4. Now in Root Gameobject (number 2 in image), set all components values from unity editor. You can easily understand the components and their work by their names and headline.

That's all. You just created a new enemy. If you want to create an enemy without shooting capabilities then remove "EnemyGunController.cs" component from the root object and also delete GunPoint Gameobject.

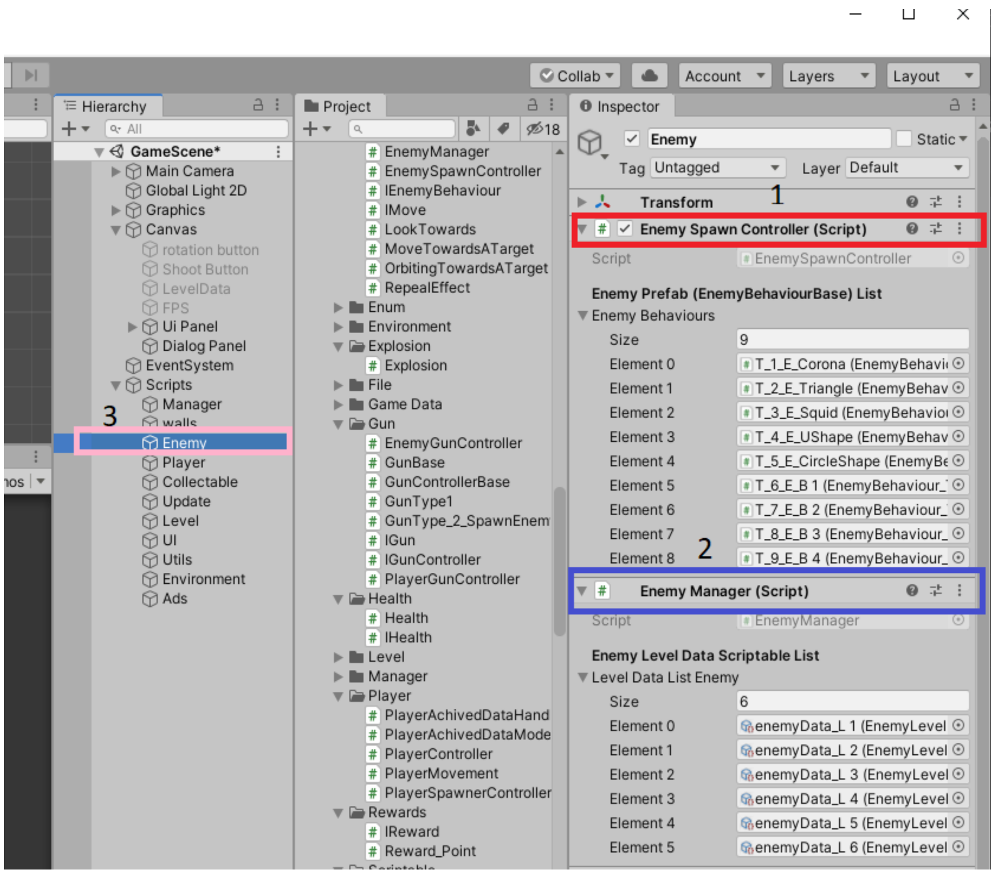
For better understanding please look at enemy prefabs.

** Please check carefully that your enemy prefab has an "enemy" layer and "Enemy" Tag.**

Enemy Spawner: In project Hierarchy (Game Scene), Under Script Gameobject you find "Enemy" (number 3 - pink rectangle) Gameobject. This GameObject has two key script

1. EnemySpawnController: This component is responsible for Enemy spawning. You just need to put all enemy prefabs in the “Enemy Behaviours” List.

** Please make sure that enemy prefab has “EnemyBehaviourBase.cs” script and has the right enemy type.**

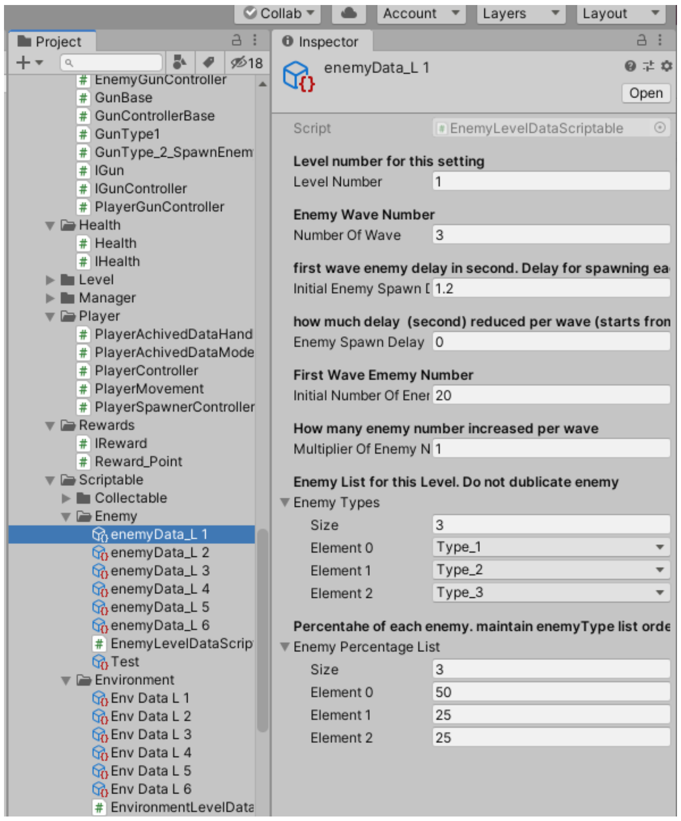


✓ Enemy	Static
Tag Untagged	Layer Default
Transform	
✓ Enemy Spawn Controller (Script)	
Script	EnemySpawnController
Enemy Prefab (EnemyBehaviourBase) List	
Enemy Behaviours	
Size	9
Element 0	T_1_E_Corona (EnemyBehavi
Element 1	T_2_E_Triangle (EnemyBehav
Element 2	T_3_E_Squid (EnemyBehavio
Element 3	T_4_E_UShape (EnemyBehav
Element 4	T_5_E_CircleShape (EnemyBe
Element 5	T_6_E_B 1 (EnemyBehaviour_
Element 6	T_7_E_B 2 (EnemyBehaviour_
Element 7	T_8_E_B 3 (EnemyBehaviour_
Element 8	T_9_E_B 4 (EnemyBehaviour_
Enemy Manager (Script)	
Script	EnemyManager
Enemy Level Data Scriptable List	
Level Data List Enemy	
Size	6
Element 0	enemyData_L 1 (EnemyLevel
Element 1	enemyData_L 2 (EnemyLevel
Element 2	enemyData_L 3 (EnemyLevel
Element 3	enemyData_L 4 (EnemyLevel
Element 4	enemyData_L 5 (EnemyLevel
Element 5	enemyData_L 6 (EnemyLevel

2. EnemyManager: This component is responsible for getting the right enemy level data. This means, on which level loads with what type of enemies and their amount. This data is stored in a scriptable object and this is customizable. We will discuss this scriptable data later.

You have to put all “EnemyLevelDataScriptable” in the “LevelDataList” list.

EnemyLevelDataScriptable: This component holds all enemy data for a specific level.To create this data navigate to “Asset > Create > EnemyLevelDataScriptable ”. Components of this scriptable objects are discussed below.



enemyData_L 1

Script	EnemyLevelDataScriptable

Level number for this setting

Level Number	1

Enemy Wave Number

Number Of Wave	3

first wave enemy delay in second. Delay for spawning ea

Initial Enemy Spawn [	1.2

how much delay (second) reduced per wave (starts fron

Enemy Spawn Delay	0

First Wave Ememy Number

Initial Number Of Ener	20

How many enemy number increased per wave

Multiplier Of Enemy N	1

Enemy List for this Level. Do not dublicate enemy

Enemy Types

Size	3
Element 0	Type_1
Element 1	Type_2
Element 2	Type_3

Percentahe of each enemy. maintain enemyType list orde

Enemy Percentage List

Size	3
Element 0	50
Element 1	25
Element 2	25

1.initialEnemySpawnDelay: first wave enemy delay. Delay for spawning each enemy

2.enemySpawnDelayReduceFactorPerWave: how much delay time reduced per wave.
Example: initialEnemySpawnDelay is 2s and enemySpawnDelayReduceFactorPerWave is 0.5s and wave number is 3. So 1st wave enemy delay is 2s , 2nd wave enemy delay is, 2 - .5 = 1.5s , 3rd wave enemy delay is 1.5 -.5 = 1s

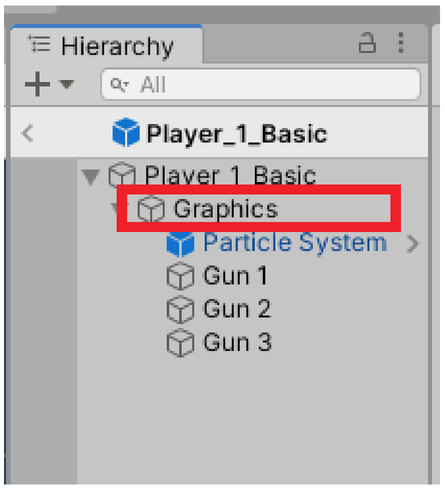
3.initialNumberOfEnemyInAWave: Means how many enemies in the first wave.

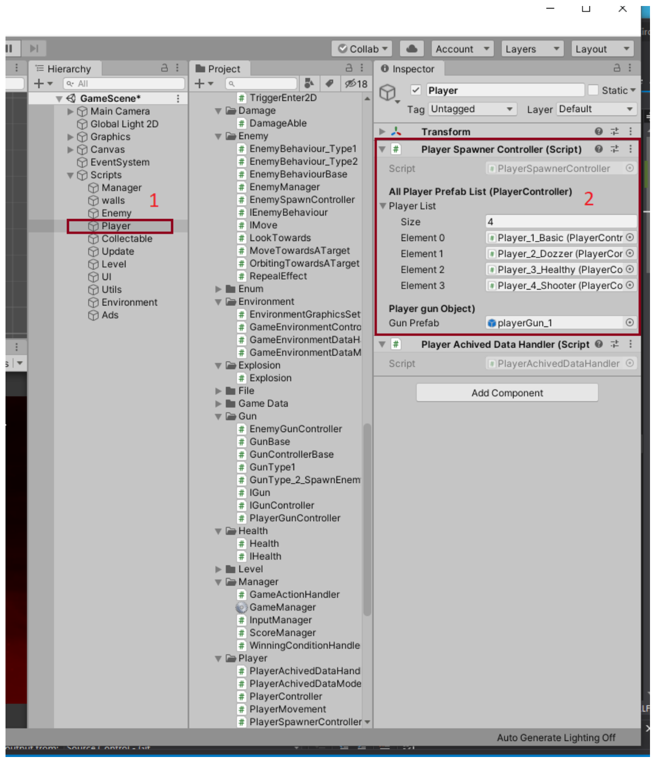
4. multiplierOfEnemyNumberPerWave: means how many enemy number increased per wave
Example: initialNumberOfEnemyInAWave is 30 and multiplierOfEnemyNumberPerWave is 1.5 and wave Number is 4. So in 1st wave enemy number is 30, 2nd wave enemy number will be (30 * 1.5) = 45 and 3rd wave enemy number will be (45 * 1.5) = 67

5.enemyTypes: what type of enemy are present in this level.
Example: In this image this level has Type_1,Type_2 and Type_3 enemy.
** Please do not input one enemy multiple time**

6.enemyPercentageList: Percentage of each enemy(from enemyType list). maintain enemyType list order.made percentage sum exactly 100
Example: In this image this level has 50% Type_1, 25 % Type_2 and 25% Type_3 enemy.
** Please made sure that sum of the Percentage list must be exactly 100**

Player: You can find an existing player prefab in the “Asset/Immune System/Prefab/Player” folder. Currently, four players are available. You can tweak them by changing their values and graphics. Add your own graphics under the Graphics GameObject. Or you can also create a new player easily. For creating a new player just see existing player prefabs. Don't forget to set the player type and set layer. You can set it in the player prefab's root gameObject.





Player
Static
Tag Untagged
Layer Default
Transform
Player Spawner Controller (Script)
Script PlayerSpawnerController
All Player Prefab List (PlayerController)
Player List
Size 4
Element 0 Player_1_Basic (PlayerContr
Element 1 Player_2_Dozzer (PlayerCor
Element 2 Player_3_Healthy (PlayerCo
Element 3 Player_4_Shooter (PlayerCo
Player gun Object)
Gun Prefab playerGun_1
Player Achived Data Handler (Script
Script PlayerAchivedDataHandler

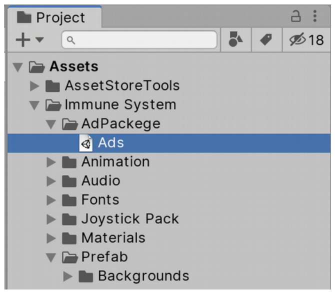
Player Spawn:

PlayerSpawnController.cs script is responsible for spawning players. For this first need to add all player prefabs (with PlayerControlle.cs script) under _playerList variable. You can simply drag and drop that player prefabs from the editor. Which player will spawn depends on game data (Which player currently user set). The default player is "Type_1_Base".

ADS: We implement Google AdMob ads in this project.As google changes their admob unity plugin "GoogleMobileAdsUnityAdsMediation"frequently, so we do not add that plugin in our existing plugin directly. But we have a plugin which has all codes for showing ads easily. For implementing Ads please follow the following instructions:

1. First Download the latest “GoogleMobileAdsUnityAdsMediation” plugin from here.

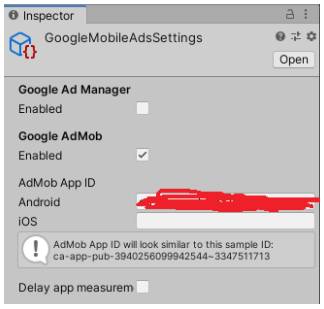
2. Now Create a project on Google Admob. Please follow the instructions from this link.



```
Project
Assets
  AssetStoreTools
  Immune System
    AdPackege
      Ads
    Animation
    Audio
    Fonts
    Joystick Pack
    Materials
    Prefab
      Backgrounds
```

3. Now double click “Ads” plugin from “Assets/Adpackage/Ads”. It will import some scripts for showing ads.

4. Now go GoogleMobileAdSetting by navigating Assets->Google Mobile Ads -> Setting . Now enable Google Admob and input your appId.



Inspector
GoogleMobileAdsSettings

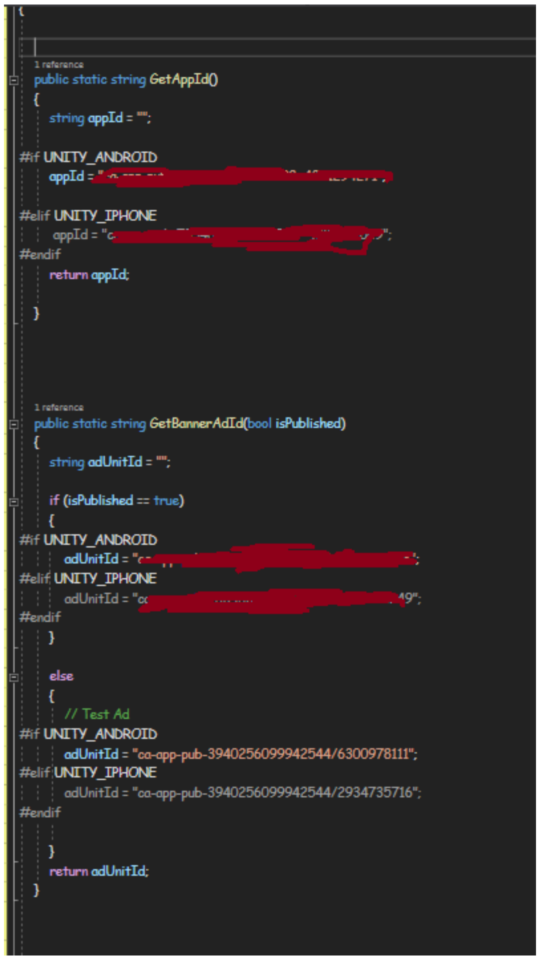
Google Ad Manager
Enabled ☐

Google AdMob
Enabled ✓

AdMob App ID
Android [redacted]
iOS

AdMob App ID will look similar to this sample ID:
ca-app-pub-3940256099942544~3347511713

Delay app measurem ☐



```
{


    1 reference
    public static string GetAppId()
    {
        string appId = "";

#if UNITY_ANDROID
        appId = [redacted]

#elif UNITY_IPHONE
         appId = "c[redacted]";
#endif
        return appId;

    }



    1 reference
    public static string GetBannerAdId(bool isPublished)
    {
        string adUnitId = "";

        if (isPublished == true)
        {
#if UNITY_ANDROID
            adUnitId = "c[redacted]";
#elif UNITY_IPHONE
            adUnitId = "ca[redacted]49";
#endif
        }

        else
        {
            // Test Ad
#if UNITY_ANDROID
            adUnitId = "ca-app-pub-3940256099942544/6300978111";
#elif UNITY_IPHONE
            adUnitId = "ca-app-pub-3940256099942544/2934735716";
#endif

        }
        return adUnitId;
    }
```

5. Open AdUtility.cs Script from “Asset/Immune System/Script/Ads/AdUtility.cs” and set add ids and app id. In this Script please insert appId in “GetAppId()” function and set Banner Ads id in “GetBannerAdId()” function. Please do not edit the test ad part (Test ads are for testing the implementation of your ads.). Only edit the red highlighted part. This Script also has “GetInterstitialAdId()” and “GetRewardAdId()” functions. Please set **Interstitial** and **Reward** ad ids properly on those functions also.

6. Now Drag the “Ads” prefab from “Assets/ImmuneSystem/Prefab/Ads.prefab” and drop it in the project hierarchy.

7. In Admanager (inside Ads prefab) set “isPubish” accordingly.
    a. True - If you build your game for release. It will show real ads.

b. False - if you want to test your game with test ads.

Now everything is set for show ads. Admanager.cs script is responsible for showing ads. Admanager.cs is a singleton class. write “AdManager.instance.ShowRewardAd()” for showing Reward ads and write “AdManager.instance.ShowInterstitialAd()” for showing InterstitialAd.Banner ad showing is controlled inside AdManager.cs script.Currently when game is not running (idle or pause state) then we load banner ad.You can easily change the implementation from AdManager.cs script`s OnGameStateChange function.

** Please uncomment ads code from StorePanel.cs script.**