ASSET DOCUMENTATION

==========

Asset Name: **SLIDING SANTA**

Publisher: **SGLIB GAMES**

Email: **sglib.games@gmail.com**

Documentation Version: **1.0**

==========

Thank you for purchasing our asset! We always try to provide the best service as we can! If you have any questions or suggestions, please email us!

1 INTRODUCTION

Help Mr. Santa slide through a slippery, zig-zag, snowy path amid a beautiful pine forest to collect presents for all the children this Christmas! Tap and hold to turn left, release to turn right. Avoid the obstacles and collect the gifts. It couldn't be simpler! Seriously, **Sliding Santa** is the perfect game for this Christmas season!

Sliding Santa is ready for release out-of-the-box. Everything just works. It is flexible, customizable and extensible. It has built-in all the standard features of a mobile casual game: advertising, leaderboards, achievements, in-app purchases, social sharing, plenty of unlockable characters and a cool, addictive endless gameplay! Here's a more detailed list of some notable features:

- Endless gameplay with the world generated randomly every round, so each one is a new game!
- Advertising options: banner, interstitial and rewarded ads from **AdMob**, **Chartboost** and **UnityAds**.
- Able to use different ad networks for different ad types (e.g. use AdMob for banner ads, Chartboost for interstitials and UnityAds for rewarded videos)
- Able to have different ad network configurations between iOS and Android (e.g. use AdMob for interstitials on Android and use UnityAds for interstitials on iOS)
- In-app purchase: Remove Ads button and a storefront to buy gifts.

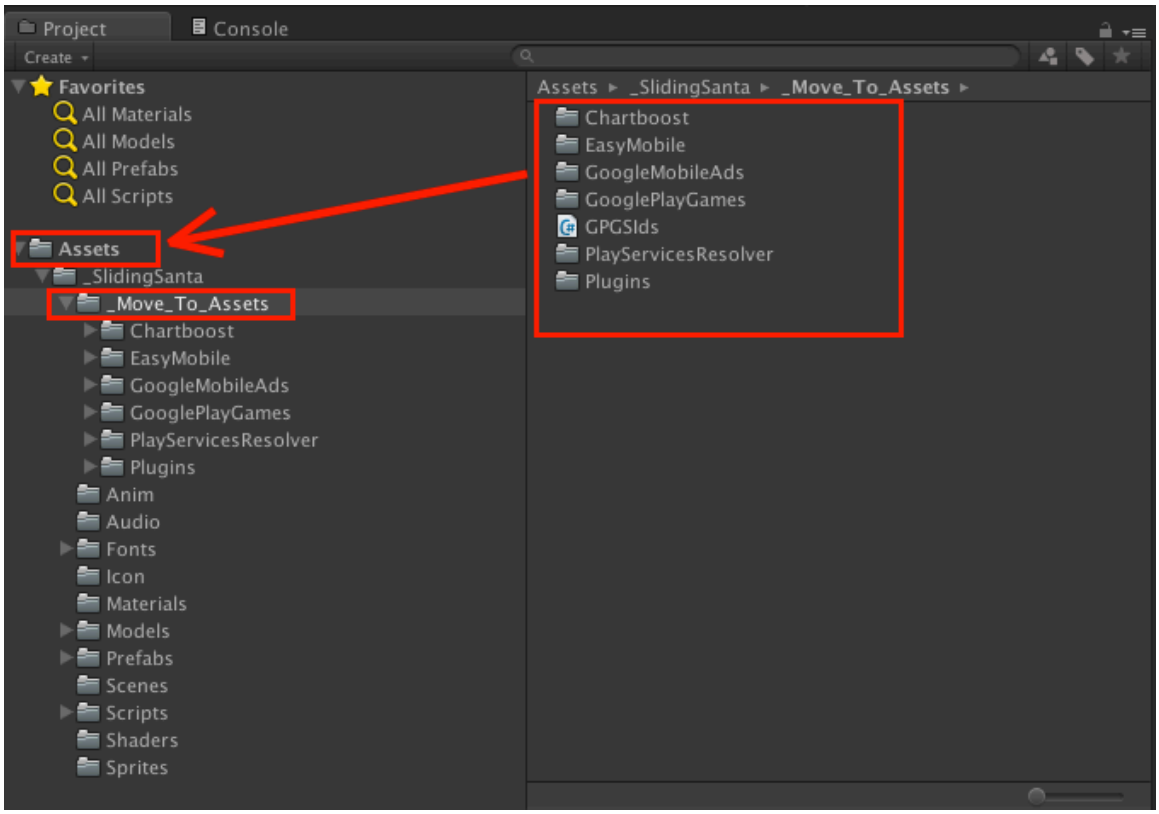
- Integrated with **Game Center** (iOS) and **Google Play Game Services** (Android) for leaderboards and achievements.
- Native screenshot sharing to social networks.
- Daily reward system.
- Dozens of cutes, unlockable characters and vehicles; adding more is a matter of drag-and-drop.
- All assets (graphics, models, sounds and fonts) are free to use for commercial games.

2 GETTING STARTED

Please follow all steps in this section before proceeding.

2.1 Reorganize folders

Asset Store submission guidelines require us to group our asset into one single folder, but some plugins used in this template need to be placed right under Assets folder to function properly, e.g. Google Play Game Services. Therefore, the first thing you need to do after importing this asset is to move everything under the folder **_SlidingSanta/_Move_To_Assets** back to folder **Assets**.

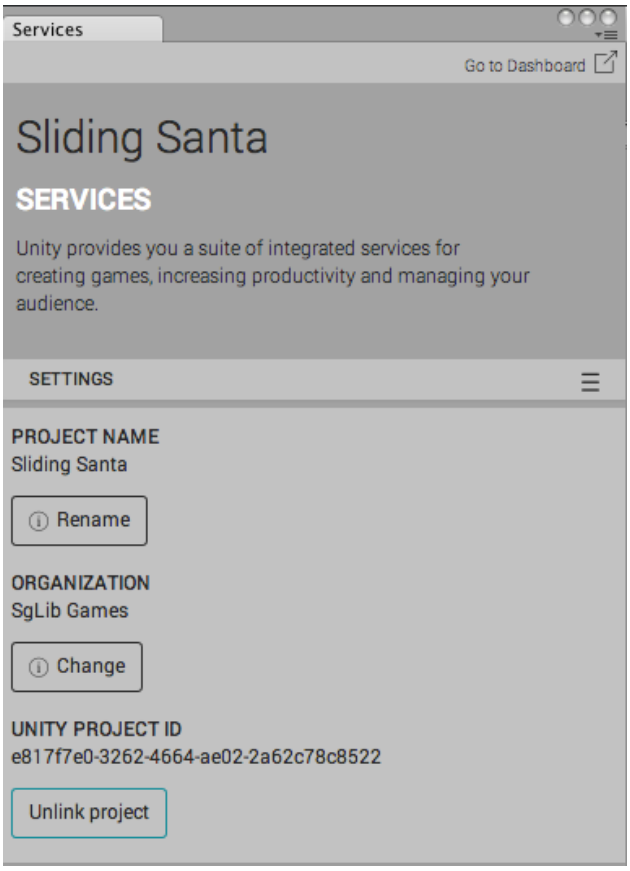


2.2 Link the game to your Unity project

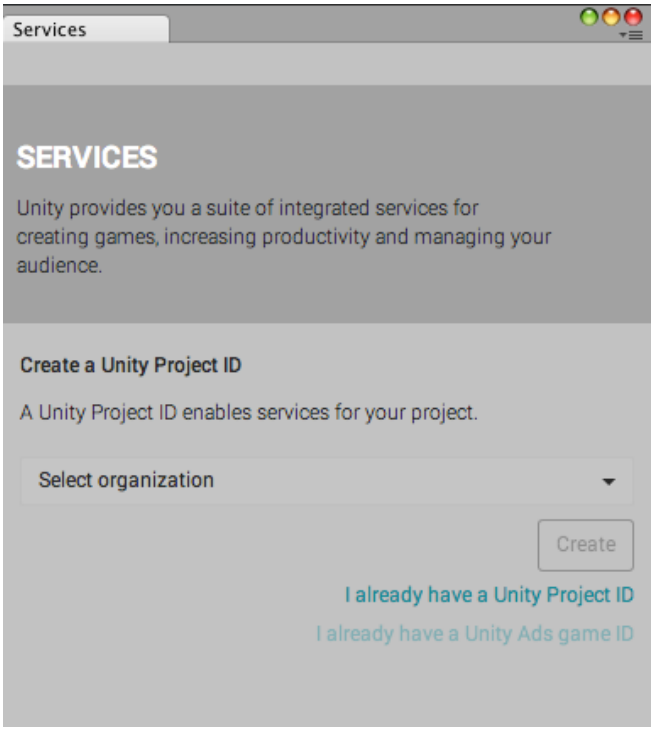
This game requires Unity services such as Ads and In-App Purchasing, therefore you need to create a new Unity project and link the game to it in order to use these services.

Note that in the development of the template, we need to link it to our own project for testing, therefore you may need to unlink it from our project before linking it to your own one. To unlink the project:

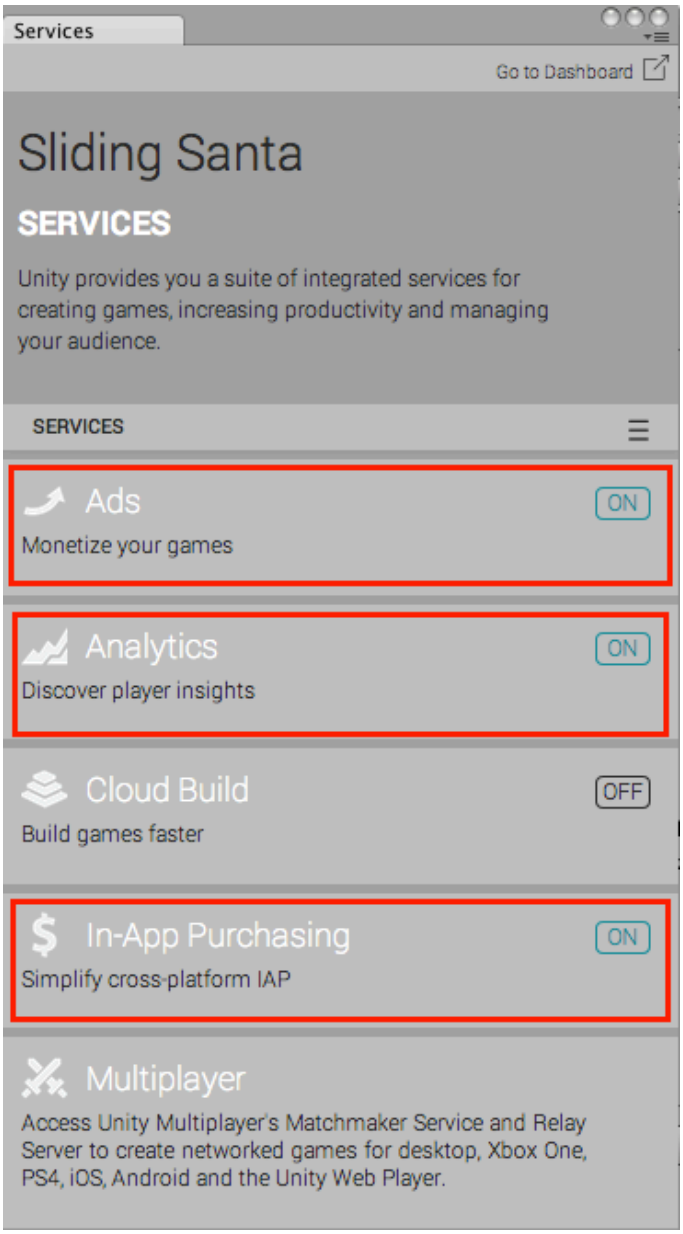
- Select Window -> Unity Services
- Select SETTINGS tab
- Click Unlink Project button



Now you can create a new project for the game.



Next you need to enable the **Ads**, **In-app Purchasing** and **Analytics** services (Analytics is required for In-App Purchasing). Note that you may see some errors in the console during this process but don't worry, they'll be gone after you turning the mentioned services on.



2.3 Install CocoaPods (for Xcode builds only)

If you're building for Xcode, you need to install **CocoaPods** as it is required by some plugins used in this game, e.g. AdMob (Google Mobile Ad). You can download CocoaPos from https://cocoapods.org/.

2.4 Perform test run and build

Now you can run the game in Unity! There're 2 scenes in folder **_SlidingSanta/Scenes**, you should run from scene **Main**.

You can also perform a test build for iOS and Android. Note that you may not be able to test with in-app purchases and leaderboards just yet. To test those features you need to create your own leaderboards, achievements and in-app products, please follow next sections for that topic.

IMPORTANT NOTE: After the Xcode project is built, you must run it from the **.xcworkspace** file rather than the **.xcodeproj** one. The former includes the Pod project, created by CocoaPods, which is required to use AdMob on iOS.

3 QUICK LAUNCH GUIDE

This section describes the step-by-step process to build and launch the game and requires no coding skills! You will be able to build your new game in probably a couple hours and start making money without writing a line of code. Seriously, isn't that cool?

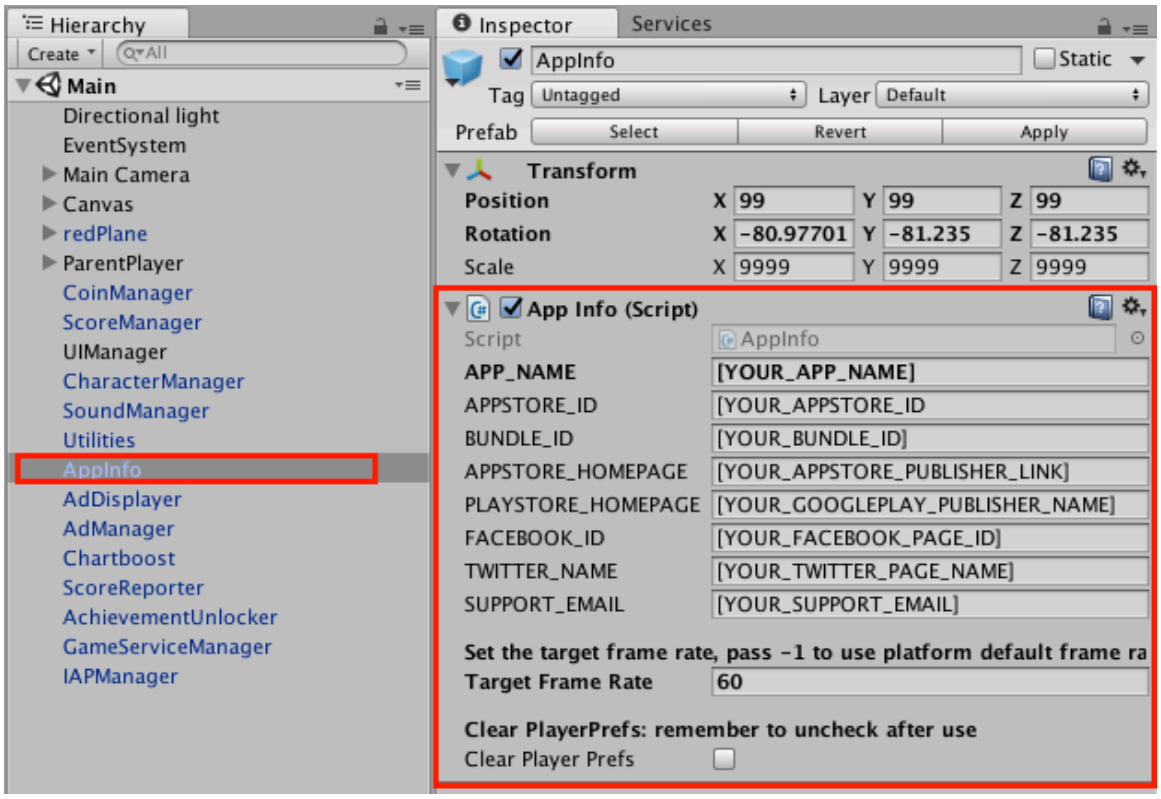
3.1 Change the Bundle Identifier (and Product Name)

The first step is to change the Bundle Identifier to your own. In Unity go to **Edit > Project Settings > Player** and change the Bundle Identifier field to the one you want. It should have the form of **com.companyName.productName**.

You can also change Product Name within this same window.

3.2 App information

The next step is fill in basic app information. The project contains a game object called **AppInfo** where you can fill in important app-related metadata like AppStore Id and Bundle Id. These values will be used for features like Share Button, Rate Us button, opening Facebook or Twitter fanpage as well as sending support emails from the game.



3.3 Advertisement setup

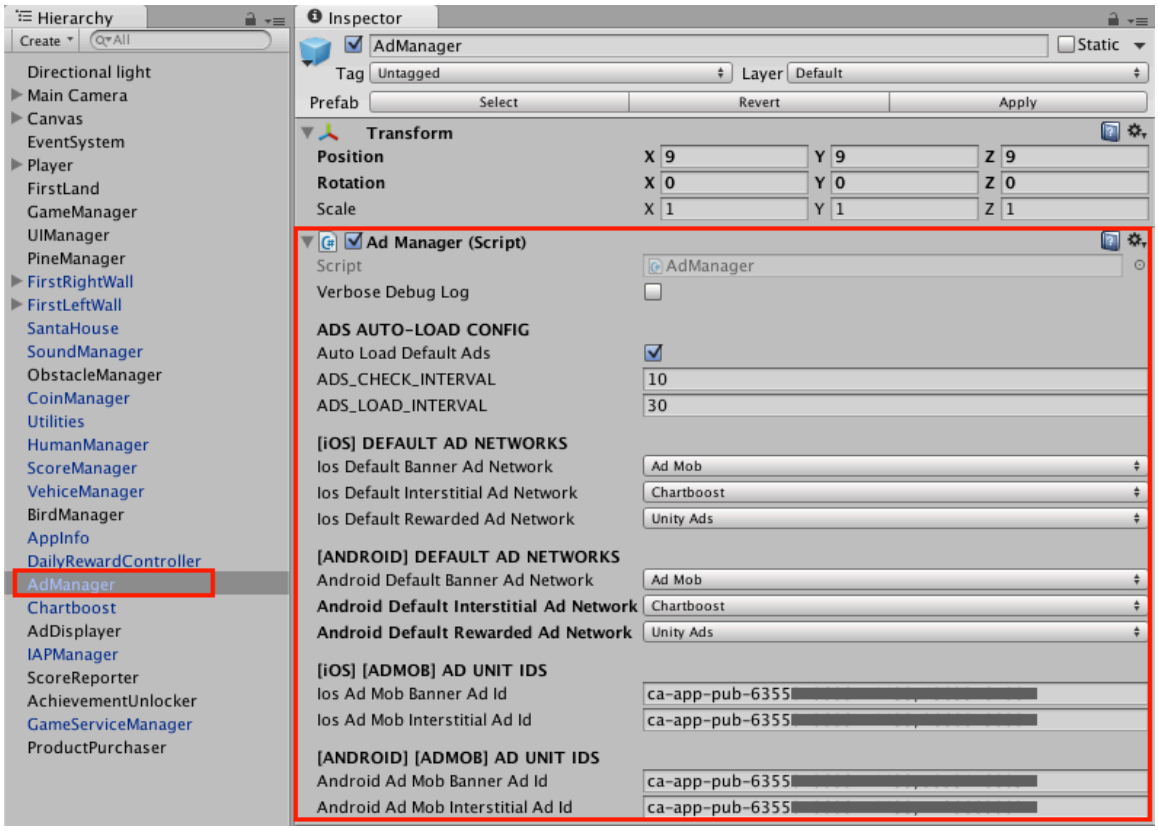
This game supports monetization with multiple advertising options. The next step is to setup and configure advertising activities to suit your needs.

By default, the game shows 3 kinds of ads:

- A banner ad at the screen bottom during gameplay.
- An interstitial ad (static or video depending on selected ad network) at game over, you can change how frequently it shows.
- A rewarded video that the user can watch to earn more gifts.

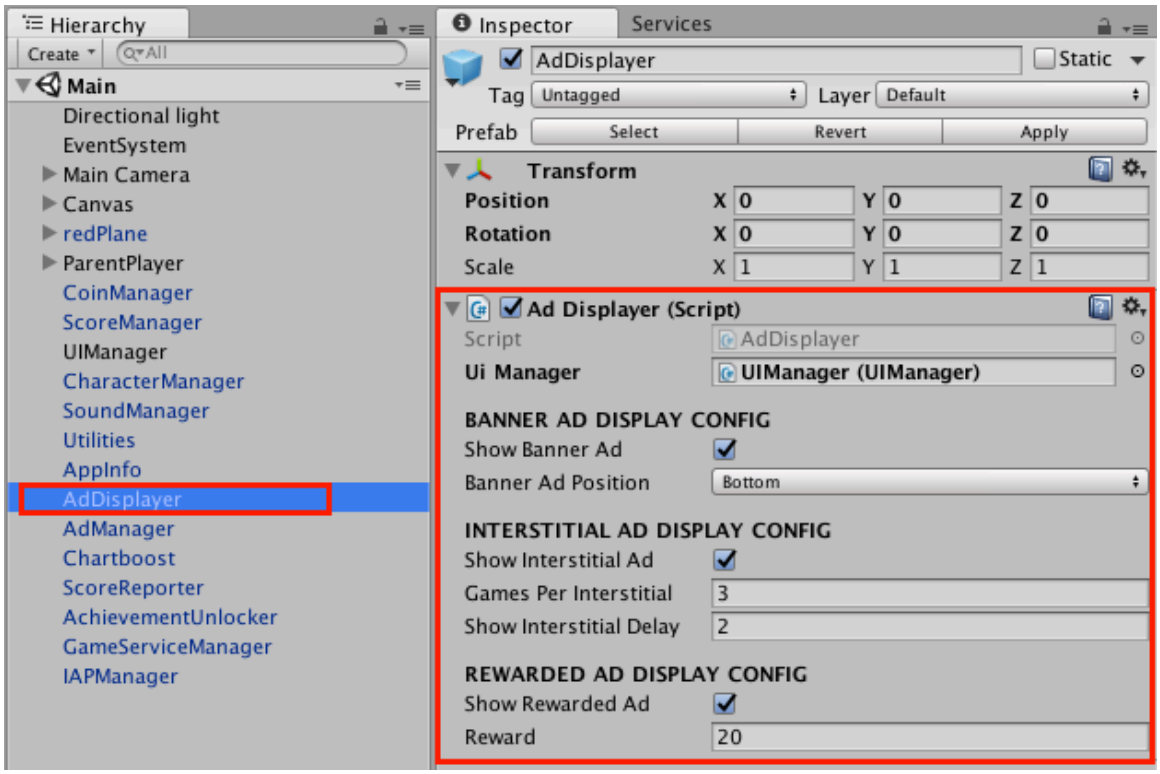
You can choose your preferred ad network for each of the ad type listed above by selecting the **AdManager** object from the hierarchy to reveal its **AdManager** component, then in the **[iOS] DEFAULT AD NETWORKS** and **[ANDROID] DEFAULT AD NETWORKS** you can select default ad networks for iOS and Android platform respectively.

If you don't want to show a certain type of ads in the game, simply set the corresponding ad network to **None**.



Next select the **AdDisplayer** object from the hierarchy to reveal its **AdDisplayer** component. Here you can configure how ads should be shown:

- **Show Banner Ad**: whether to show banner ad or not
- **Banner Ad Position**: where to show the banner ad
- **Show Interstitial Ad**: whether to show interstitial ad when game over
- **Games Per Interstitial**: how many games should be played before an interstitial ad is shown
- **Show Interstitial Delay**: how many delay seconds after game over that the interstitial ad is shown
- **Show Rewarded Ad**: whether to show rewarded ad
- **Reward**: the number of gifts rewarded after the user watched the ad



The last step is to setup the ad networks that you selected previously. Please follow the instructions below for each of the selected ad networks.

3.3.1 AdMob

To use AdMob you need to:

- Create a new AdMob app for each target platform (iOS and Android).
- Create a banner ad unit (if you use AdMob banner ad) for each created app.
- Create an interstitial ad unit (if you use AdMob interstital ad) for each created app.

If you're not familiar with the process please follow the guide provided by Google here https://support.google.com/admob/answer/2773509

Once you create the necessary ad units, you'll have the unit ids which look similar to **ca-app-pub-0664570763252260/3326342124**.

Now back to Unity, you need to fill in these ids to AdManager in **[iOS][ADMOB] AD UNIT IDS** and **[ANDROID][ADMOB] AD UNIT IDS** sections.

If you're not using AdMob for a certain ad type, feel free to leave the corresponding fields empty.

3.3.2 Chartboost

To use Chartboost you need to:

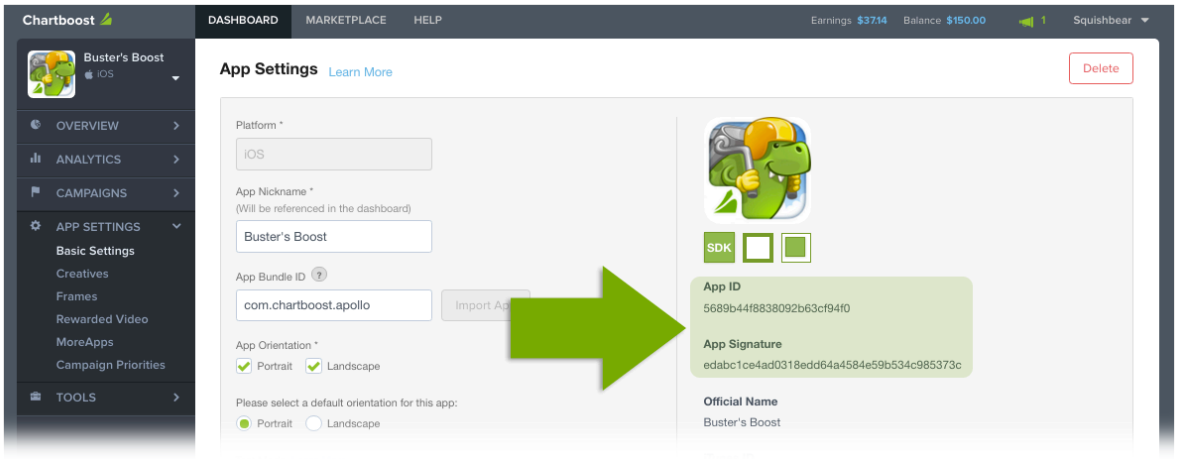
- Create a Chartboost app for each target platform. Set the app orientation to **Portrait**.
- Create an **Interstitial Publishing Campaign** (if you're using Chartboost interstitial ad) for each newly created app.
- Create a **Rewarded Video Publishing Campaign** (if you're using Chartboost rewarded ad) for each newly created app.

If you're not familiar with the process, please follow the guides provided by Chartboost:

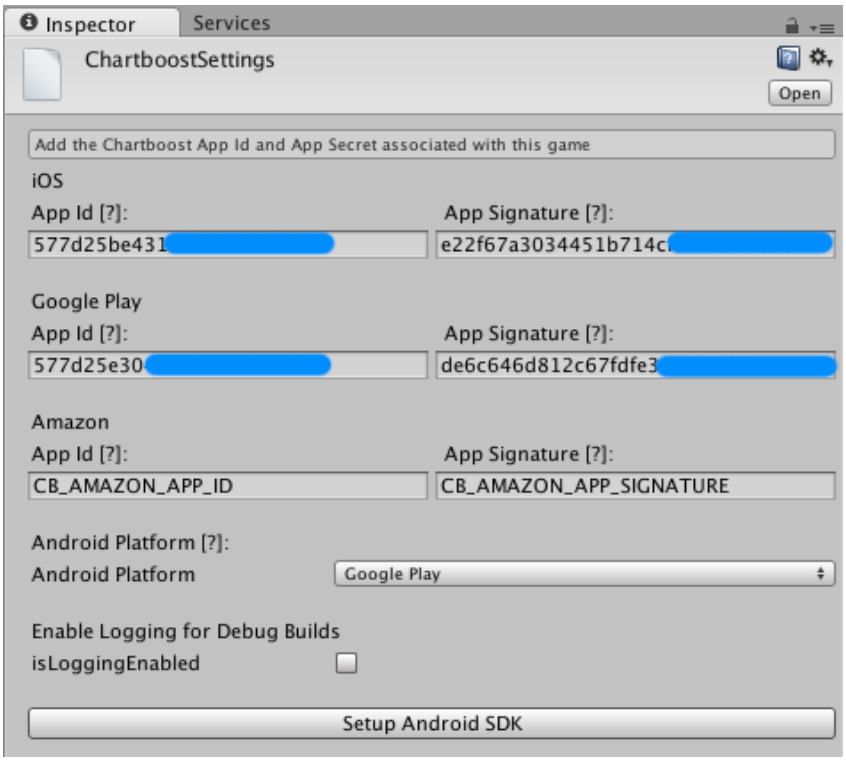
- Create a new app: https://answers.chartboost.com/hc/en-us/articles/200797729-Adding-Your-First-App-and-Campaign-
- Create a publishing campaign: https://answers.chartboost.com/hc/en-us/articles/204930539-Creating-a-Publishing-Campaign

Once you create a new Chartboost app, you'll have its **ID** and **Signature**:

- Select your app in the upper left corner of your Chartboost dashboard.
- Go to App Settings > Basic Settings.
- Your app ID and app signature appear underneath your app's icon.



Back to Unity, select menu **Chartboost>Edit Settings** and fill in the app Id and Signature for each target platform.

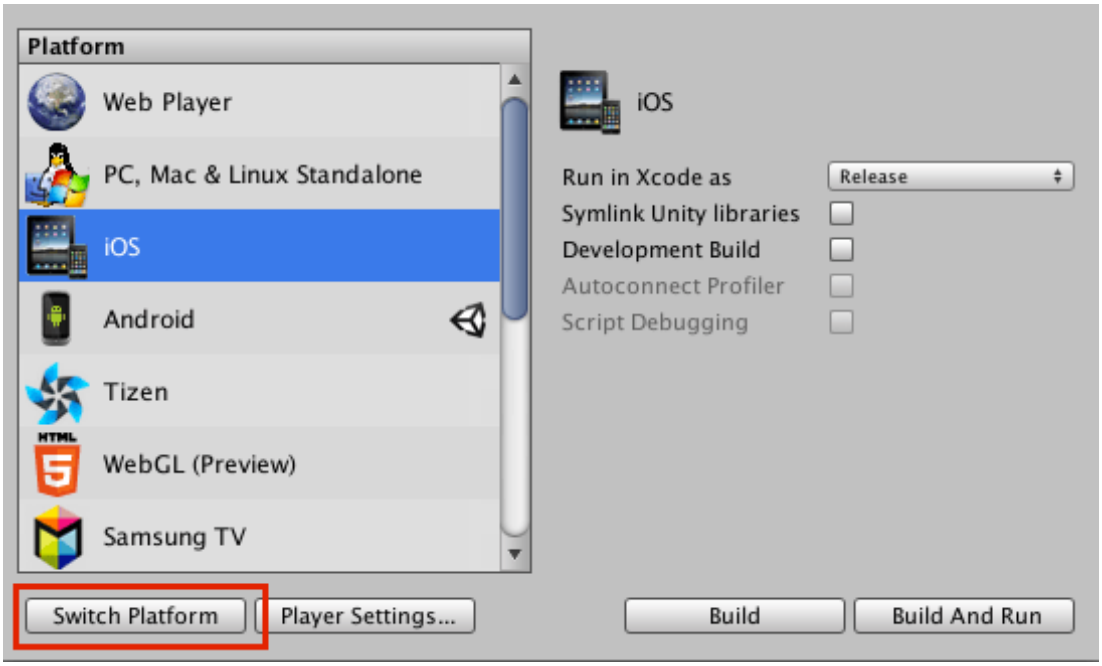


** Note: there's no need to click **Setup Android SDK**, it is setup for you already!*

3.3.3 UnityAds

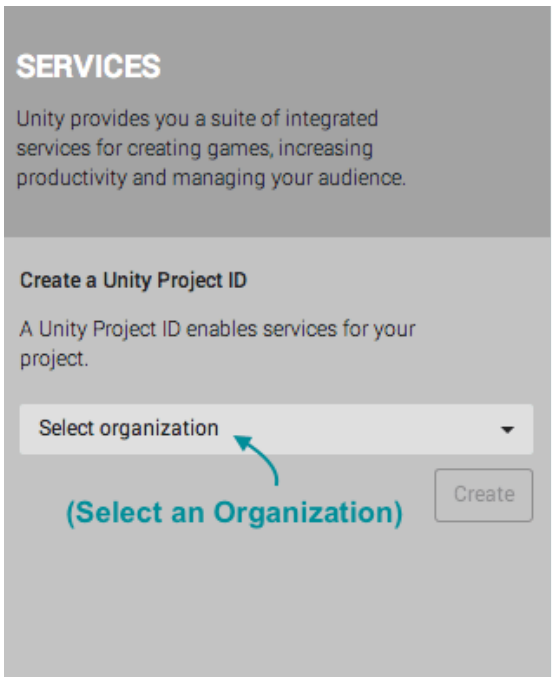
UnityAds is built-in to the engine since Unity 5.3.0 so setup is very easy. Here are the simple steps to enable UnityAds in this game.

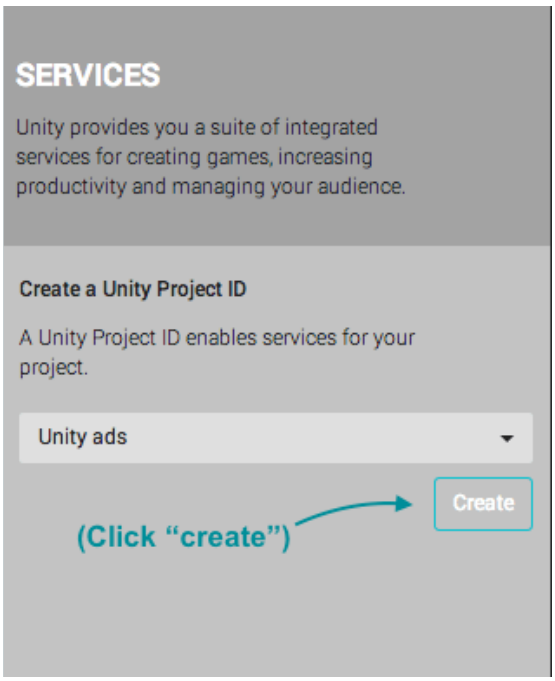
Switch the platform to iOS or Android

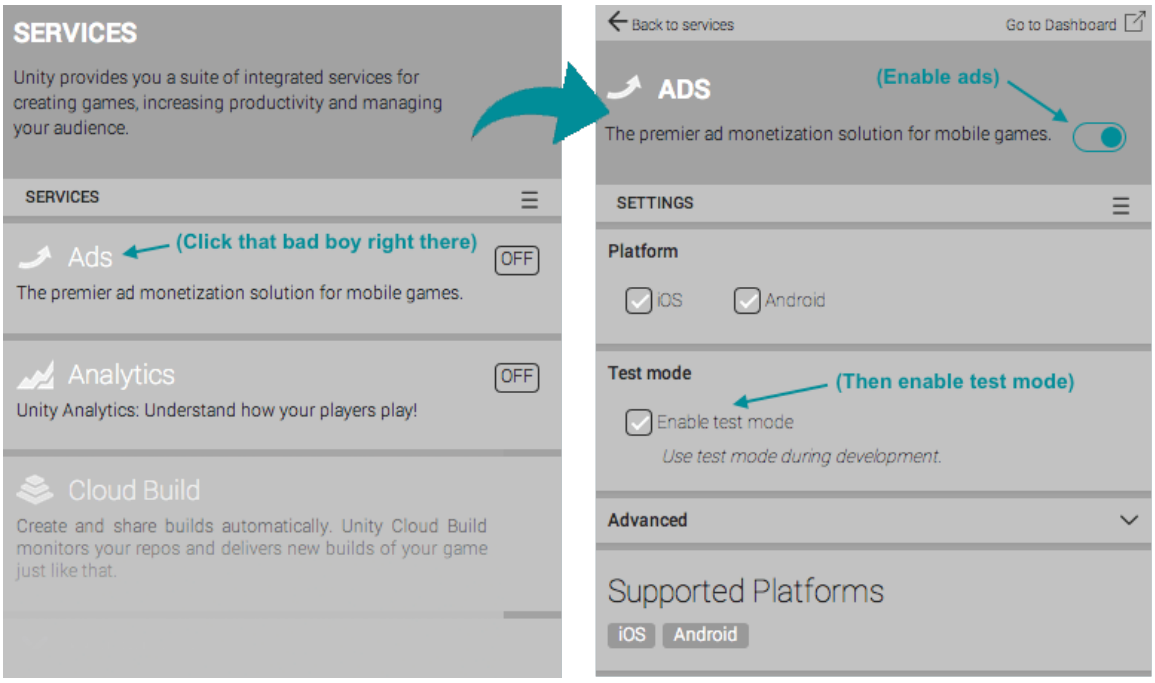


Enable ads in the Unity Services panel

- Select Window -> Unity Services
- Login using your Unity account if needed
- Enable Ads service if needed







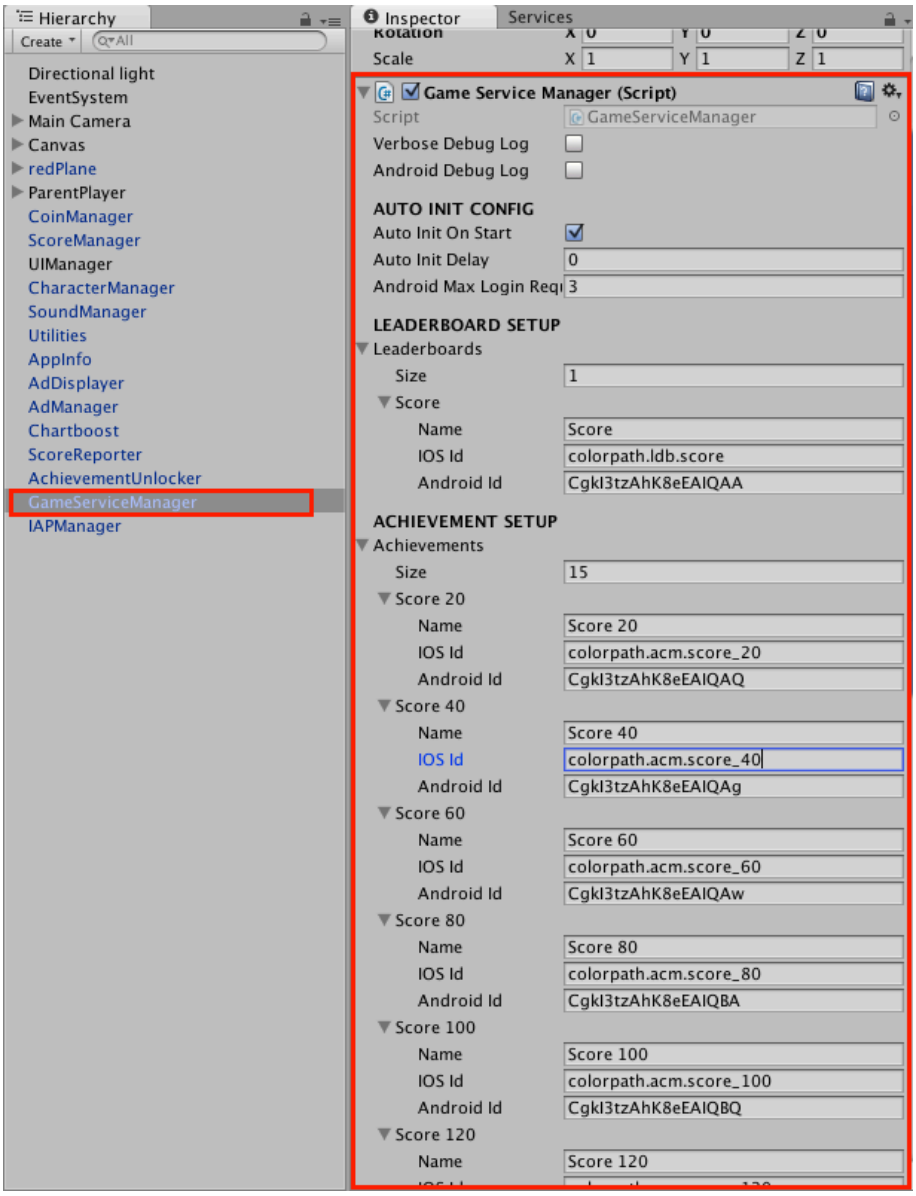
You can find more information about getting started with UnityAds at https://unity3d.com/services/ads/quick-start-guide.

3.4 Leaderboards and Achievements Setup

This game supports services like leaderboards and achievements.

- On iOS, it uses the official Game Center plugin provided by Unity.
- On Android, it uses the official Google Play Game Services Unity plugin provided by Google.

You can setup leaderboards and achievements in **GameServiceManager**.



3.4.1 Leaderboards

The game comes with 1 leaderboard named **Score** which ranks the users by the score they achieved in game.

Scores will be reported automatically after each game thanks to the **ScoreReporter** object. You only need to provide the IDs of the leaderboards for each target platform.

First you need to create a leaderboard named **Score** for each platform. If you're not familiar with the process, please follow guides provided by Apple and Google:

- Create leaderboards in iTunes Connect: https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/GameKit_Guide/Achievements/Achievements.html#//apple_ref/doc/uid/TP40008304-CH7-SW10
- Create leaderboards in Google Play Developer Console: https://developers.google.com/games/services/common/concepts/leaderboards

Once you create the required leaderboard for each platform, enter its ID to appropriate fields in the **LEADERBOARD SETUP** section within the GameServiceManager object mentioned above. Please note that you should leave the leaderboard names unchanged for the ScoreReporter to function as expected.

3.4.2 Achievements

The game comes with 20 pre-designed achievements: **Score_10**, **Score_20**, **Score_30**, etc. These achievements are to be unlocked when the user achieves the corresponding score milestone e.g. "Score_20" will be unlocked once the user scores 20 for the first time. The game will automatically monitor user score and unlock appropriate achievement, thanks to the **AchievementUnlocker** object. You only need to provide the IDs of the achievements for each target platform.

First you need to create the required achievements with names as listed in the **Achievements** array in **ACHIEVEMENT SETUP** section within the GameServiceManager object.

** Tip: You can have less achievements than listed in the Achievements array as long as their names match.*

If you're not familiar with the process of creating achievements, please follow guides provided by Apple and Google:

- Create achievements in iTunes Connect: https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/GameKit_Guide/Achievements/Achievements.html
- Create achievements in Google Play Developer Console: https://developers.google.com/games/services/common/concepts/achievements

Note that you should create the achievements as:

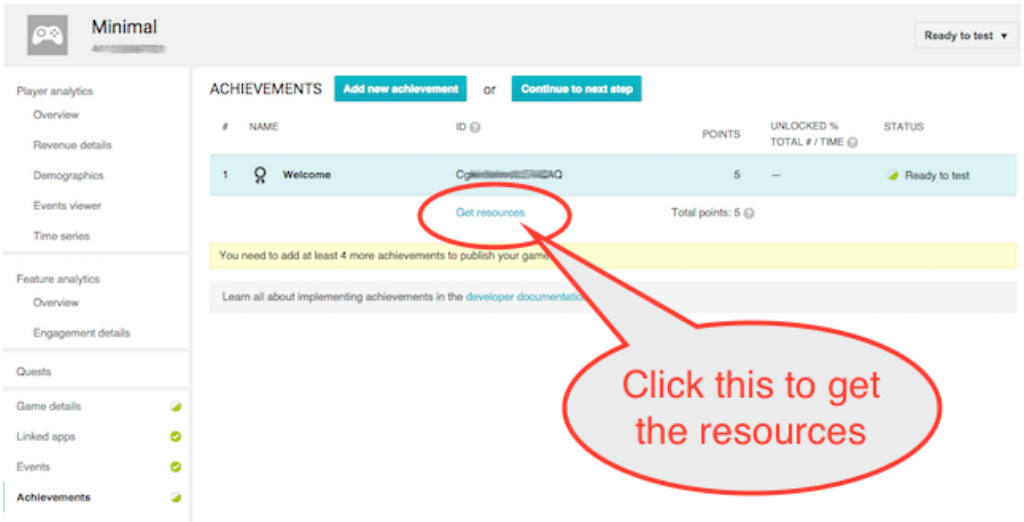
- **Revealed** (not Hidden)
- NOT **Available More Than Once** (iOS)
- NOT **Incremental achievements** (Android)

Once you create the required achievements, enter their IDs to appropriate fields in the GameServiceManager object. Note that you should leave the achievement names unchanged for the AchievementUnlocker to function as expected.

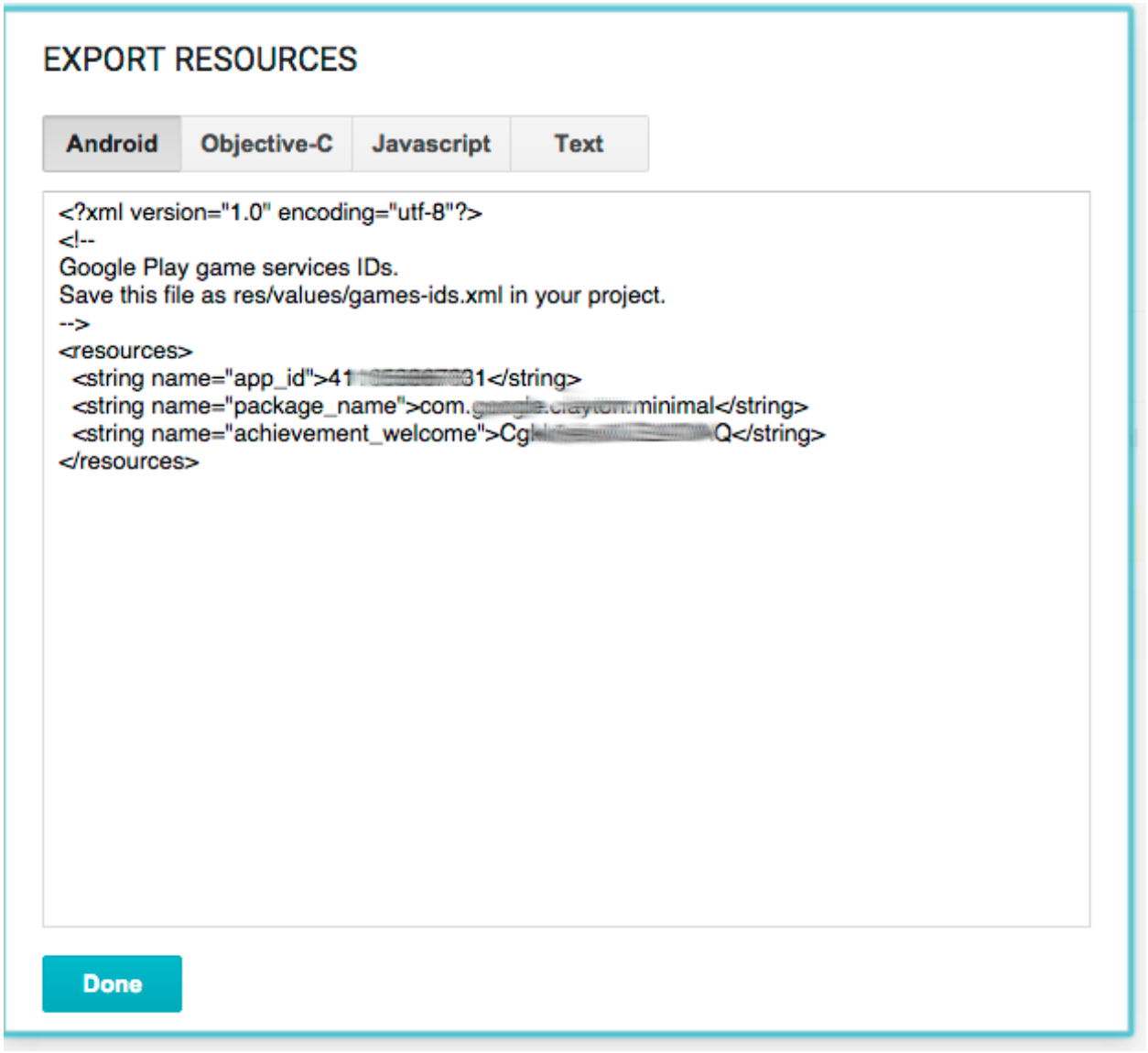
3.4.3 Configure Google Play Game Services for Android

Now that you've created the required leaderboards and achievements, the final step in this section is to configure the Google Play Game Services plugin in Unity. This is for Android build only.

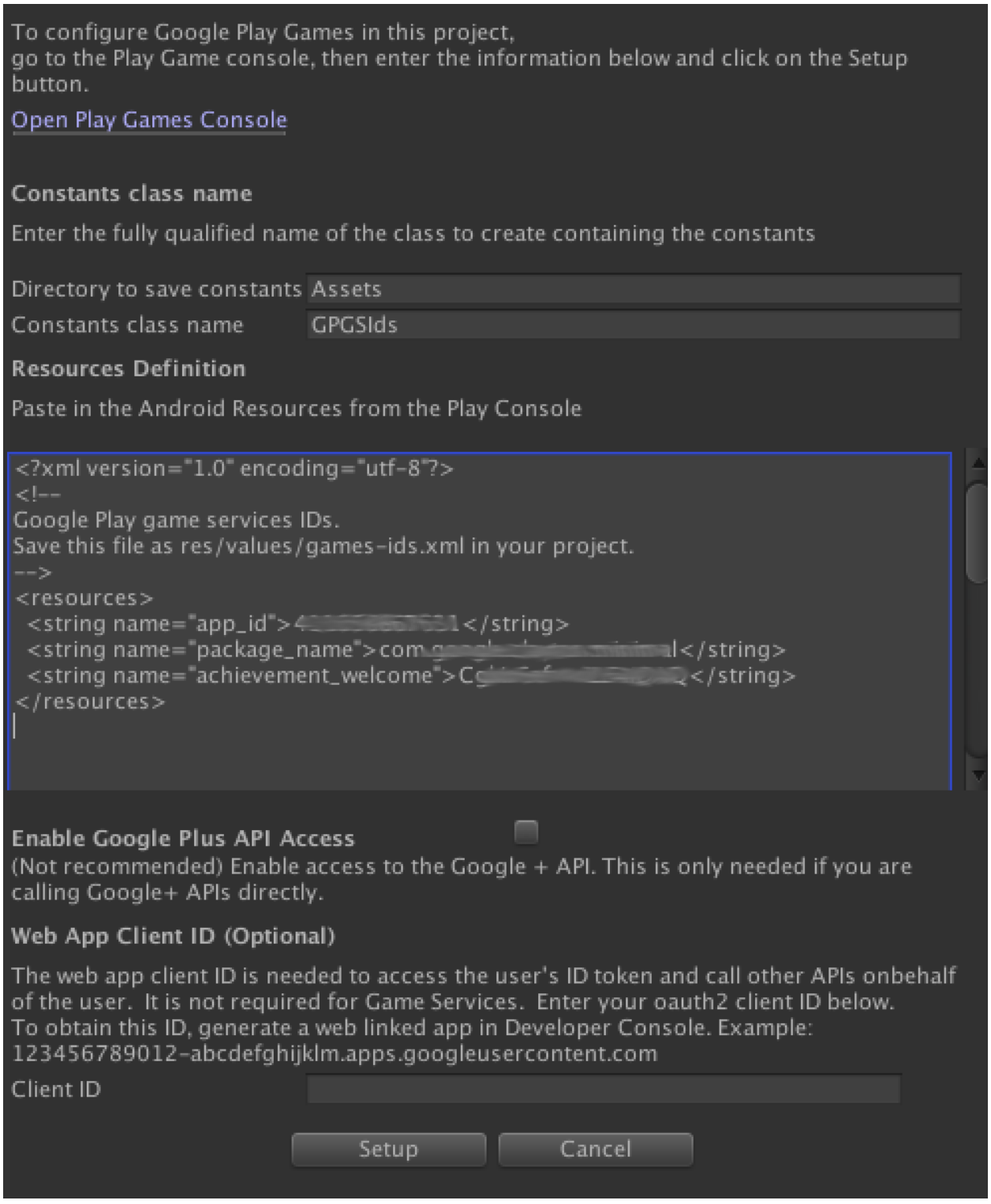
First you need to copy the game resources from Google Play Developer Console. To get the resources go to the the **Achievements** tab, then click on "Get resources" on the bottom of the list.



Copy text from the Android tab.



Back to Unity, open the setup dialog at **Window > Google Play Games > Setup... > Android Setup**, if you don't see it, switch to Android platform in **Build Settings**. Now paste the copied text to the dialog, set the **Constants class name** to **GPGSIds**, then hit **Setup**. You should leave other options unchanged.



Now the game is ready for leaderboards and achievements!

3.5 In App Purchase Setup

This game is ready for in-app purchases on both Apple App Store and Google Play Store. It comes with a Remove Ads button (and a Restore Purchases button as required by iOS) that allows the user to pay a small amount of money to permanently stop showing ads during in the game. It also has a Store where the user can spend real money to buy in-game gifts. Please follow the instructions in the next sections to enable this feature in your game.

3.5.1 Enable Unity IAP Service

First make sure Unity In-App Purchasing service is enabled. If you're not familiar with the process please follow the detailed step-by-step guide here http://docs.unity3d.com/Manual/UnityIAPSettingUp.html. Note that you don't need to import the IAP package (Step 4) since it is included in the project already.

3.5.2 Create IAP Products for Target Stores

The next step is to create the IAP products for our target stores i.e. **Apple App Store** and **Google Play Store**.

- For Apple App Store, create a new IAP product in **iTunes Connect**. If you're not familiar with the process, please follow the guide here http://docs.unity3d.com/Manual/UnityIAPAppleConfiguration.html. This guide also includes the instructions for testing IAP with Apple App Store.
- For Google Play Store, create a new IAP product in **Google Play Developer Console**. If you're not familiar with the process, please follow the guide here http://docs.unity3d.com/Manual/UnityIAPGoogleConfiguration.html. This guide also includes the instructions for testing IAP with Google Play Store.

IMPORTANT NOTES:

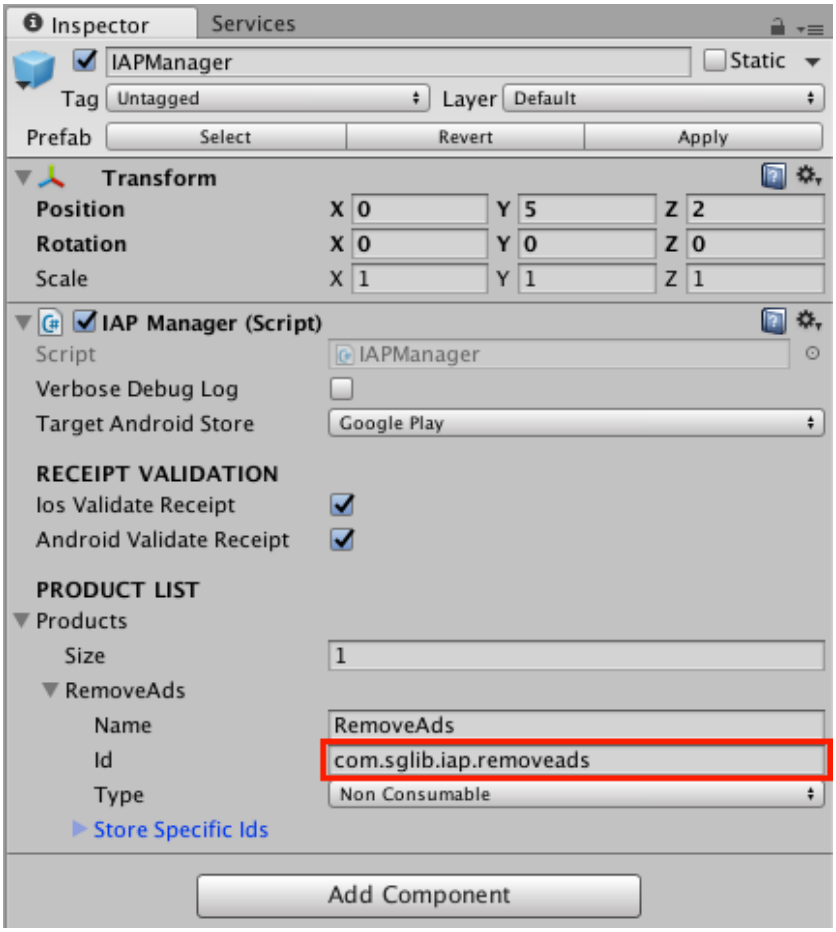
- The template has 6 pre-designed products: **Remove Ads** (non-consumable) and 5 consumable products: **200_Gifts**, **450_Gifts**, **1000_Gifts** and **3000_Gifts**.
- Non-consumable products should be marked as **Non Consumable** in iTunes Connect and **Managed** in Google Play Developer Console; and they should have the type of **Non Consumable** when added to

IAPManager in Unity.

- For Google Play Store, all products should be **Managed** product.
- Use the same product ID for both platform, the product ID should be in reverse-domain notation, for instance **yourgame.iap.removeads**
- To create an IAP product in Google Play Developer Console, you must first build a APK and upload it to **ALPHA TESTING**.

3.5.3 Configure IAPManager in Unity

Now that you have set up your IAP products for the target stores, the next step is go back to Unity and configure the **IAPManager** module, which is located at the IAPManager object in the hierarchy.



IMPORTANT NOTES:

- The existing product names and types should be left unchanged.
- Copy and paste the product IDs you chose in the previous step to the **Id**

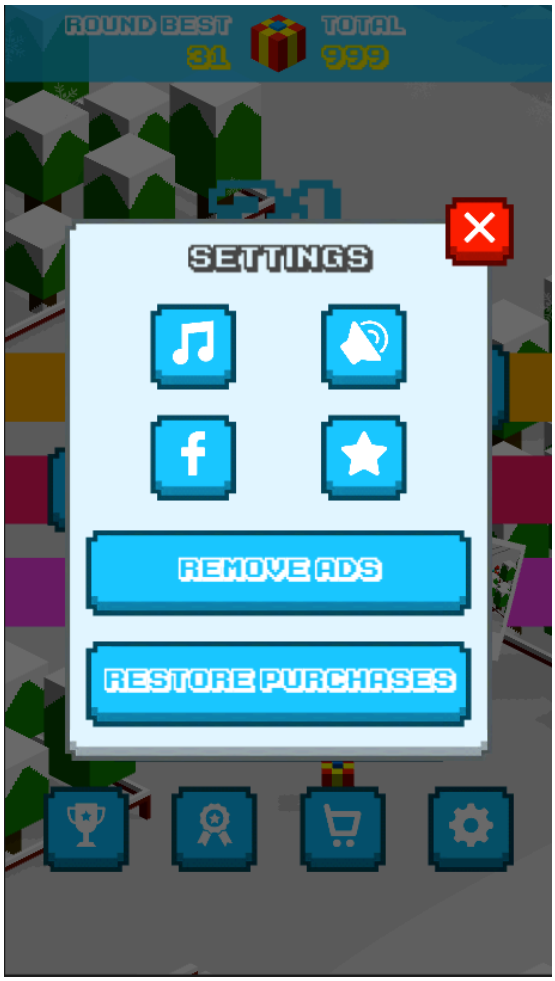
field of the appropriate product.

- When build for Android, you must select your target store using the dropdown at **TargetAndroidStore** in IAPManager AND then select the same store at menu **Window > Unity IAP > Android**.

3.5.4 Modifying IAP products

You can set the prices of the IAP products as you wish. You can also add more products beside the existing ones (or you can remove the existing products if you want). If these are the case, you need to update the Store UI to reflect the changes. The **Store** object is located under the **Canvas** object in the hierarchy. Below is the built-in storefront of the template.

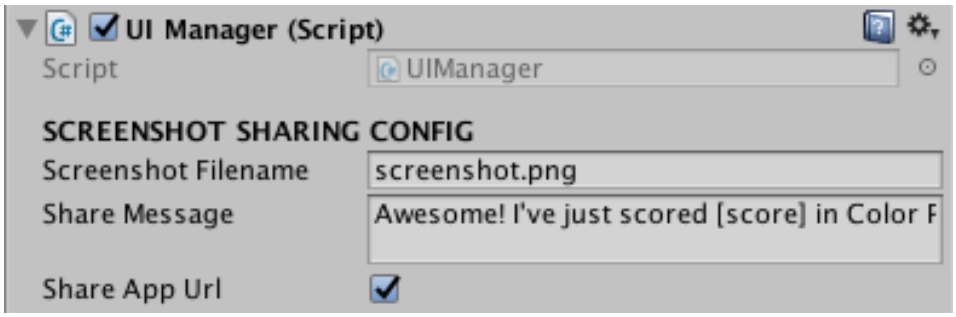
The Remove Ads and Restore Purchases buttons are located in the Settings panel.



Also note that the purchasing of products is handled by a class named **ProductPurchaser**, so if you're adding new products you must update this class to provide your own logic for the new products.

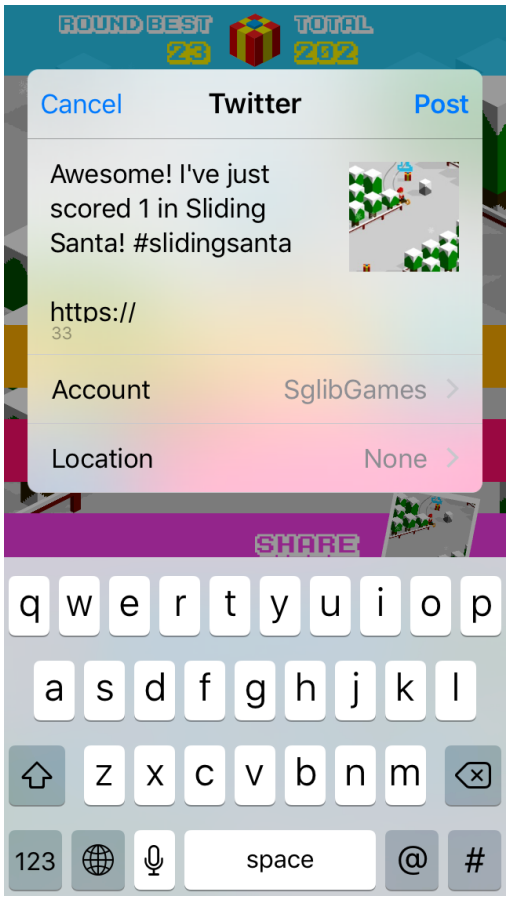
3.6 Screenshot Sharing

The game comes with a Share button that allows the user to share their results to social networks using the native sharing utility. You can customize this feature from the **UIManager** component, located within the namesake object.



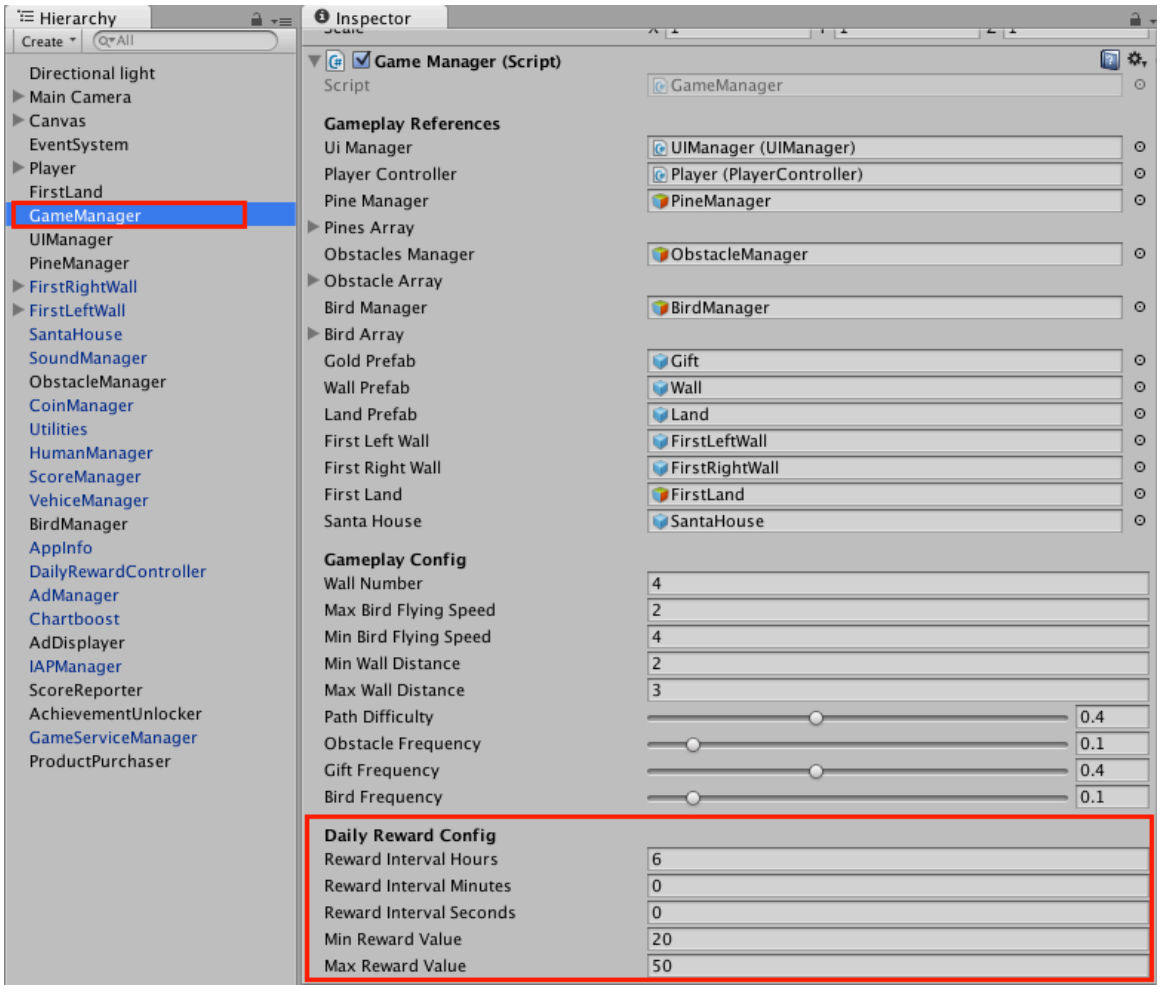
In the **SCREENSHOT SHARING CONFIG** section there're a few options:

- **Screenshot Filename**: the screenshot file name on device storage.
- **Share message**: the default message that gets shared with the screenshot, you can use hashtag here. Note that any instance of "[score]" will be replaced by the actual score value at runtime.
- **Share App Url**: check this to attach the app download links on App Store and Google Play. These are generated automatically from the **APPSTORE_ID** and **BUNDLE_ID** that you fill into the **AppInfo** object in previous step.



3.7 Daily Reward

This game has a built-in daily reward system, which rewards the player with some gifts every predefined amount of time. This is an effective way to incentivize the player to keep coming back and play the game, which will likely result in better retention and engagement! To configure this feature, select the **GameManager** object in the hierarchy.



- **Reward Interval Hours**, **Reward Interval Minutes** and **Reward Interval Seconds**: the number of hours, minutes and seconds between two rewards.
- **Min Reward Value** and **Max Reward Value**: the actual number of gifts reward will be randomized between these 2 values, and then rounded so that it's a multiple of 5.

3.8 Build

The last step is, of course, build the game for target platforms and release. There're a few things you need to set in **Player Settings**:

- You must set the **Bundle Identifier** to the one you use when create the app on iTunes Connect as well as Google Play Developer Console.
- When build for Xcode, make sure the **NO_GPGS** symbol is listed in the **Scripting Define Symbols** field. This is to exclude the unused Google Game Services library when build for iOS.
- After the Xcode project is built, you must run it from the **.xcworkspace** file rather than the **.xcodeproj** one. The former includes the Pod project which is required to use AdMob on iOS.

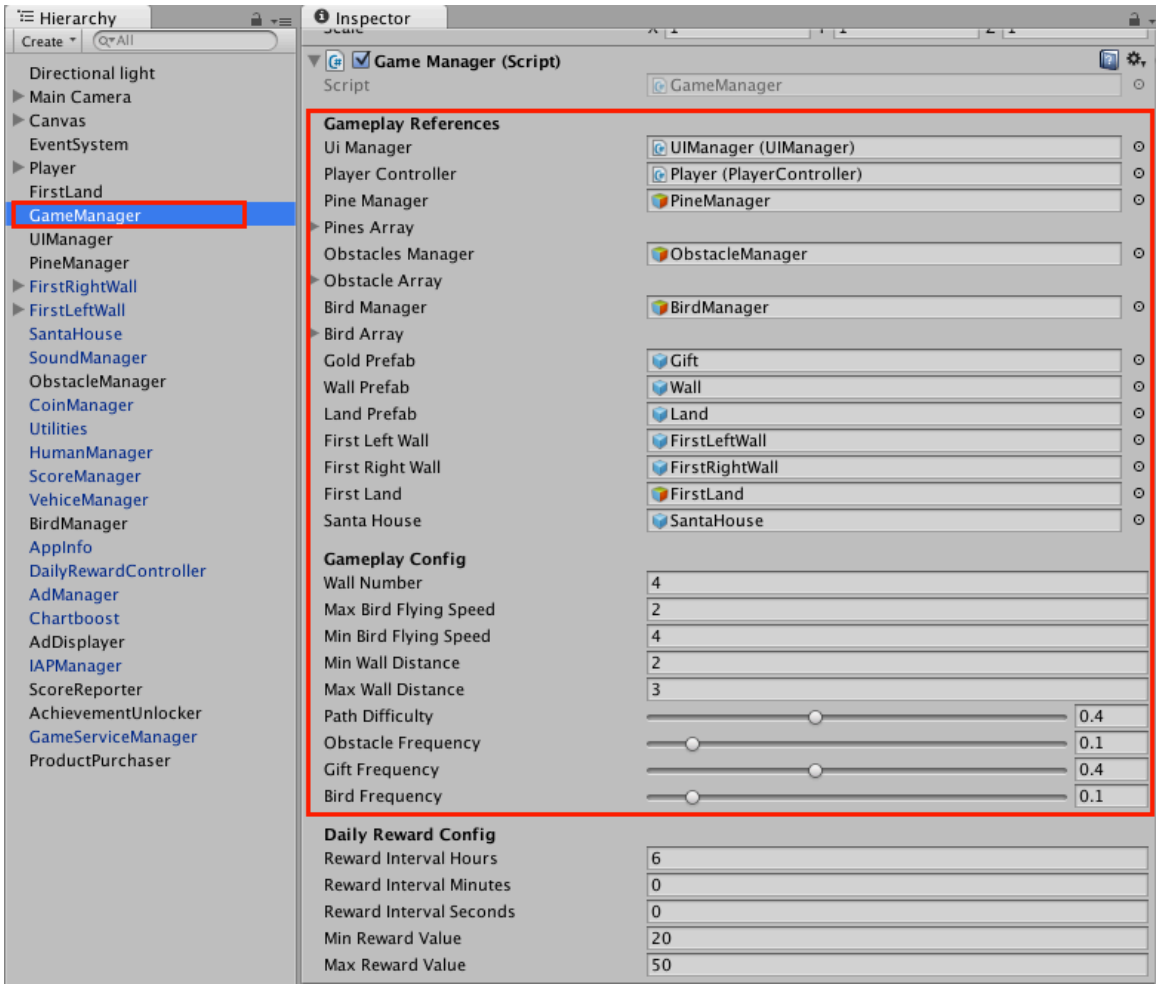
That's it! Now everything is up and ready. It's time to build your new game and submit it to app stores. Congrats!

4 HOW TO CUSTOMIZE GAMEPLAY AND UI

4.1 Gameplay tweaking

4.1.1 GameManager

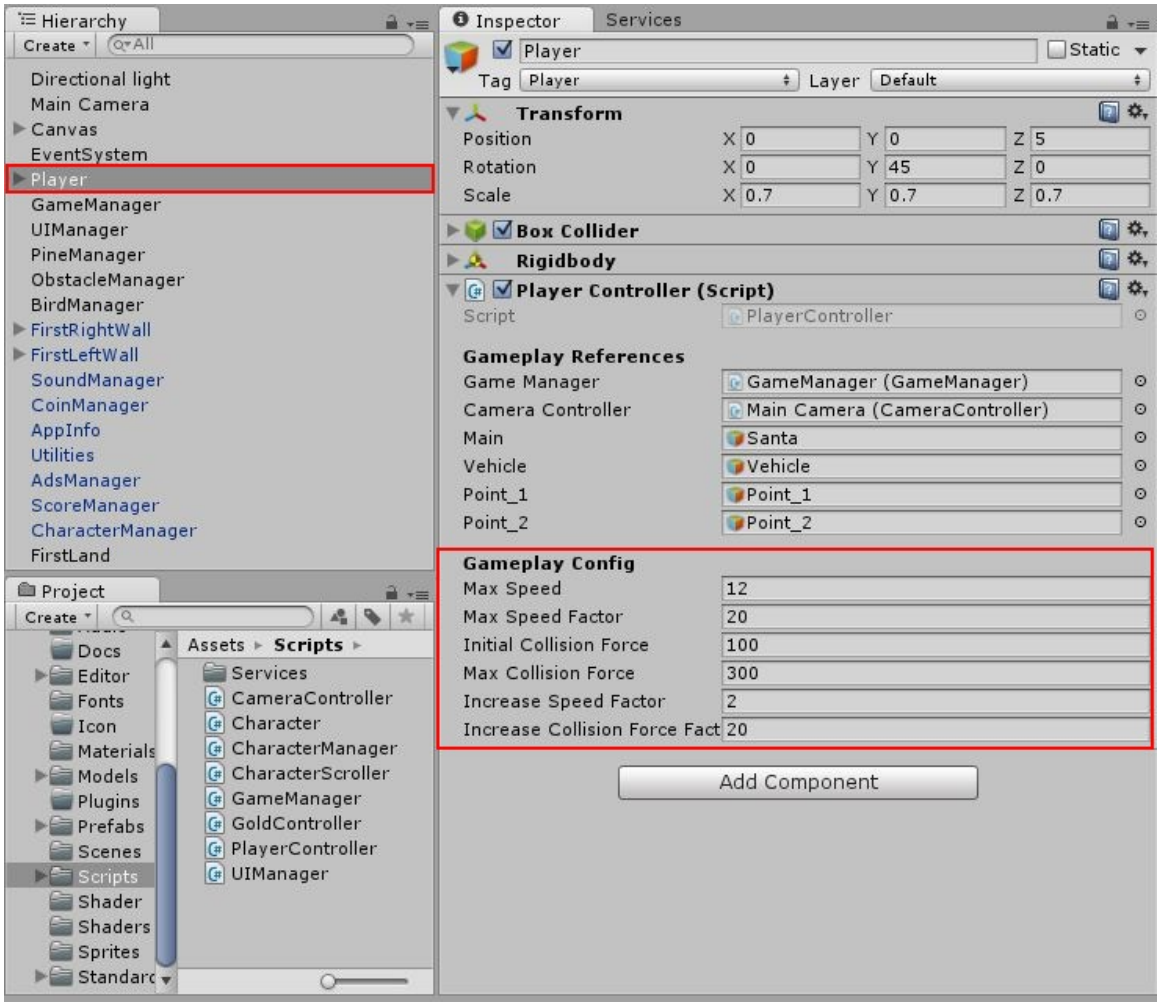
Most of important game parameters are located in **GameManager** component, which is attached to **GameManager** game object.



You can tweak the gameplay by modifying following variables:

- **WallNumber**: how many wall is created when the game start.
- **MaxBirdFlyingSpeed**: max speed of the bird.
- **MinBirdFlyingSpeed**: min speed of the bird.
- **MinWallDistance**: min distance of 2 walls.
- **MaxWallDistance**: max distance of 2 walls.
- **PathDifficulty**: the larger value will generate the path harder.
- **OnstacleFrequency**: the larger value will generate more obstacle.
- **GiftFrequency**: the larger value will generate more gold.
- **BirdFrequency**: the larger value will generate more bird.

4.1.2 PlayerController



- **MaxSpeed:** how fast player turn right (max value)
- **SpeedFactor:** how fast player turn left
- **InitialCollisionForce:** the collision force when the game start
- **MaxCollisionForce:** max value of collision force
- **IncreaseSpeedFactor:** how fast player reach max speed
- **IncreaseCollisionForceFactor:** how fast player reach max collision force

4.2 Adding more characters

Out-of-the-box, **Sliding Santa** is already packed with dozens of characters and vehicles, cute and ready to use!

If you want to add more vehicles, follow these simple steps:

i. Create a vehicle model.

ii. Navigate to **_SlidingSanta/Prefabs/Characters/VehiclePrefabs** and duplicate one of the available vehicle prefabs.

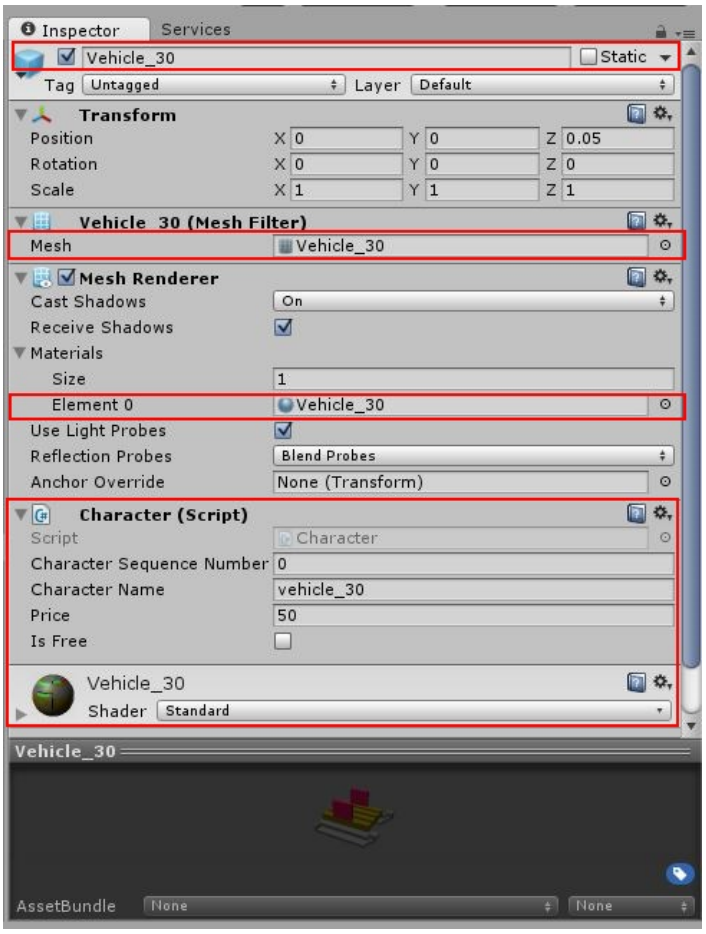
iii. Change the name of the prefab to a preferred one.

iv. Replace the **Mesh** in the **MeshFilter** component with your new model mesh.

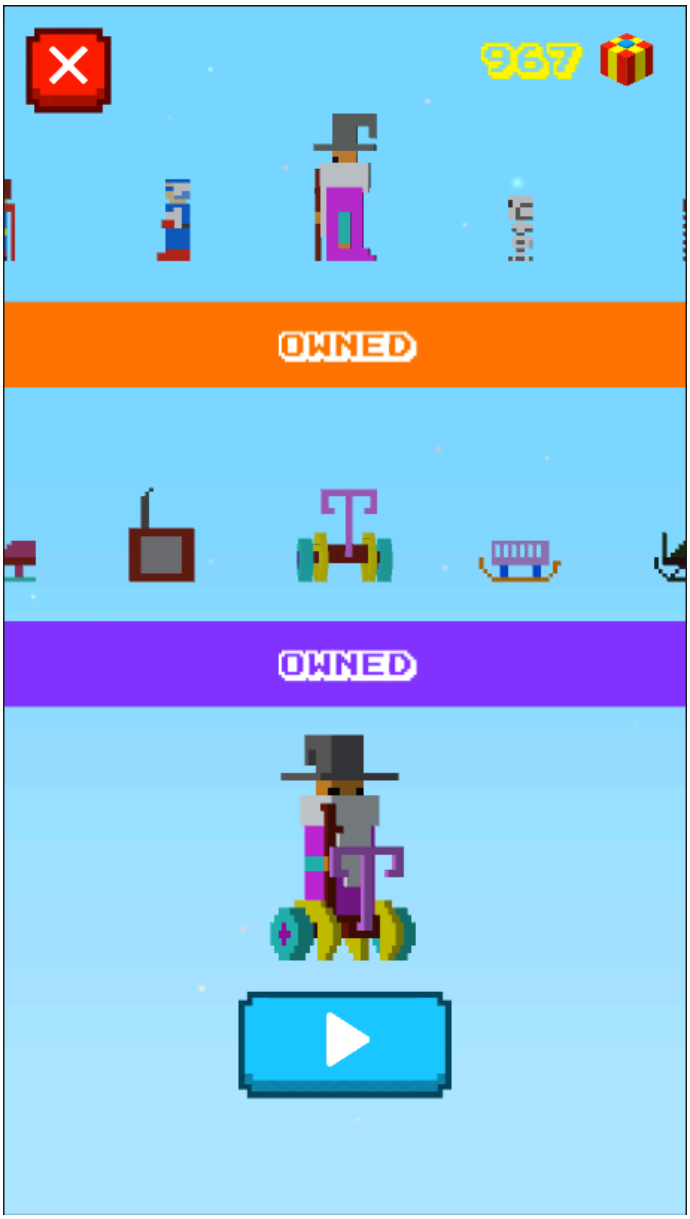
v. Replace the **Material** in the **MeshRenderer** component with your new character material.

vi. Enter the vehicle name and price to the **Character** component. Check the **isFree** box if you want to give out this character for free (it will be automatically unlocked).

vii. Resize the array in **VehicleManager** game object then drag the new character to it and hit Apply to save changes to its prefab.



Now the new vehicle has been added and ready to use in game! You will see it listed in the **CharacterSelelection** scene.

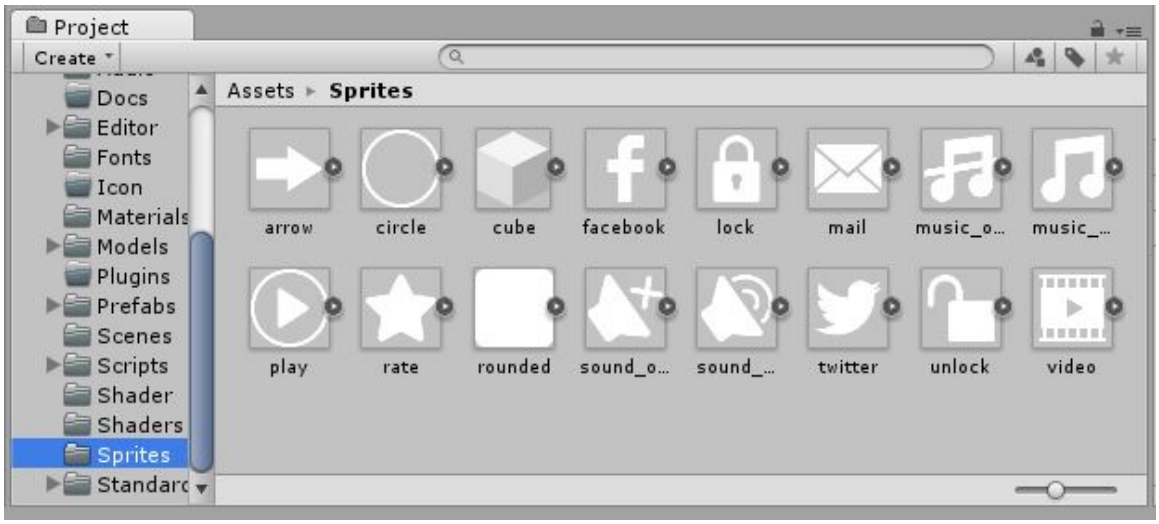


You can add more characters similar to adding vehicles by creating new characters in **HumanPrefabs** folder and add it to **HumanManager** game object.

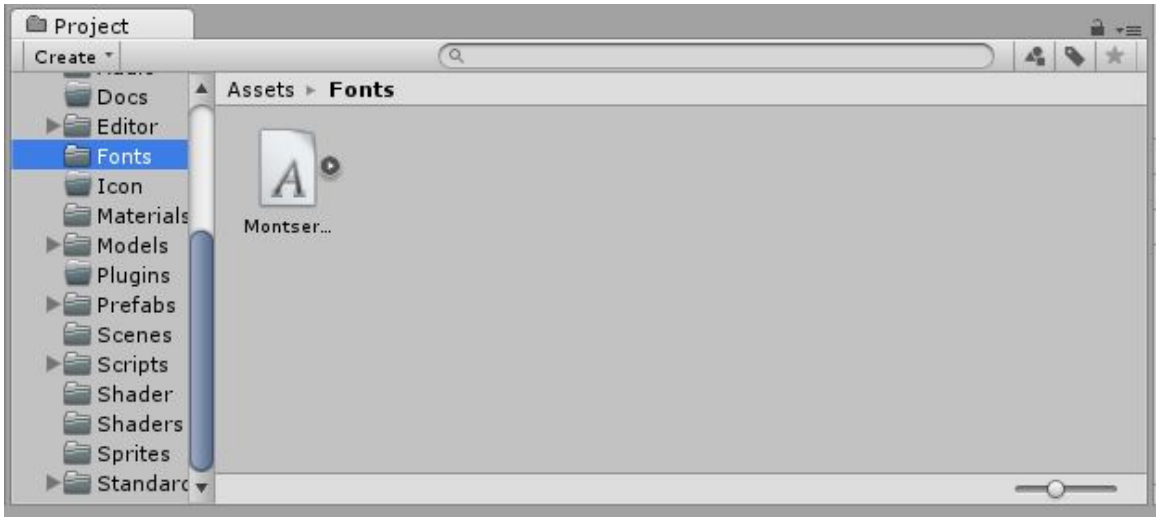
IMPORTANT NOTE: *new character must have a different name than all existing characters***.**

4.3 UI

All sprites used in this game (for buttons and other UI components) are located in folder **Asset/Sprites**. You can replace them with your own sprites to modify the UI as you like.



All fonts used in this game are free-to-use in commercial projects. Fonts are located under folder **Assets/Fonts** together with appropriate license files.

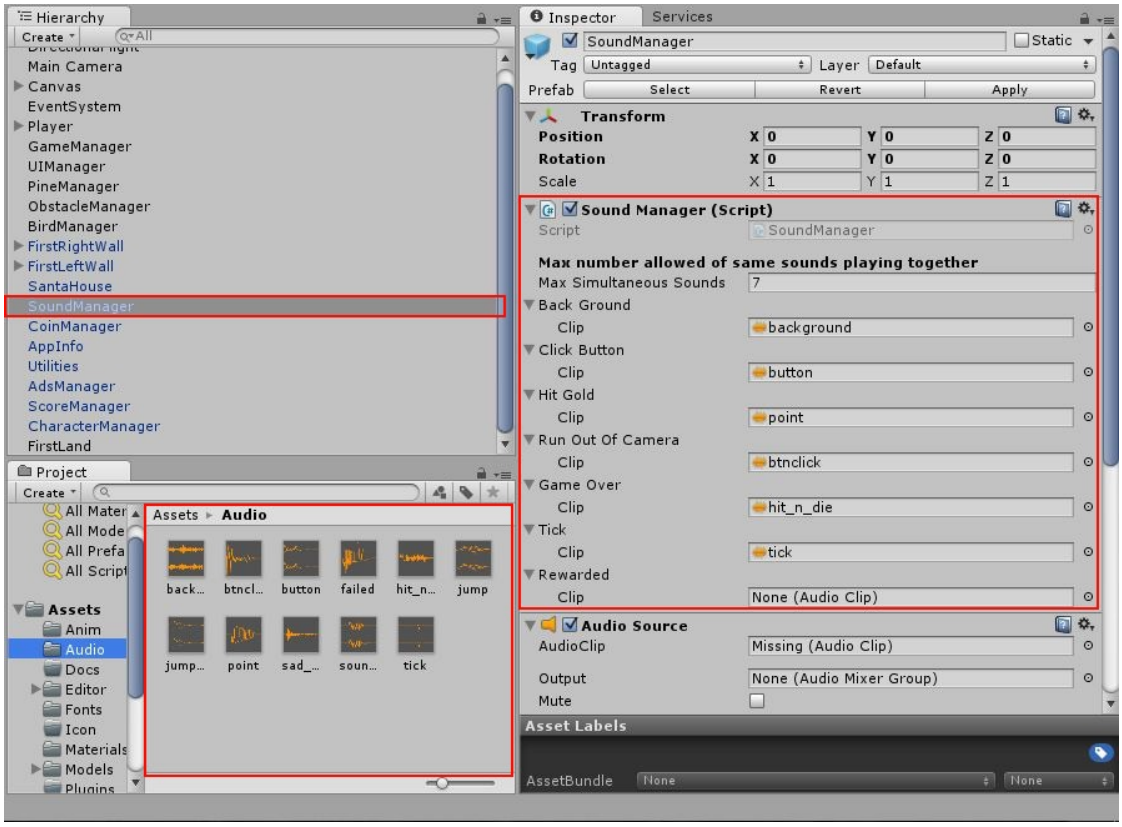


4.4 Sounds

All sounds included in this game are free-to-use in commercial projects and are located in folder **Assets/Audio**.

This game features a **SoundManager** class to manage all sound activities in game like playing sound or mute/unmute sounds. SoundManager has a feature to automatically reduce volume when multiple copies of the same sound are played together. This will effectively prevent the sound amplitudes from adding up and becoming annoyingly too loud.

If you want to replace sounds in this game, simply drag and drop new sounds to appropriate slots in the SoundManager component.



4.5 Custom Advertising

If you don't like the way ads are shown automatically and want to decide when, where and how ads should be served, simply disable or remove the AdDisplayer object from the hierarchy. The AdManager provides you with convenient methods to load and show ads. It also regularly checks for ads readiness and perform automatic ads loading if necessary (using the default ad networks). This is to ensure that default ads are loaded early and available the moment you need them to be shown. Note that AdManager employs singleton pattern so you can access its method from any class via the public member **Instance**.

To show the default banner ad (one that uses the default banner ad network):

```
AdManager.Instance.ShowBannerAd(bannerPosition);
```

To hide the default banner ad:

```
AdManager.Instance.HideBannerAd();
```

To load the default interstitial ad:

```
AdManager.Instance.LoadInterstitial();
```

To check if the default interstitial ad is ready call

```
AdManager.Instance.IsInterstitalReady();
```

To show the default interstitial ad:

```
AdManager.Instance.ShowInterstitial();
```

There're also similar methods for rewarded ads:

```
AdManager.Instance.LoadRewardedAd();
AdManager.Instance.IsRewardedAdReady();
AdManager.Instance.ShowRewardedAd();
```

All public methods inside AdManager are fully commented, please take a look at the class source code for more information.

4.6 Custom Leaderboards and Achievements

If you're not comfortable with the existing leaderboards and achievements, you can create new ones that suit your needs. There're a few things you need to do:

- Create new leaderboards, achievements and enter their names & ids to

appropriate fields in **GameServiceManager**.

- Repeat the **Configure Google Play Game Services for Android** step above with the new resources obtained from Google Play Developer Console.
- Disable or remove the **ScoreReporter** and **AchievementUnlocker** objects from the hierarchy.
- Write your own script to report scores and unlock achievements when appropriate.

The GameServiceManager class provides you with easy-to-use methods to submit scores and unlock achievements. You can access these methods via its public member **Instance**.

To show leaderboard UI or achievements UI:

```
GameServiceManager.Instance.ShowLeaderboardUI();
GameServiceManager.Instance.ShowAchievementsUI();
```

To report scores:

```
GameServiceManager.Instance.ReportScore(score, leaderboardName);
```

To unlock an achievement:

```
GameServiceManager.Instance.UnlockAchievement(achievementName);
```

THANK YOU FOR BUYING OUR ASSET!

GOOD LUCK WITH YOUR GAMES!