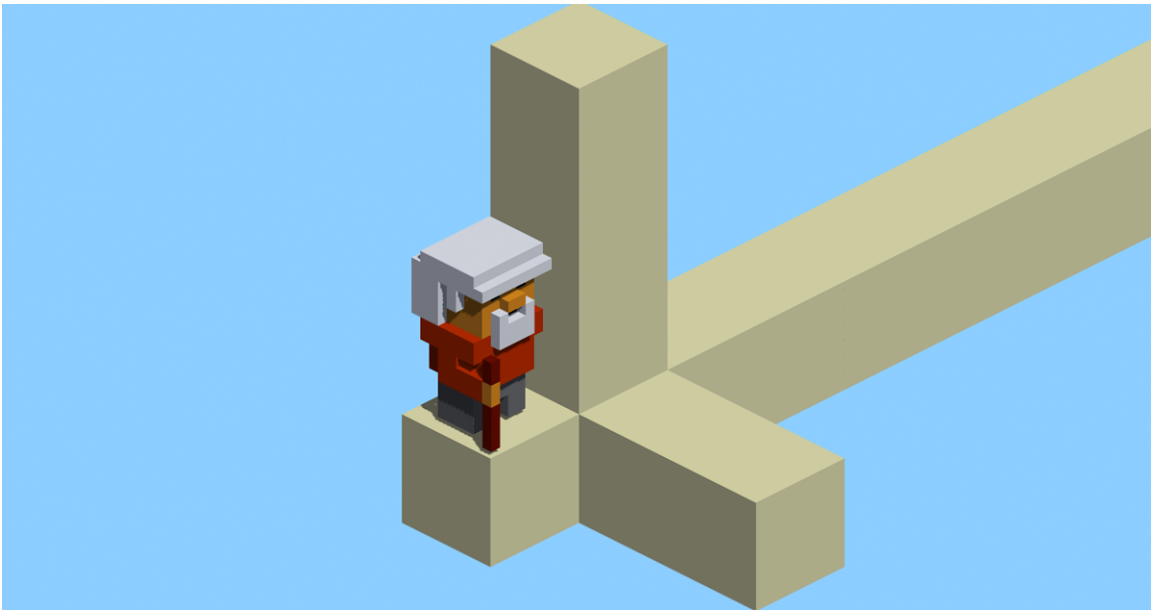# BRANCHES TEMPLATE

USER GUIDE

We strive to provide the best service as we can, if you have any questions or suggestions, please contact us!
Thank you!

SgLib Games

Table of Contents

1   INTRODUCTION

Branches is a simple yet very fun and addictive game. Tap to rotate the trunk, don't get hit by the branches or lagged behind. Collect golds and unlock new characters. One mistake and you die! Simple as that!

Branches is ready for release out-of-the-box. Everything just works. It is also flexible and customizable. Some highlights:

- Addictive one-touch gameplay
- Daily reward system for better retention
- 30 built-in unlock-able characters with cute blocky style
- Free-to-use assets (fonts, sounds, music, model, etc.)
- Optimized for mobile

Most importantly, this template includes **Easy Mobile** plugin, making it a truly fully-featured game that is release-ready. Easy Mobile is a comprehensive, cross-platform package that provides most of desired features of mobile games:

- Support for AdMob, Chartboost, Heyzap (with mediation) and UnityAds
- In-app purchasing
- Support for Game Center (iOS) and Google Play Games Services (Android) for leaderboards and achievements
- Sharing to social networks
- Push notifications using OneSignal

2 GETTING STARTED

2.1 Enter 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 Rate Us button and opening Facebook or Twitter page.
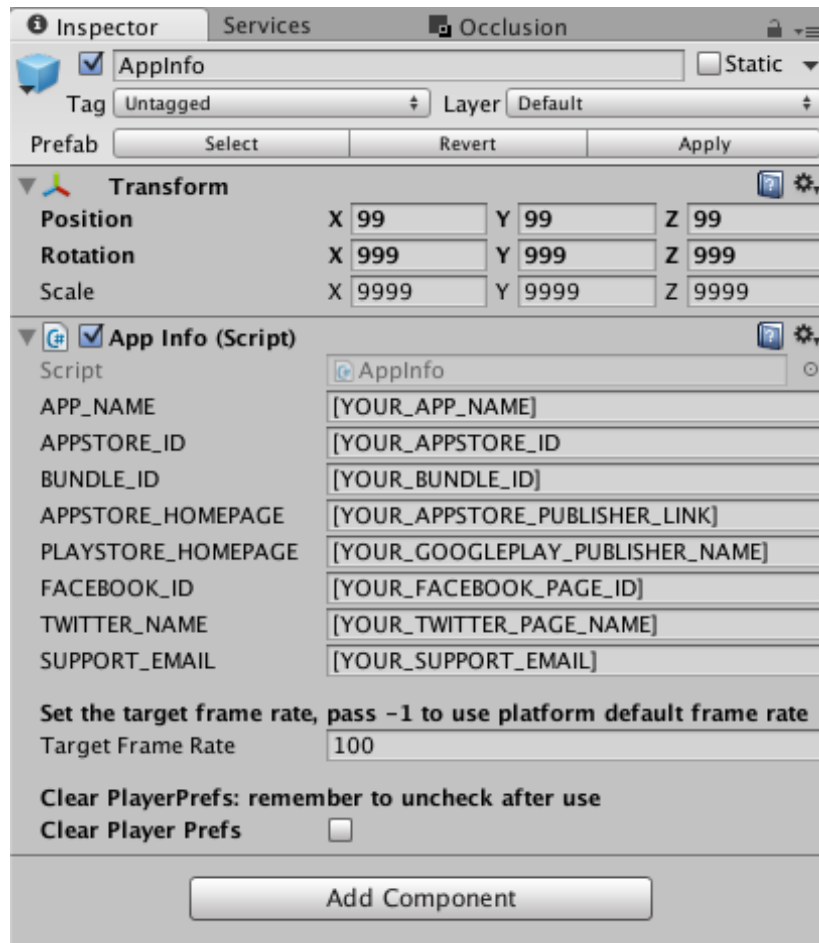
Besides basic app info, there're 2 other important variables:

- *TargetFrameRate*: set the target frame rate for the game, which should be at least 60fps for smooth motion.

- *ClearPlayerPrefs*: a convenient variable used for debugging purpose, check this and run the game will clear the *PlayerPrefs* (using *PlayerPrefes.DeleteAll()* method). It only works in the editor but it's worth remembering to uncheck the variable after use to avoid later confusion.

### 2.2 Link the game to your Unity project

When developing this template, we normally need to link it to our own Unity project

for testing, therefore you may need to unlink it from our project and link it to your own one, if you're going to use Unity services (e.g. if you want to enable premium features of this template, you'll need to use Unity IAP service). To unlink the project:

- Select Window -> Unity Services
- Select SETTINGS tab
- Click Unlink Project button
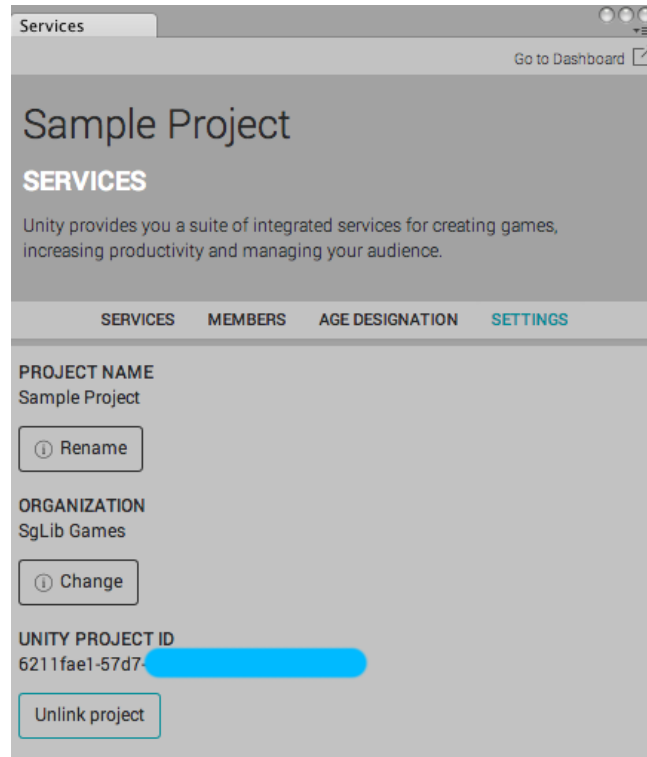
Now you can create a new project for the game.
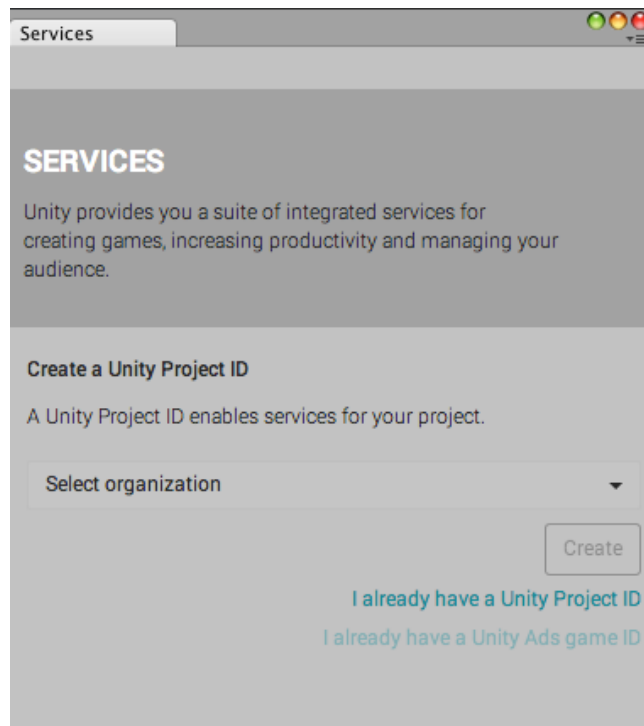
Now you game is linked to your own Unity project and is ready to use Unity services.

2.3 Testing Note

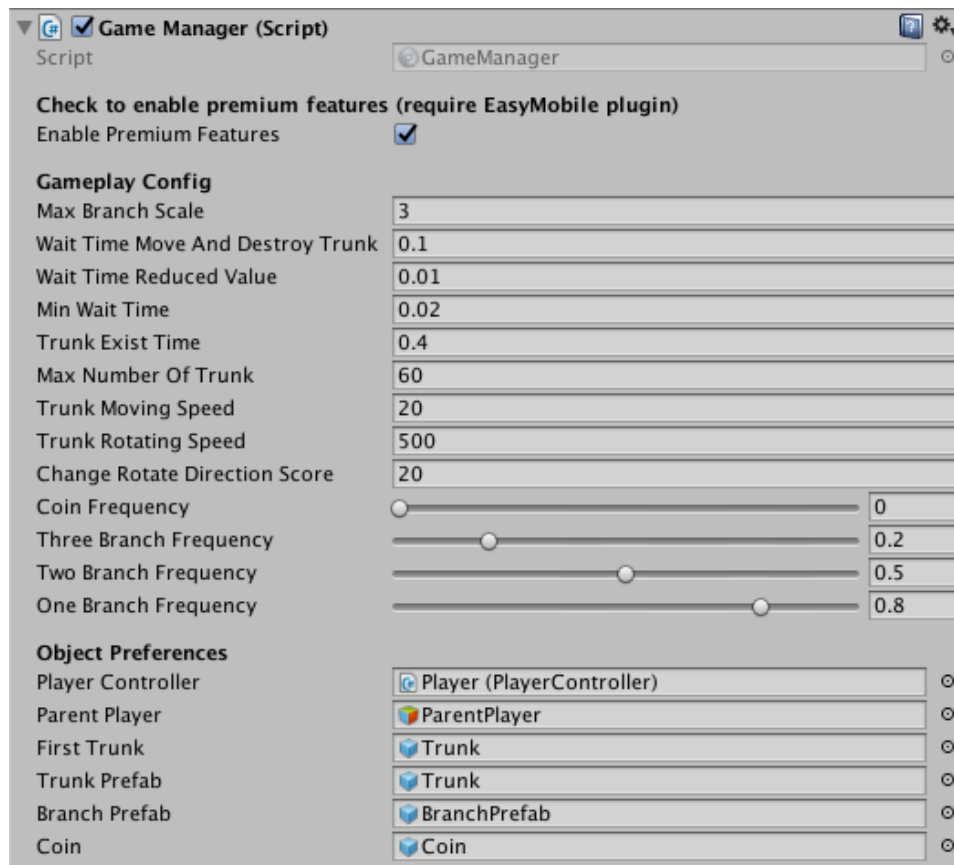There are 2 scenes in this game, it should be run from scene *Main*.

3 TEMPLATE CUSTOMIZATION

3.1 Gameplay tweaking

Most of important gameplay parameters can be configured in 2 components: GameManager and ParentPlayerController.

3.1.1 GameManager

The GameManager component is attached to a game object also named GameManager in the hierarchy.

You can tweak the gameplay by modifying following variables:

- *MaxBranchScale*: the maximum length of branch's scale when branch is visible. For Example: if you change this value to 4, the branch will scale from 1 to 4 when it visible (default scale of branch is 1).

- *WaitTimeMoveAndDestroyTrunk*: when the player go through the trunk, the trunk will be moved back. This value mean how long you must wait since a trunk is moved until next trunk is moved.

- *WaitTimeReducedValue*: with the same score variable in ParentPlayerController script, when you reached this score, the variable TimeToWaitToMoveAndDestroyTrunk will be minus by DecreasingTimeToWaitToMoveAndDestroyTrunk. That mean the trunk will be moved and destroyed faster.

- *MinWaitTime*: when game over the value TimeToWaitToMoveAndDestroyTrunk will be equal with this value.

- *TrunkExistTime*: that mean how long the trunk exist since the player go through of it until it is destroyed.

- *MaxNumberOfTrunk*: how many trunk on scene

- *TrunkMovingSpeed*: that mean how fast the trunk moving back when player through of it.

- *TrunkRotatingSpeed*: that mean how fast the trunk rotate.

- *ChangeRotateDirectionScore*: when you reached this score, the trunk will change rotate direction (rotate direction only left or right).

- *GoldFrequency*: probability to create gold.

- *ThreeBranchFrequency*: probability to create three branch at a node.

- *TwoBranchFrequency*: probability to create two branch at a node.

- *OneBranchFrequency*: probability to create one branch at a node.

3.1.2 *ParentPlayerController*

The ParentPlayerController is attached to an object named ParentPlayer in the hierarchy.
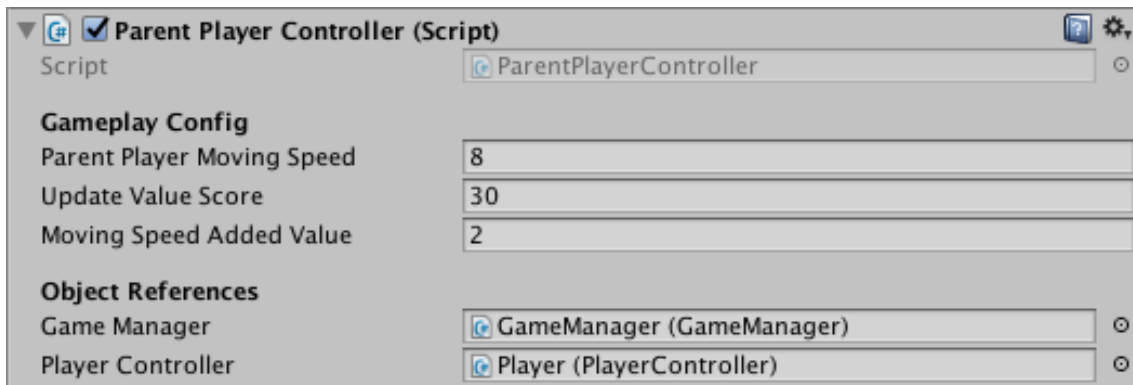

You can adjust following parameters:

- *ParentPlayerMovingSpeed*: how fast parent player moving.

- *UpdateValueScore*: that mean when you reached this score, the value Moving Speed Of Parent Player in this script and RotatingSpeedOfTrunk in PlayerController script will be increase.

- *MovingSpeedAddedValue* Score: when you reached the value ScoreToIncreaseMovingSpeedAndRotating Speed, value MovingSpeedOfParentPlayer in this script will be plus with this value. That mean player will run faster.

- *RotatingSpeedAddedValue*: when you reached the value ScoreToIncreaseMovin SpeedAndRotatingSpeed, value RotatingSpeedOfTrunk will be plus with this value. That mean the trunk will be rotate faster.

3.2 Daily reward feature

This template has a built-in daily reward system in which the user will be rewarded with coins every predefined interval of time. This is an effective way to increase user engagement and retention for your game. You can configure this feature from the *DailyRewardController* object in the hierarchy.
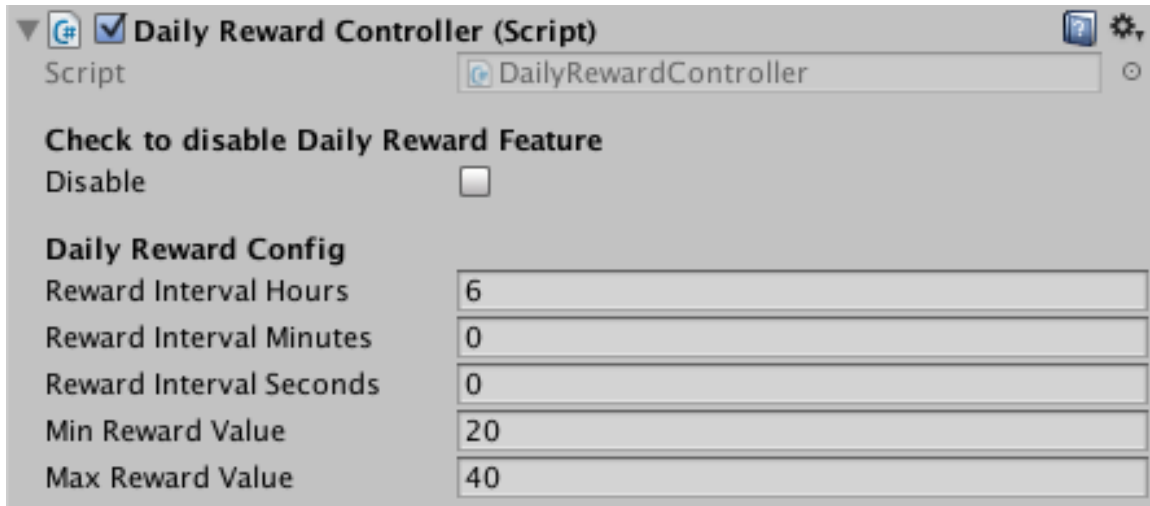

- *Disable*: check to disable this feature
- *Reward Interval Hours, Minutes and Seconds*: the amount of time until the next reward
- *Min Reward Value* & *Max Reward Value*: the actual rewarded coins will be randomized between these two values

3.3 Adding more characters

Out-of-the-box, this game is already packed with 30 characters, cute and ready to use! If you want to add more, follow these simple steps:

a. Create a character model with the pivot at the bottom center.

b. Navigate to *Assets/Prefabs/Characters/CharacterPrefabs* and duplicate one of the available character prefabs.

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

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

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

f. Enter the character name and price to the *Character* component. Check the *isFree* option if you want to give out this character for free (it will be automatically unlocked). **Important:** *the new character's name must not repeat any existing character name.*

g. Resize the character array in *CharacterManager* game object then drag the

new character to it and hit Apply to save changes to its prefab.
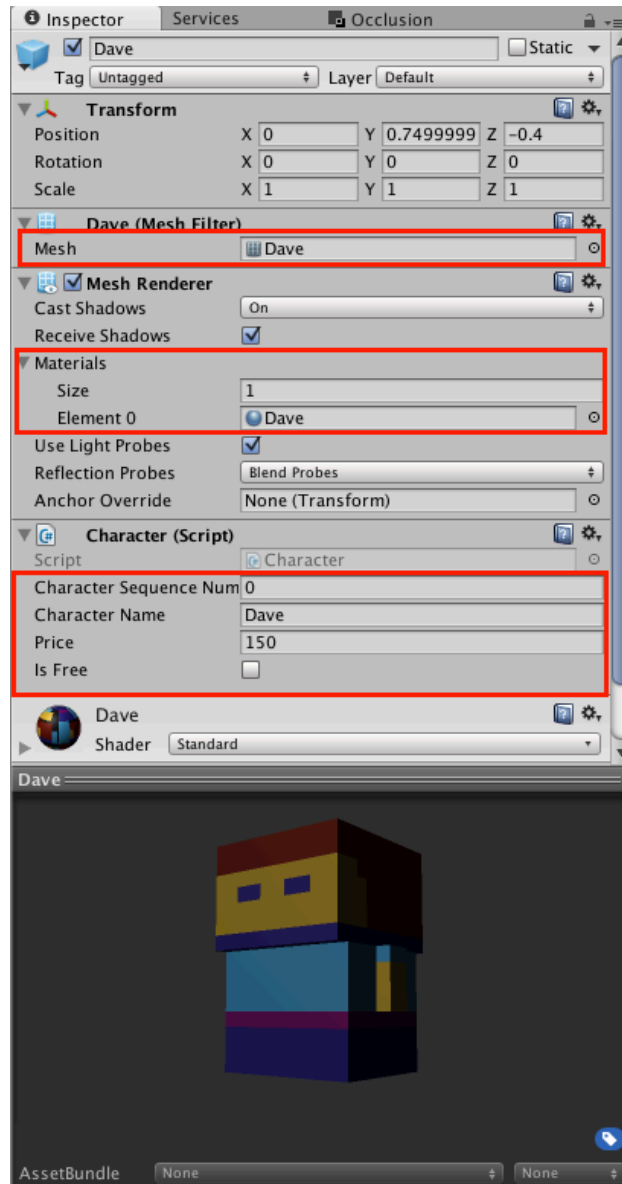
Now the new character has been added and ready to use in game! You will see it listed in the *CharacterSelelection* scene.

3.4 Changing background gradient

The gradient sky background in this game is implemented by a gradient shader call *VerticalGradientSkybox* located in the *Shaders* folder. It's an easy-to-use vertext/fragment shader. To modify the background gradient colors:

▪ Navigate to game object *CameraBackground* under *MainCamera*.

▪ Modify the gradient colors: *ColorTop*, *ColorBottom* and adjust the
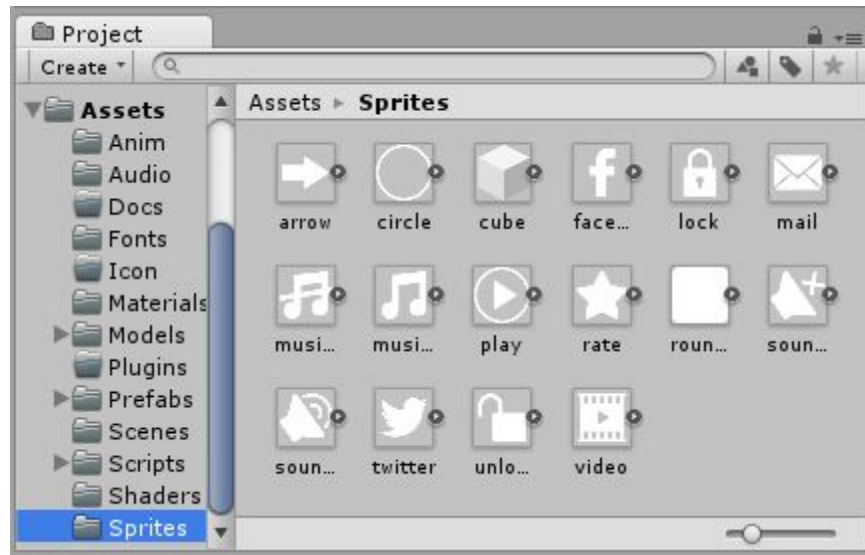
GradientBias until you're satisfied with the result.

- Hit *Apply* to save changes.
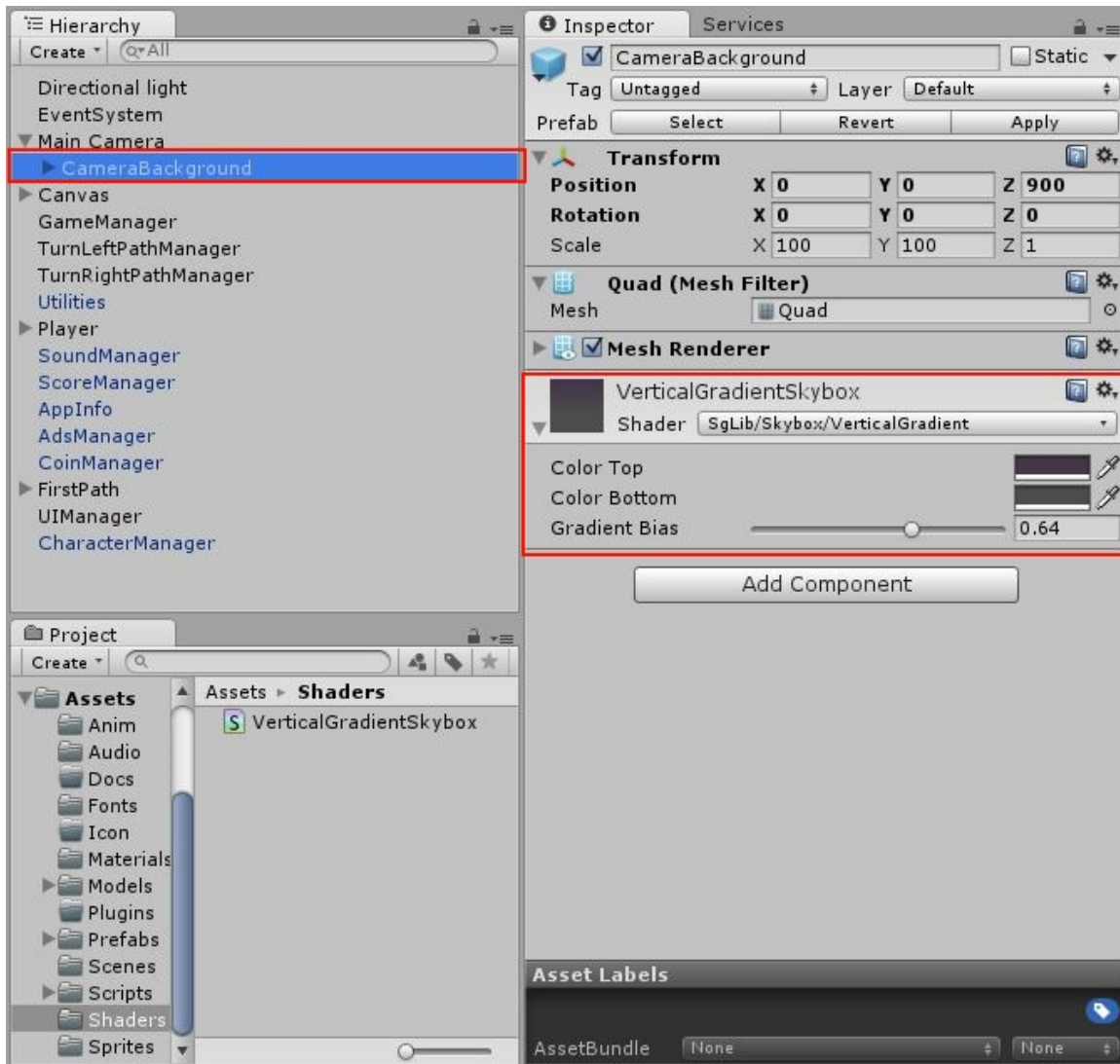
3.5 Customizing UI

All sprites used in this game (for buttons and other UI components) are located under the *Sprites* folder. 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 the *Fonts* folder together with appropriate license files.
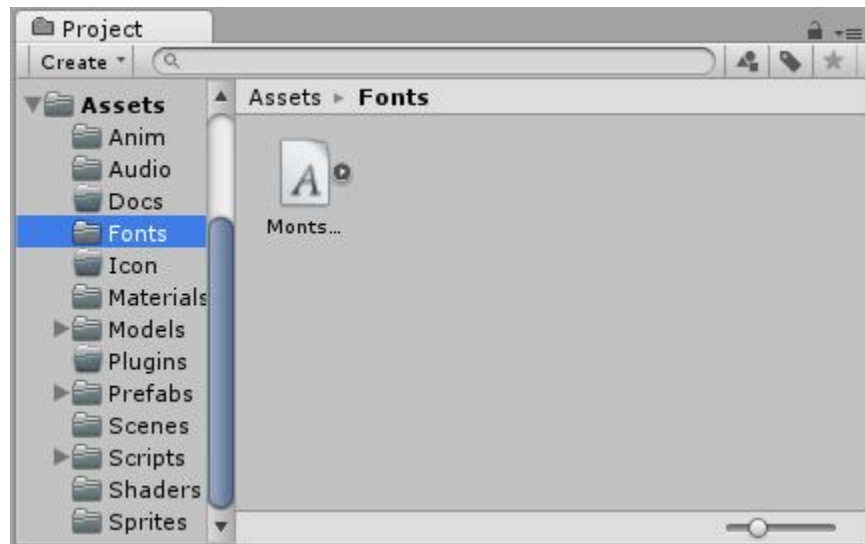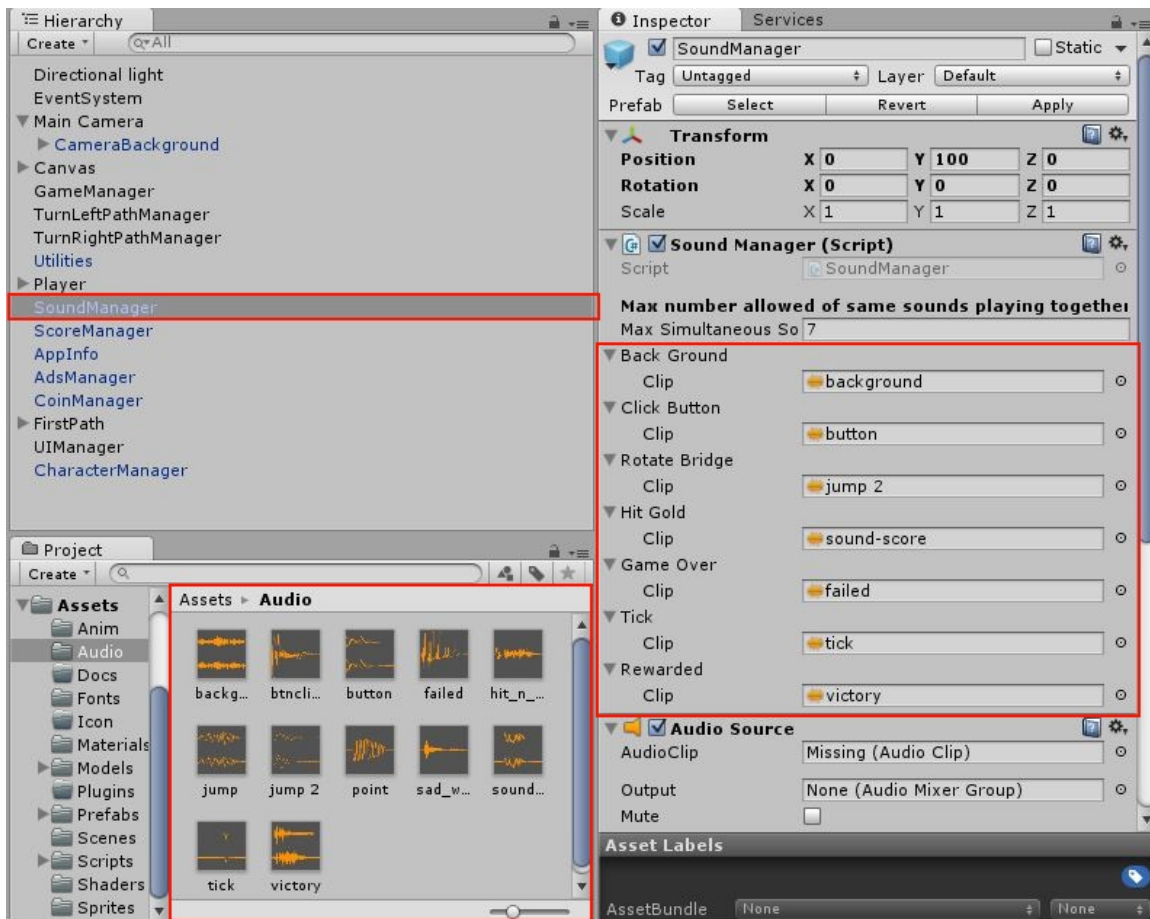

3.6 Sounds

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

This game features a *SoundManager* class to manage activities in game like playing music or mute/unmute sounds. If you want to replace sounds in this game, simply drag and drop new sounds to appropriate slots in the *SoundManager* component.

# 4   ENABLING PREMIUM FEATURES

Premium features include:

- Advertising
- In-app purchasing
- Leaderboards & achievements
- Sharing
- Push notifications

This section provides a guide on configuring these features for your game. As the features are implemented using EasyMobile plugin, this guide is intended to be used alongside EasyMobile's official user guide, which can be found at https://sglibgames.gitbooks.io/easy-mobile-user-guide/content/.

4.1   Before You Begin

- Make sure the *EnablePremiumFeatures* option in the *GameManager* object is checked.
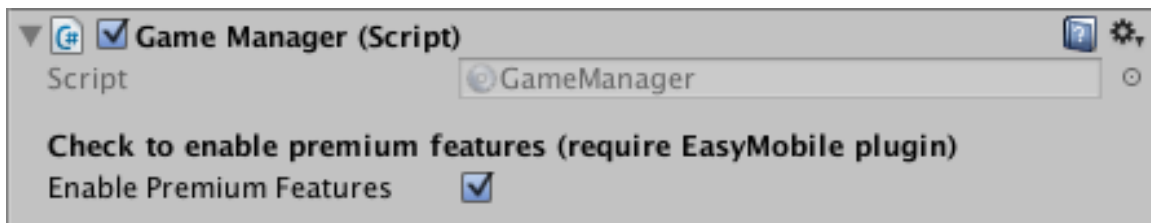

- Make sure there's an EasyMobile object in the Main scene, if not you can add it using the EasyMobile prefab at folder *Assets/EasyMobile*. It is necessary for the plugin to function properly.
- The settings interface of Easy Mobile can be opened via menu *Window > Easy Mobile > Settings*, this is the only place to go to configure this plugin.
- Note that you won't need to write a single line of integration code for Easy Mobile to work, as the integration was done beforehand, you only need to configure the plugin in the editor (that means you can safely ignore all the Scripting sections in Easy Mobile user guide).

4.2   Advertising

4.2.1   Template-specific setup

This template contains a game object named *AdDisplayer* which is responsible for all ads displaying activities in the game. There you can configure how ads should be served in your game.
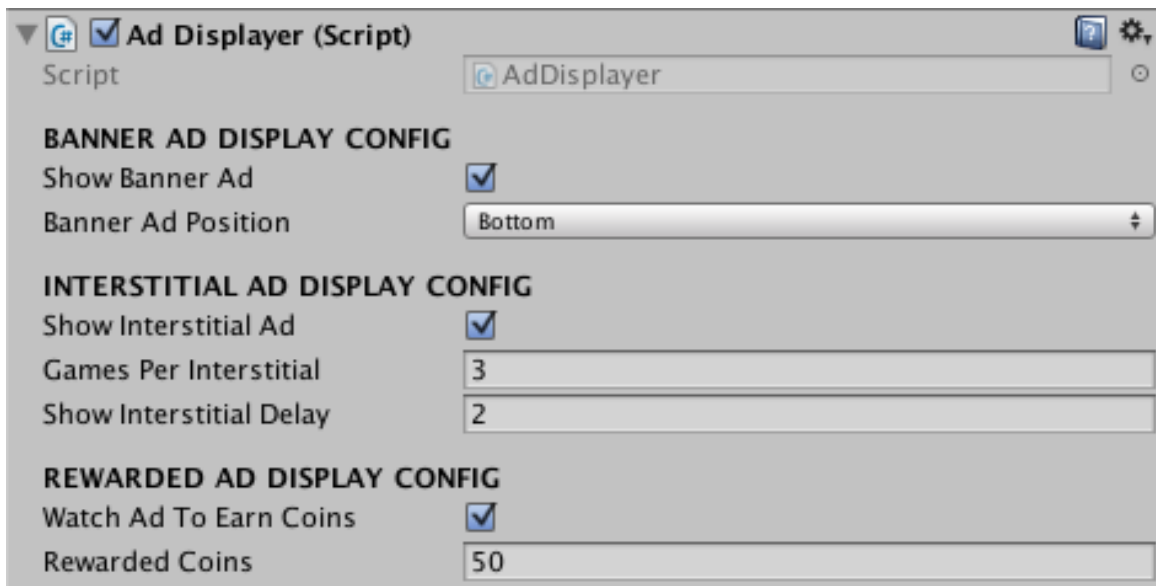
Banner ads are configured in the **BANNER AD DISPLAY CONFIG** section.

- *Show Banner Ad*: whether to show a banner ad in game
- *Banner Ad Position*: which position the banner should be placed

Interstitial ads are configured in the **INTERSTITIAL AD DISPLAY CONFIG** section.

- *Show interstitial ad*: whether to show interstitial ads when game over
- *Games Per Interstitial*: how many games to be played before showing ad
- *Show Interstitial Delay*: how many seconds after game over that ad is shown

Rewarded ads are configured in the **REWARDED AD DISPLAY CONFIG** section.

- *Watch Ad To Earn Coins*: whether to allow the user to watch an ad to earn extra coins
- *Rewarded Coins*: how many coins should be awarded after watching an ad

4.2.2 Easy Mobile setup

With Easy Mobile's Advertising module, you'll have support for AdMob, Chartboost, Heyzap (with mediation) and Unity Ads. You can use multiple ad networks at once and have different configurations for iOS and Android. For example, you can use AdMob for banner ads, Chartboost for interstitial ads and Unity Ads for rewarded ads on iOS, and yet another combination on Android.

To configure the Advertising module, open Easy Mobile settings interface and select the Advertising tab. Below is the settings interface of the module.

You can setup the module in just a few steps as below. Please see the Advertising section in Easy Mobile's user guide for detailed instructions on each step.

    a. Setup the ad networks you want to use, this includes importing the required plugins for each network, please see Easy Mobile user guide for more information

    b. Enable auto ad-loading feature: simply leave the *Auto-Load Default Ads* option as checked and other parameters as default, the plugin will automatically load ads in the background

    c. Select default ad networks for each platform: choose your preferred network for each type of ad on each platform

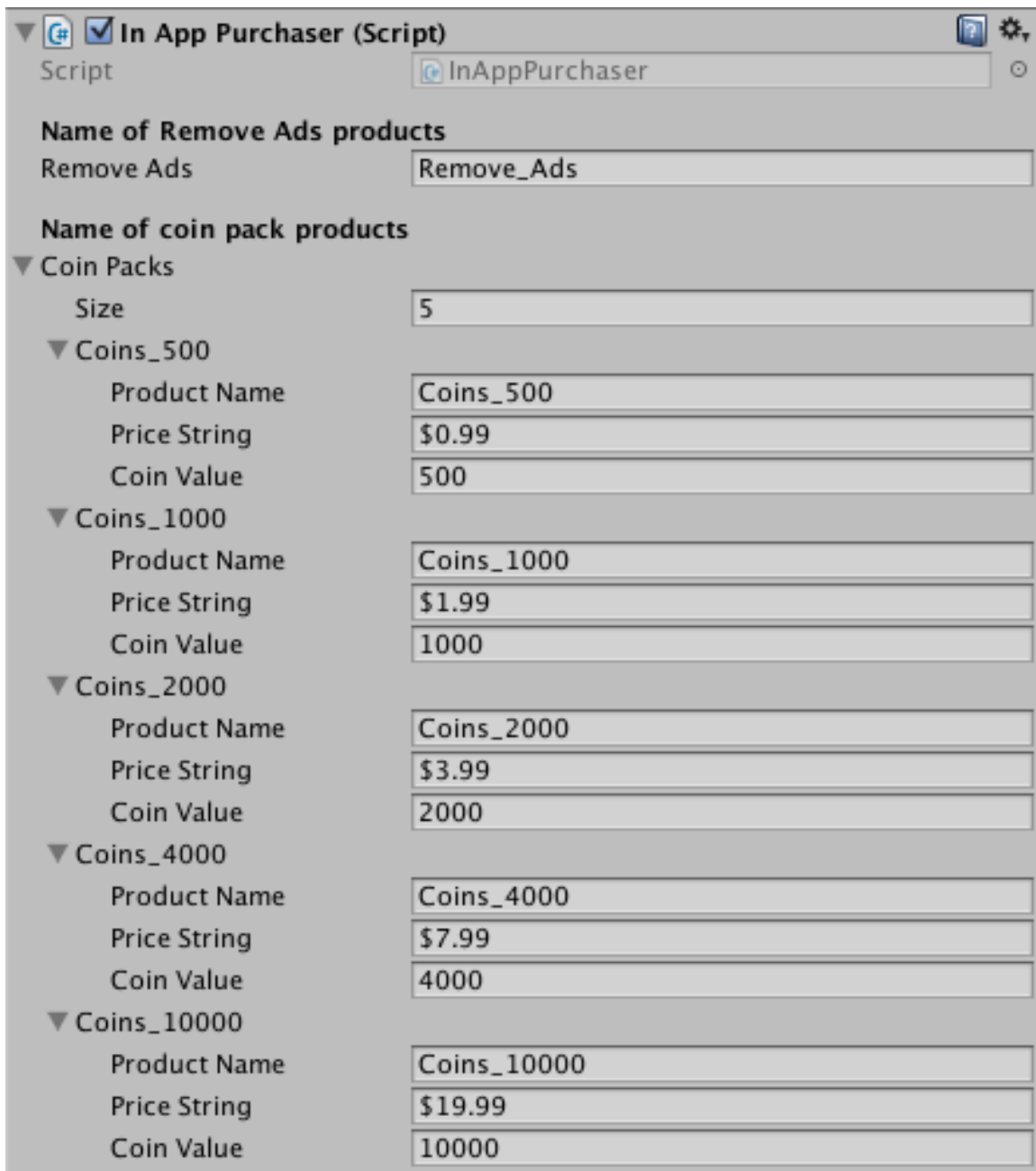That's it! Now your game is ready for showing ads!

4.3 In-App Purchasing

4.3.1 Template-specific setup

The built-in in-app purchases of this template include a *Remove Ads* button, and several coin packs. You can modify existing products and add more coin packs if you like. There's also one *Restore Purchase* button as required on iOS.
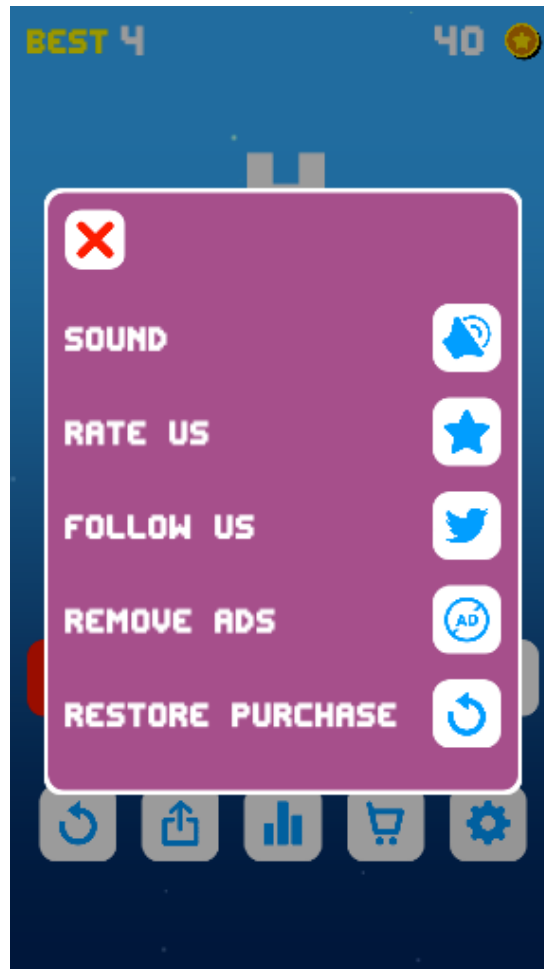
In the hierarchy, there's a game object named *InAppPurchaser* which manages all the in-app purchasing activities in this game.

In App Purchaser (Script)

Script	InAppPurchaser

Name of Remove Ads products

Remove Ads	Remove_Ads

Name of coin pack products

▼ Coin Packs

Size	5

▼ Coins_500

Product Name	Coins_500
Price String	$0.99
Coin Value	500

▼ Coins_1000

Product Name	Coins_1000
Price String	$1.99
Coin Value	1000

▼ Coins_2000

Product Name	Coins_2000
Price String	$3.99
Coin Value	2000

▼ Coins_4000

Product Name	Coins_4000
Price String	$7.99
Coin Value	4000

▼ Coins_10000

Product Name	Coins_10000
Price String	$19.99
Coin Value	10000

Here you can modify the product definitions including the displayed name, price or coin value of the coin packs. To add more coin packs, simply increase the *CoinPacks* array size and enter necessary information for your new packs. The built-in store UI will automatically update to your changes in the product list without you having to do anything.
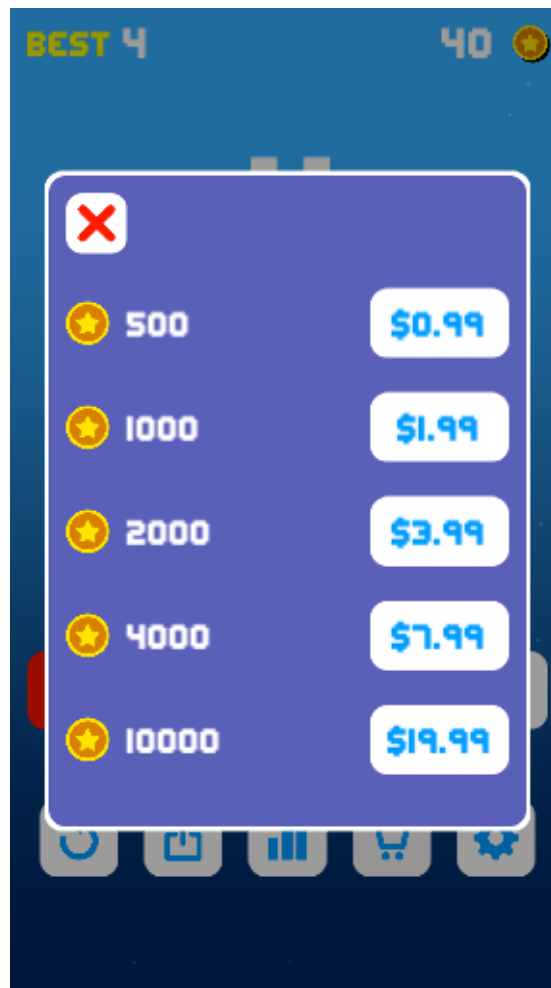
4.3.2 Easy Mobile setup

Setting up the In-App Purchasing module of Easy Mobile includes the following steps. Please see the In-App Purchasing section in Easy Mobile's user guide for detailed instructions on each step.

    a. Enable Unity In-App Purchasing service
    b. Select target store if you're on Android
    c. Enable receipt validation if you wish
    d. Declare the products

Below is the settings interface of the In-App Purchasing module of Easy Mobile.

IN-APP PURCHASING

[ANDROID] TARGET STORE

Target Android Store   Google Play

RECEIPT VALIDATION

Unity IAP offers local receipt validation for extra security. Apple stores and Google Play store only.

Validate Apple Receipt

Validate Google Play Receipt

⚠️ Please go to Window > Unity IAP > IAP Receipt Validation Obfuscator and create obfuscated secrets to enable receipt validation for Apple stores and Google Play store. Note that you don't need to provide a Google Play public key if you're only targeting Apple stores.

PRODUCTS

▶ 6 Products

Add New Product

CONSTANTS CLASS GENERATION

Generate the static class EasyMobile.EM_IAPConstants that contains the constants of product names. Remember to regenerate if you make changes to these names.

Generate Constants Class

Note that the products declared with Easy Mobile must have names that match with the ones you have in the aforementioned *InAppPurchaser* object. Also note that *Remove Ads* is a non-consumable product, while the coin packs must be consumable.

4.3.3 Create the products for targeted stores

That last step in configuring the in-app purchasing feature is to create products for your targeted stores (e.g. Google Play and Apple App Store). Make sure the product ID, product type and price match the ones you have in your game.

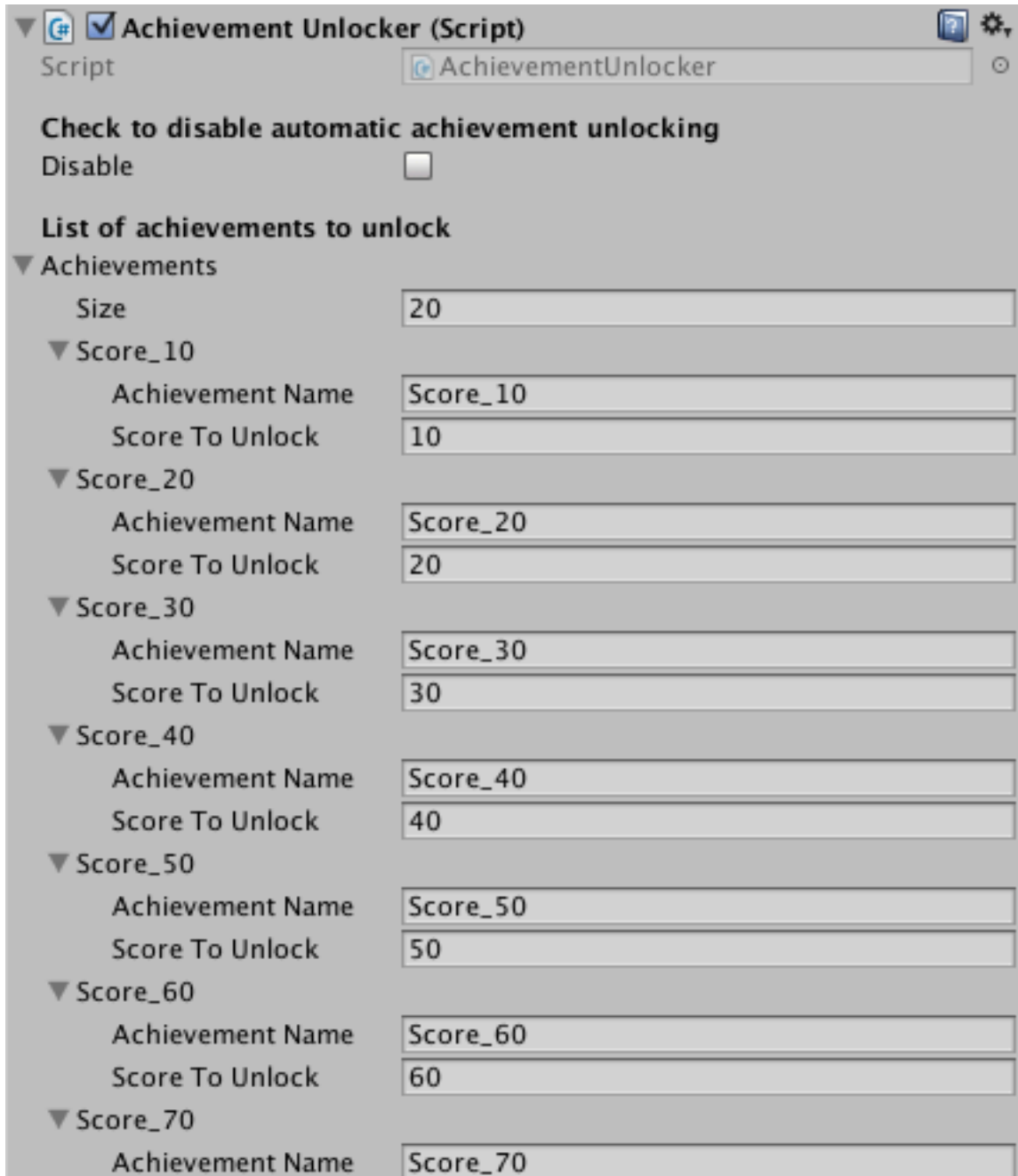## 4.4 Game Service

4.4.1 Template-specific setup

This template has a built-in leaderboard for ranking users' scores, and many achievements. It works with Game Center (iOS) and Google Play Game Services (Android).

User's score will be submitted automatically when game over by an object named *AutoScoreReporter*. There you can change the leaderboard name or even disable automatic score reporting altogether.

Achievements will be unlocked automatically when the user reaches a certain score. The achievement unlocking is handled by an object named

AutoAchievementUnlocker. From this object, you can modify existing achievements and add or remove achievements. You can also disable the automatic achievement unlocking feature if you wish.

4.4.2 Setup for your targeted stores

The next step is to create the required leaderboard and achievements for your targeted stores (i.e. in iTunes Connect for App Store and the Developer Console for Google Play). Take note of their IDs for use in the next step.

4.4.3 Easy Mobile setup

Setting up the Game Service module of Easy Mobile includes the following steps. Please see the Game Service section in Easy Mobile's user guide for detailed instructions on each step.

a. Setup Google Play Games if you're targeting Android
b. Enable the automatic initialization feature: just leave everything under the **AUTO-INIT CONFIG** section as default
c. Declare the leaderboards and achievements

Below is the settings interface of the Game Service module of Easy Mobile.


Note that you must declare the leaderboard and achievements with the same names as the ones you have in the *AutoScoreReporter* and

AutoAchievementUnlocker objects. Also their IDs must match the ones you created in iTunes Connect and Google Play Developer Console.

4.5 Native Sharing

This template has a Share button that allows the user to share the game's screenshot to social networks using the native sharing functionality. This activity is managed by the object named *ScreenshotSharer*.
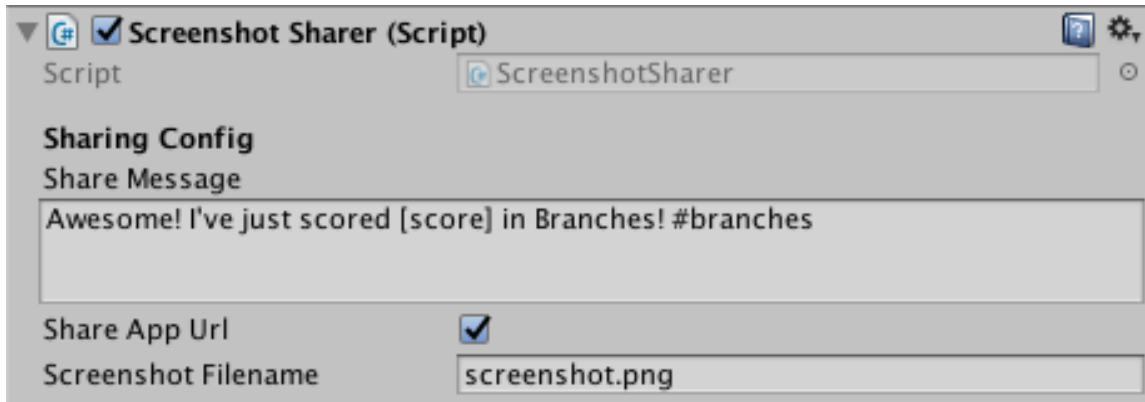
Here you can configure the sharing feature.

- *Share Message*: the default sharing message, note that [score] will be automatically replaced by actual score
- *Screenshot Filename*: filename to store the screenshot in the device storage

Note that you need to enable the *external write permission* for this feature to function properly on Android. Please see the Native Sharing section in Easy Mobile user guide for detailed instructions on doing that.

4.6 Push notifications

Enabling push notifications for your app using OneSignal service includes following steps. Please see the Notification section in Easy Mobile user guide for detailed instructions on each step.

- Open the Notification tab in Easy Mobile's settings interface
- Import OneSignal plugin
- Prepare your app for push notifications, e.g. enable the Push Notification capability for the provisioning profile on iOS (please see Easy Mobile user guide as well as OneSignal documentation for detailed instructions).
- Add your app to OneSignal dashboard
- Enter your app ID to Easy Mobile settings in Unity

Below is the settings interface of the Notification module of Easy Mobile where you can enter your app ID and Google project number.

That's it! You've just finished implemented premium features for your game!

THANK YOU AND GOOD LUCK WITH YOUR GAMES!