SHOOTY CLOCKS 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

Aim the clock hand accurately at another clock and shoot it down! It looks easy, but it's not. **Shooty Clocks** is a simple, minimalist game that is surprisingly engaging and addictive. With two modes: endless mode (survival) and quest mode with 100 ready levels, this game will keep the player entertained for hours.

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

- Minimalist game with engaging gameplay
- **Two game modes: endless and quest**
- ***100 built-in levels***
- ***Easy-to-use level editor for creating new levels and editing existing levels***
- Free-to-use assets (fonts, sounds, music, sprites, etc.)
- Minimalist design
- Optimized for mobile

Most importantly, this template is *pre-integrated* with **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 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 notification using OneSignal service

Being *pre-integrated* means this template is already configured to work with Easy Mobile. All you need is import Easy Mobile and do a few setup steps, and have all the above features readily implemented. You don't even have to write a single line of integration code!

** This package doesn't include Easy Mobile*

** The use of Easy Mobile is totally optional: as long as it's not imported, all the integration code will automatically be excluded from compilation, so that no impact will be made on the game, which is fully functioning on its own.*

2 GETTING STARTED

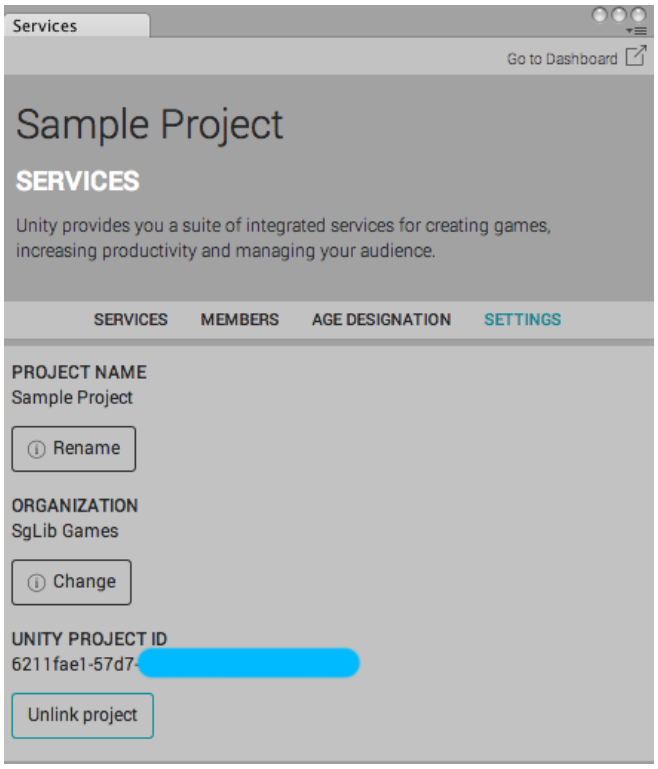
2.1 Enter app information

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.

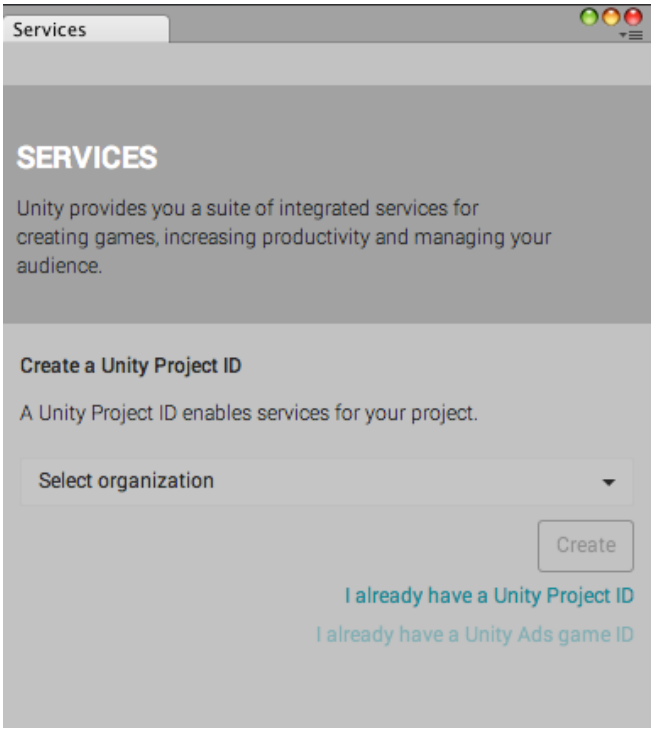
2.2 Link the game to your Unity project

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 and link it to the game.



Now you game is linked to your own Unity project and is ready to use Unity services.

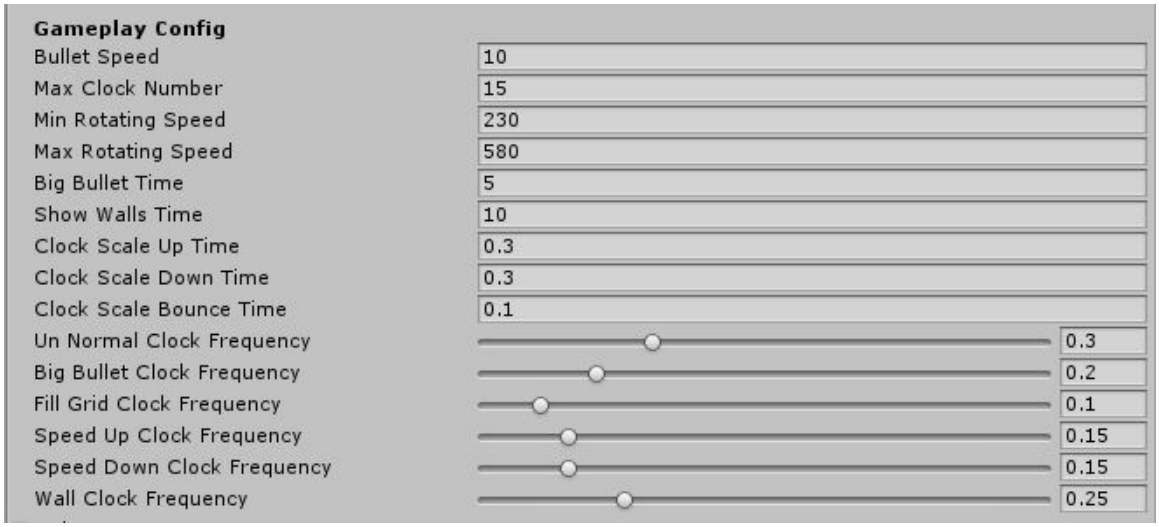
2.3 Testing & Building Note

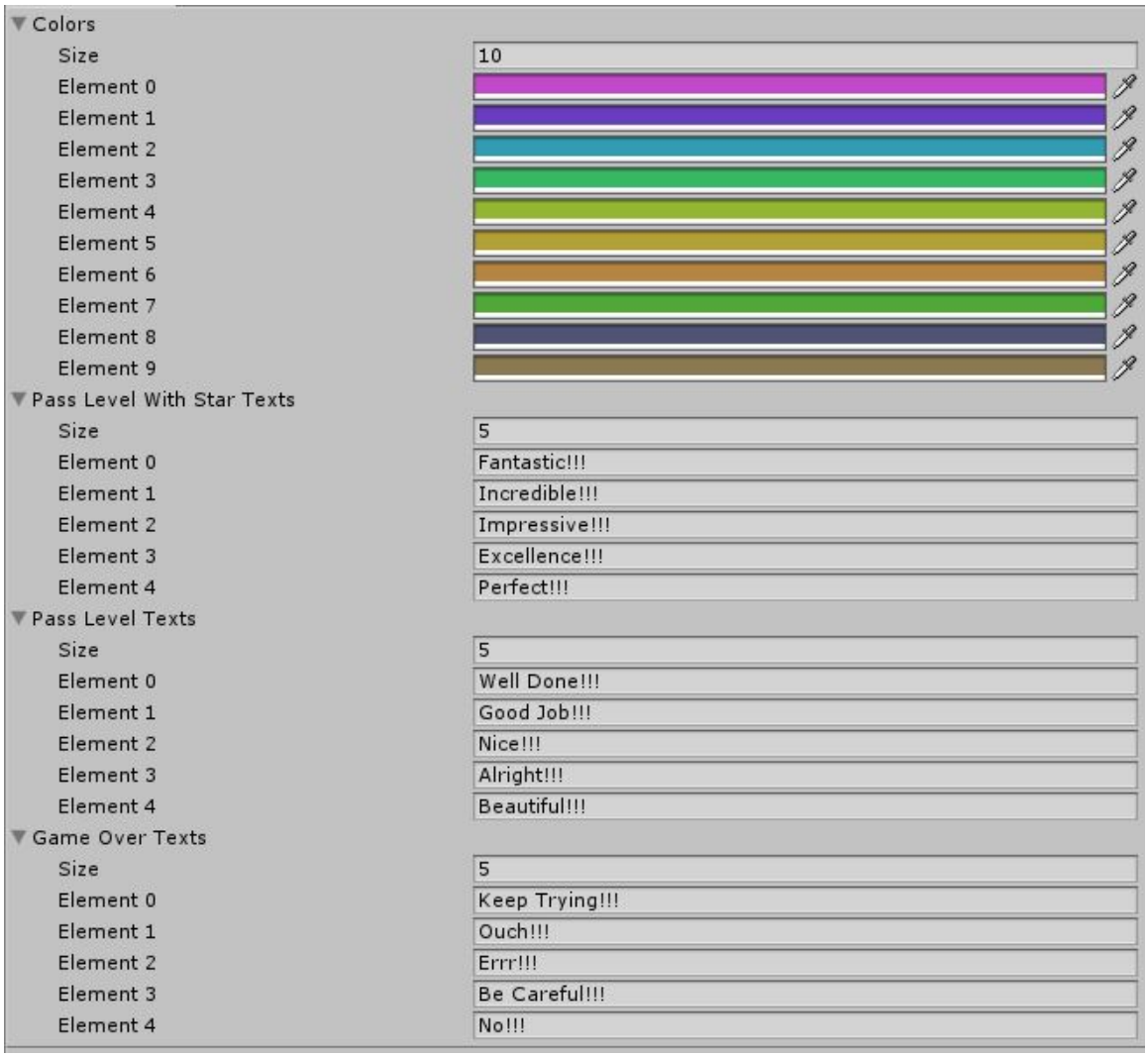
The game should be run from scene *FirstScene*. Additionally, the *LevelEditor* scene is meant to run in the editor only and should not be included in the build settings.

3 TEMPLATE CUSTOMIZATION

3.1 Gameplay tweaking

Most of important gameplay parameters can be configured within the *GameManager* component which is attached to a game object also named *GameManager* in the hierarchy of *GameScene*.





You can tweak the gameplay by modifying following variables:

- *BulletSpeed*: the speed of the bullet
- *MaxClockNumber*: max number of the lock that we'll create on survival mode.
- *MaxRoratingSpeed* & *MinRotatingSpeed*: maximum & minimum rotating speed of the hand of the clock, the actual speed will be randomized between these two values.

  IMPORTANT: *If you modify these values, you need to go to Scripts/Gameplay/ClockController.cs and adjust the speed range of arrowRotatingSpeed variable to the above values.*

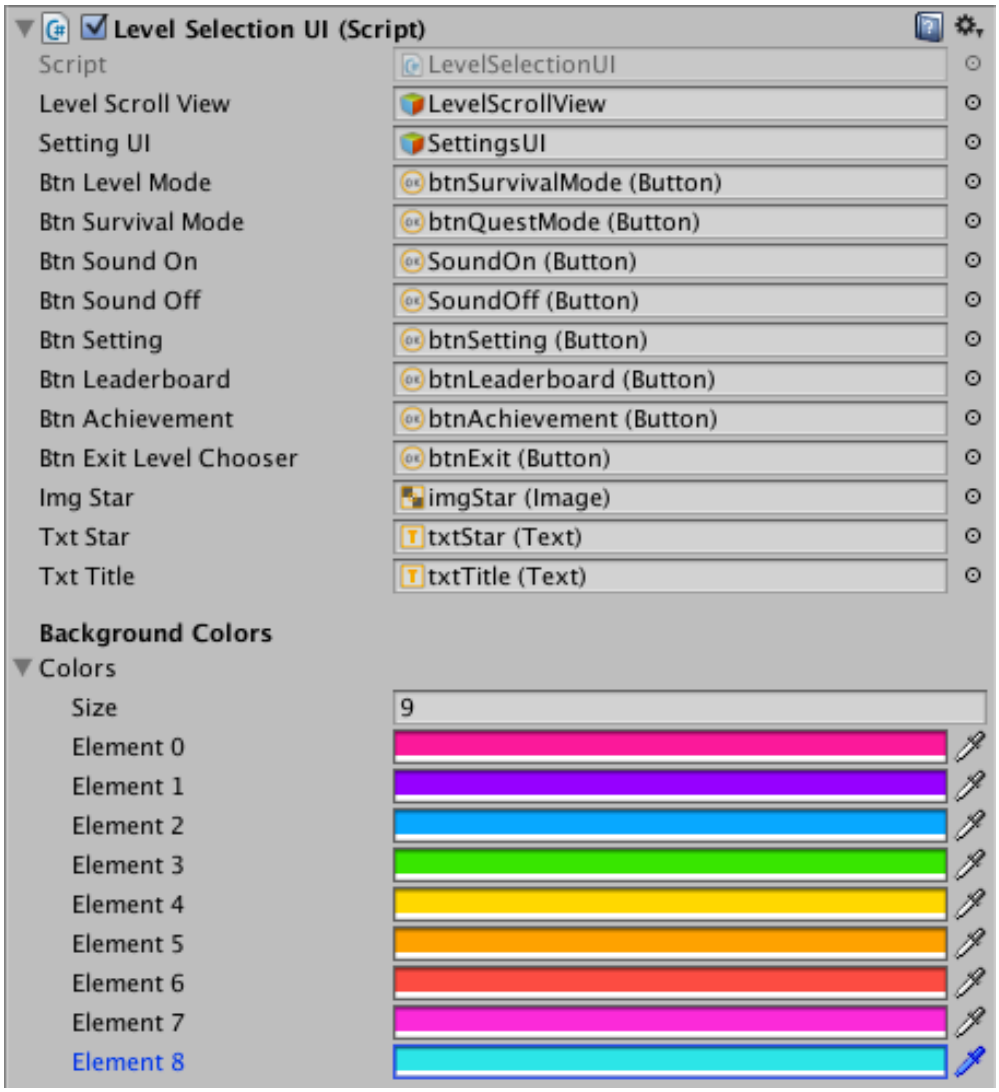
```
[Header("Clock Config")]
public ArrowRotatingDirection arrowRotatingDirection;
[Range(100,400)]
public float arrowRotatingSpeed;
public bool isShootingClock;
```

- *BigBulletTime*: the time that big bullet effect lasts.
- *ShowWallsTime*: how long the wall power-up lasts.
- *ClockScaleUpTime*: the time that clock scaling up.
- *ClockScaleDownTime*: the time that clock scaling down.
- *ClockScaleBounceTime*: the time that clock scaling up and down.
- *UnNormalClockFrequency*: appearance frequency of special-effect clocks.
- *BigBulletClockFrequency*: appearance frequency of big bullet clock.
- *FillGridClockFrequency*: appearance frequency of fill grid clock.
- *SpeedUpClockFrequency*: appearance frequency of speed up clock.
- *SpeedDownClockFrequency*: appearance frequency of speed down clock.
- *WallClockFrequency*: appearance frequency of wall-showing clock.
- *Colors*: the color array from which background color is randomized.
- *PassLevelWithStarTexts*: the text that showing when you pass level with star.
- *PassLevelTexts*: the text that showing when you pass level with no star.
- *GameOverTexts*: the text showing when game over.

3.2 Customizing UI

3.2.1 Background color

In *FirstScene*, there's an object named LevelSelectionUI where you can configure the background color of the scene. The actual color will be randomized from the Colors array, feel free to add more colors or modify the existing colors.

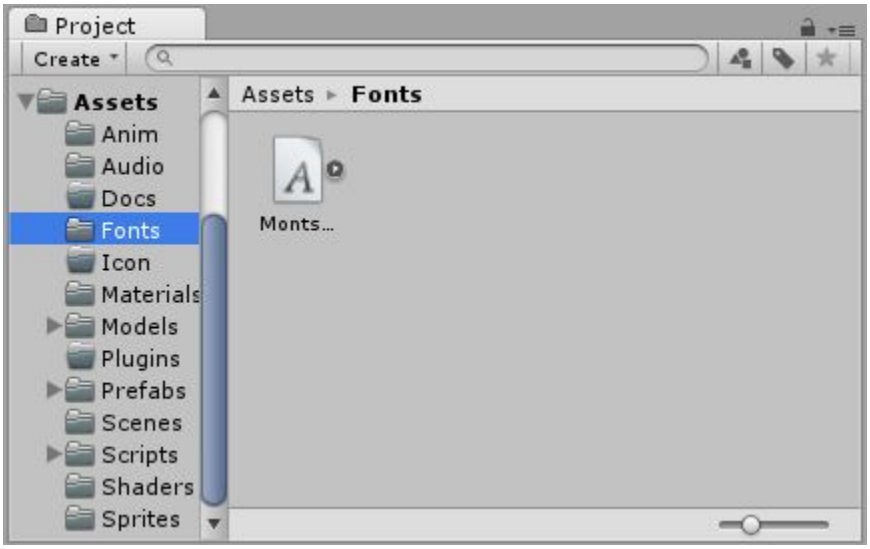


Likewise, the background color of *GameScene* can also be adjusted using the Colors array in the *GameManager* object as mentioned above.

3.2.2 Sprites

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

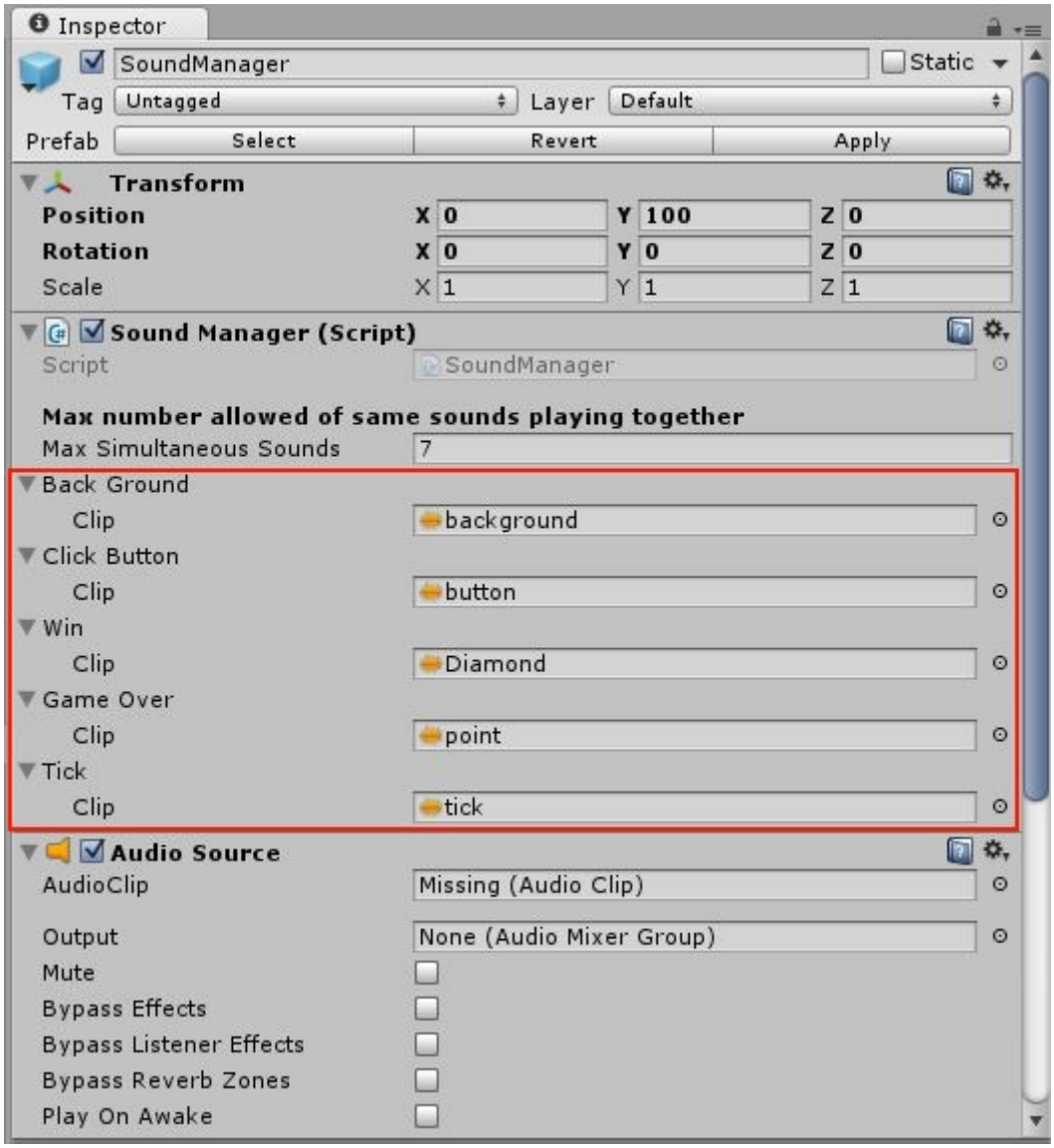
All fonts used in this game are free-to-use in commercial projects. Fonts are located under the *Fonts* folder together with appropriate license files.



3.3 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 LEVEL EDITOR

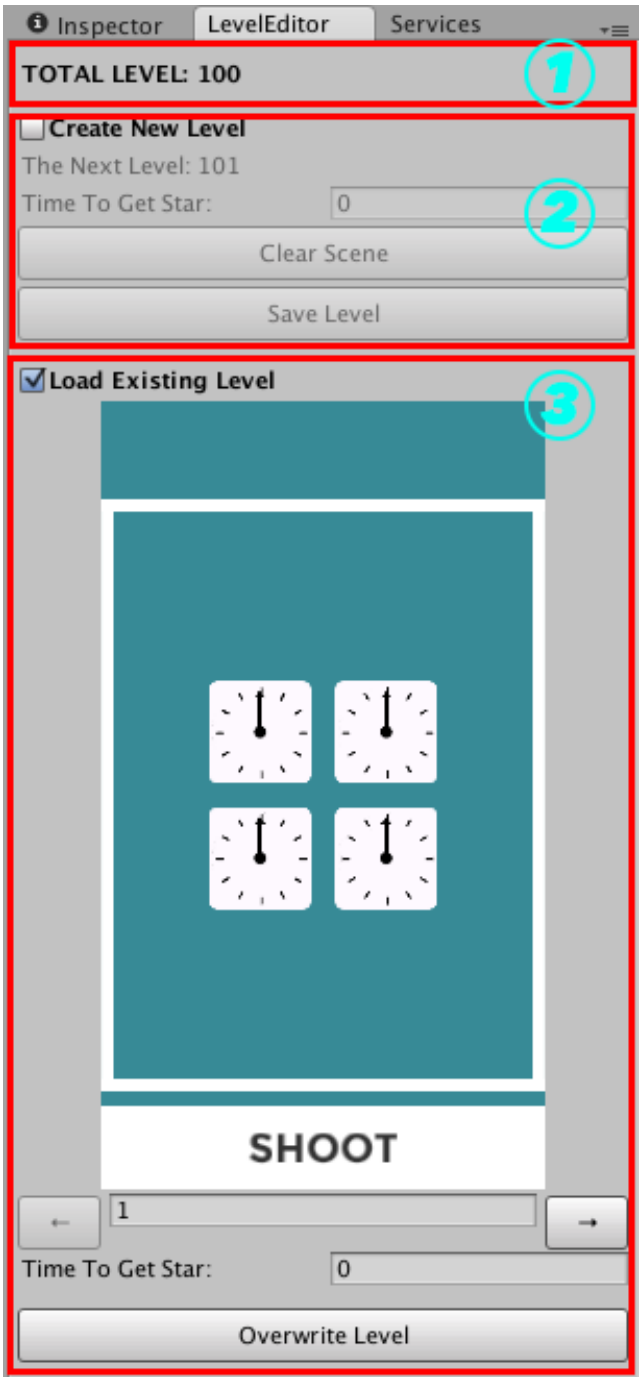
This template comes with a simple and easy-to-use level editor which enables you to create new levels, or edit the existing 100 built-in levels. This section will show you how to use this level editor.

If you prefer a video tutorial, we have one at https://youtu.be/eR2tuxrmt9Y

4.1 Overview

To access the level editor, go to menu *Tools/Level Editor*. Note that this will open the *LevelEditor* scene and close the current scene, so you should save this current scene before open the level editor, or all the unsaved changes will be lost.

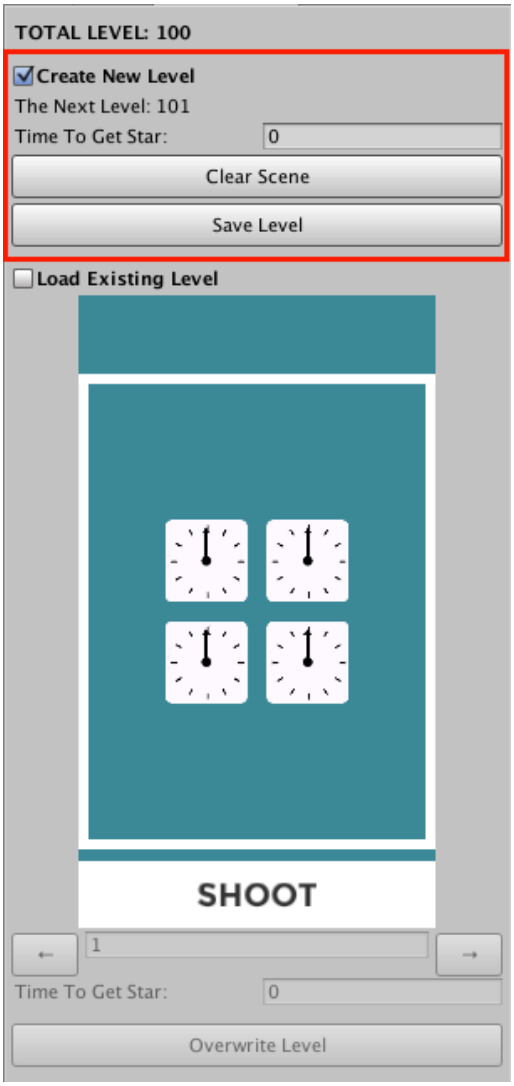
Below is the interface of the level editor.



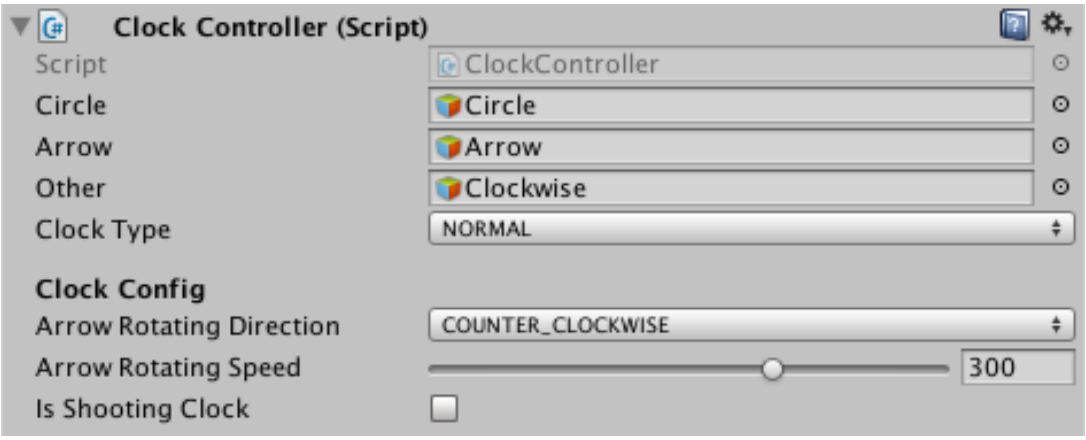
The level editor has three sections:

- ***Section 1 – Information***: display the total number of existing levels
- ***Section 2 – Create New Level***: select this option to create a new level
- ***Section 3 – Load Existing Level***: select this option to load and optionally edit an existing level

4.2 Creating a new level



1. Select the **Create New Level** option of the level editor
2. The number of this new level is displayed at the *Next Level* line
3. Optionally, click the *Clear Level* button to remove all level objects that currently exist in the scene
4. Go to folder *Prefabs/Game/Clocks* and drag appropriate clocks to the scene
5. Arrange and scale the clocks appropriately to form the desired level
6. Configure the clocks: each clock has a few parameters that define its behavior which you can adjust; you can click a clock to reveal its *ClockController* component.



a. *ClockType*: whether this clock is normal or has special effect
b. *ArrowRotatingDirection*: the rotating direction of clock's hand (arrow).
c. *ArrowRotatingSpeed*: the rotating speed of the clock's arrow.
d. *isShootingClock*: start shooting at this clock when the level begins, *each level must have only one clock with this variable set as true*.

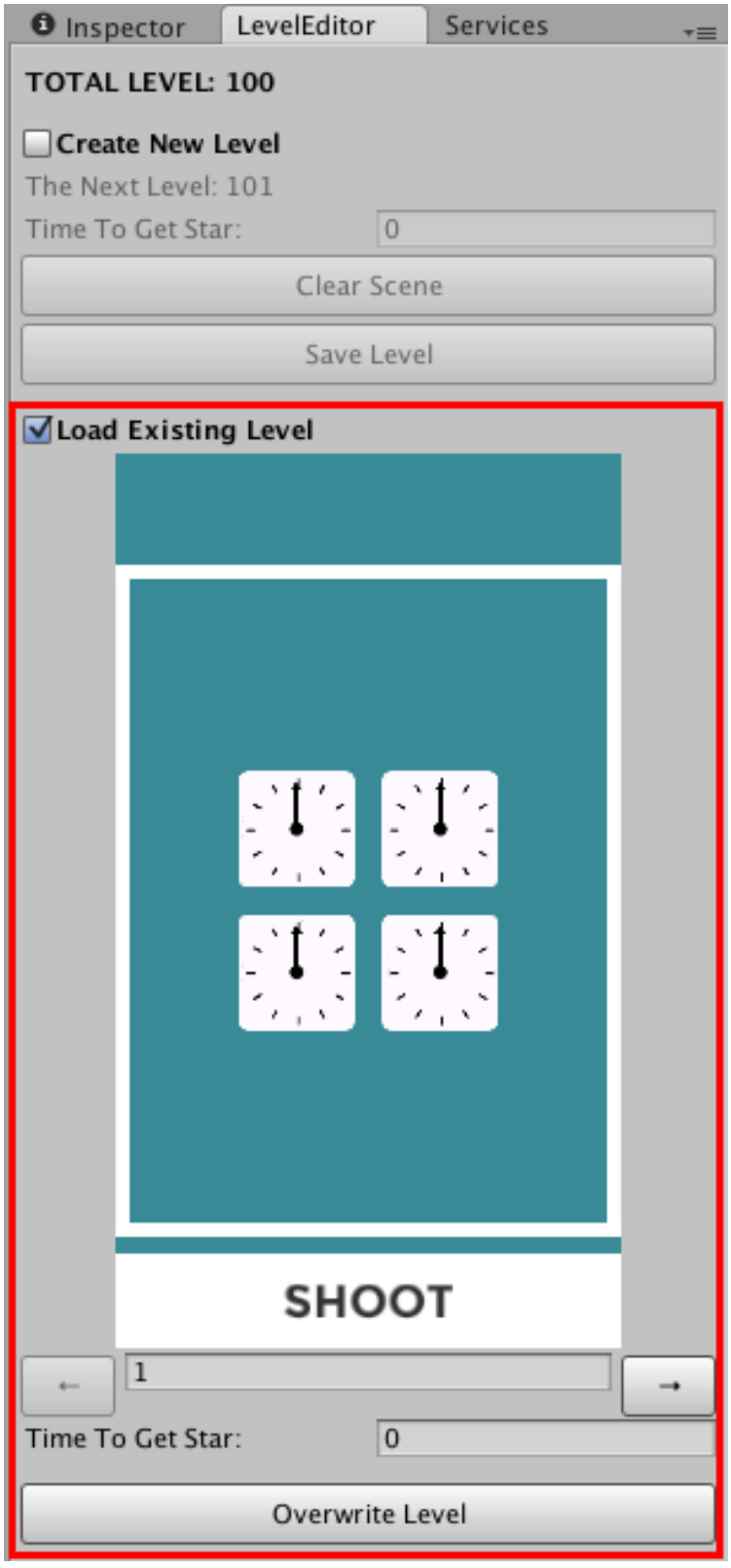
7. Select a clock as the starting one and set its *isShootingClock* to TRUE as mentioned in previous step
8. Enter the maximum time allowed to pass this level with a star (in seconds)
9. Run the scene and test your level
10. Click *Save Level* button in the level editor when the level looks good to you

**** IMPORTANT:** note that you should select the resolution of the Game view to 9:16 (or iPhone 5 Tall), and use that as the standard when designing levels as it is arguably the "thinnest" portrait resolution. This is to prevent potential displaying issue: e.g. if you designed on an iPhone Pad Tall resolution (3:4), and placed the props very near the screen edges, they might appear outside the viewable area of an iPhone 5 screen.*

There're two things happen when a new level is saved:

1. The level screenshot is automatically captured and saved in folder *Resources/Screenshots* using the level number as the filename
2. The LevelsData.json file at folder *Resources* is updated with the data of the new level

4.3 Editing an existing level



1. Select the **Load Existing Level** option in the level editor
2. Use the two arrow buttons navigate to the required level, or type its number to the middle text field; the selected level will be loaded to the Game view of the scene
3. Edit the level by adding, removing, rearranging or adjust clock parameters as needed in the scene; you can also change the *TimeToGetStar* value.
4. When you're done with the editing, hit the *Overwrite Level* button in the level editor to save the changes

***Tip:** you can load an existing level, then select the **Create New Level** option, then modify the level and save it as a new one, which may be faster than creating the whole level from scratch.*

5 ENABLING PREMIUM FEATURES

Premium features include:

- Advertising
- In-app purchasing
- Leaderboard and achievement
- Sharing
- Push notifications

To enable premium features of this template, you need to download and import Easy Mobile plugin from http://u3d.as/Dd2.

This section provides a guide on configuring each feature for your game. If you're not familiar with using Easy Mobile, it is strongly recommended that you read through its user guide to familiarize yourself with the plugin.

5.1 Before You Begin

- In the *FirstScene* scene's hierarchy, there's an object named *PremiumFeaturesManager* which contains all the relevant components from which you can configure how premium features behave in your game.
- Make sure the *EnablePremiumFeatures* option in the *GameManager* object (in scene *GameScene*) is checked.

- Make sure there's an EasyMobile object in the *FirstScene* scene's hierarchy, otherwise 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* >

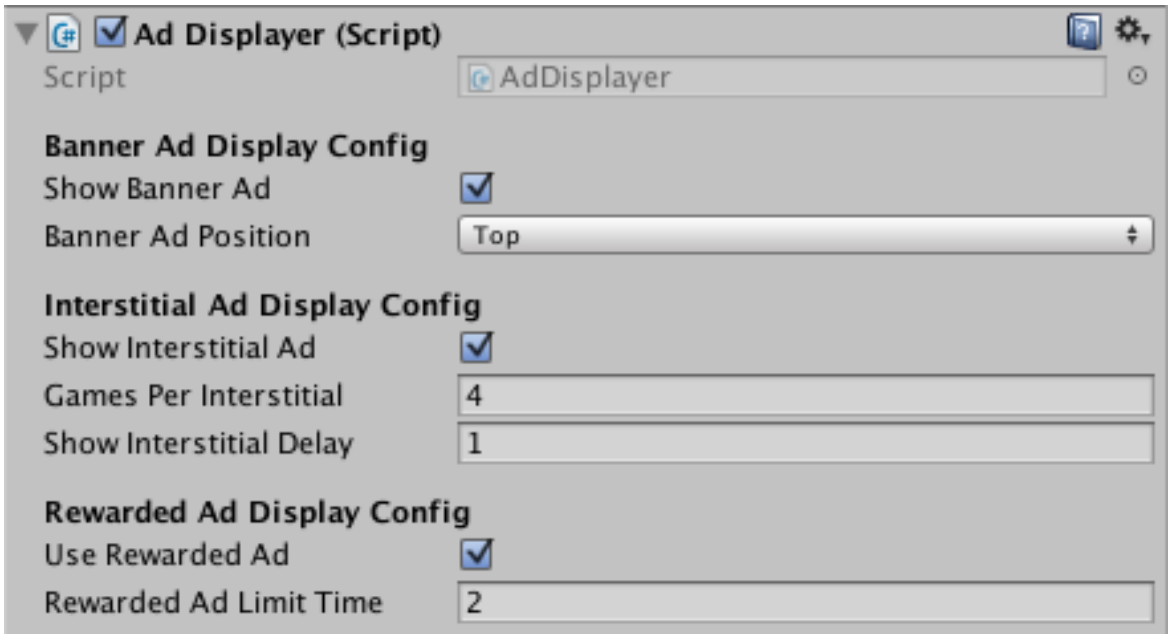
Easy Mobile > *Settings*, this is the only place to go to configure this plugin.

- 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).

5.2 Advertising

5.2.1 Template-specific setup

The *PremiumFeaturesManager* object contains a component 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.

- *Use Rewarded Ads*: whether to allow the user to watch an ad to undo a missed shot
- *Rewarded Ad Limit Time*: minimum time (minutes) that the player needs to wait until another rewarded ad can be served

5.2.2 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

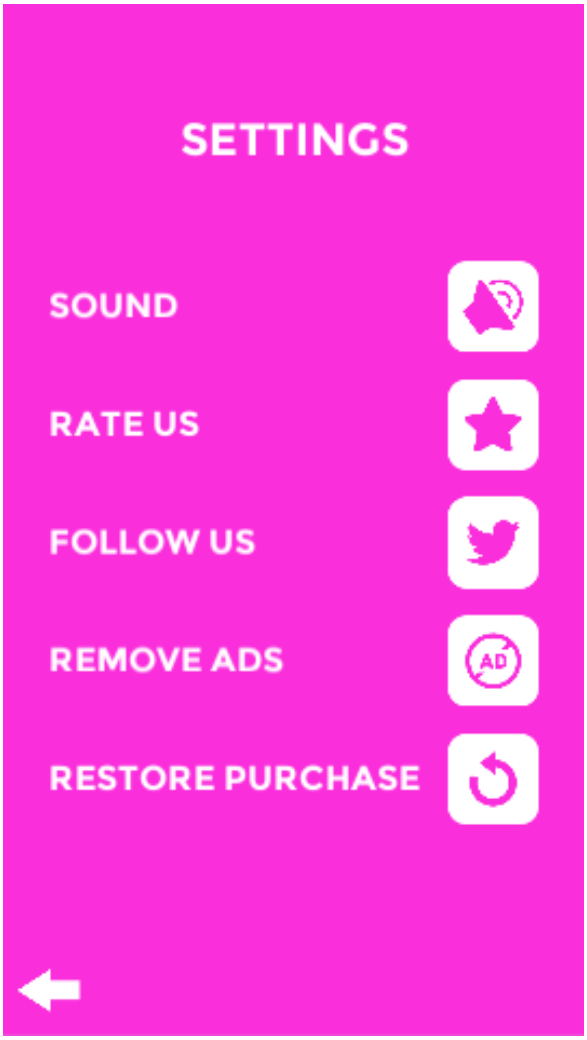
b. Enable auto ad-loading feature: simply leave the *Auto-Load Default Ads* option as checked and other parameters as default, the plugin will automatically load ads in the background
c. Select default ad networks for each platform: choose your preferred network for each type of ad on each platform

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

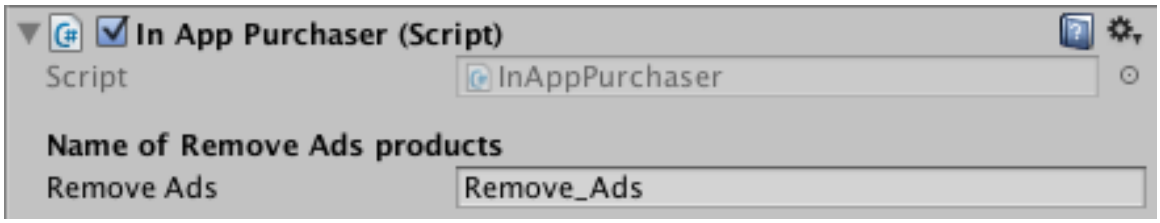
5.3 In-App Purchasing

5.3.1 Template-specific setup

This template includes a *Remove Ads* button as an in-app purchase item. There's also one *Restore Purchase* button as required on iOS.



The *PremiumFeaturesManager* contains a component named *InAppPurchaser* which manages all the in-app purchasing activities in this game.



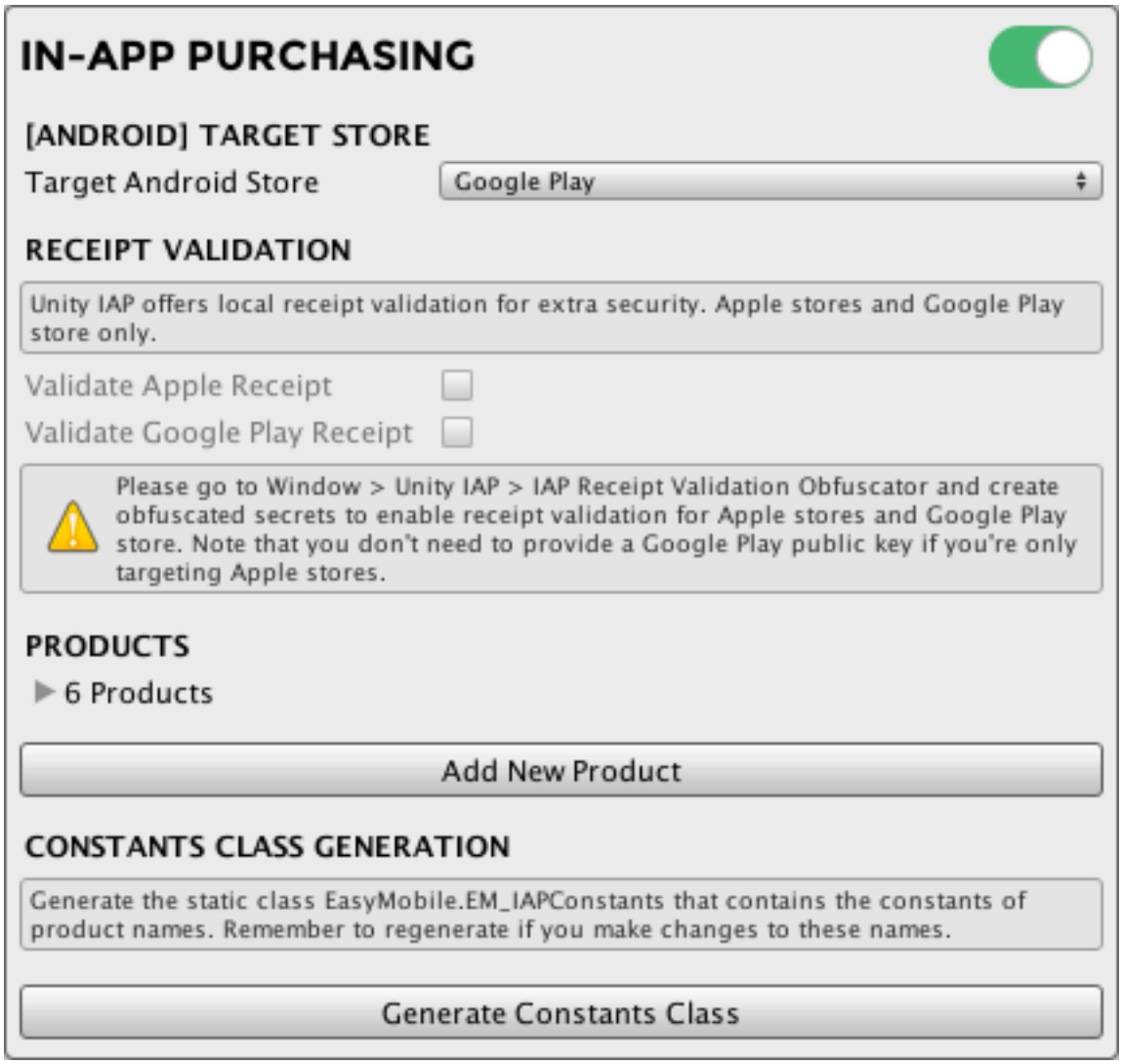
You can change the name Remove_Ads if you wish, only make sure that you use the exact same name when creating the product in Eays Mobile.

5.3.2 Easy Mobile setup

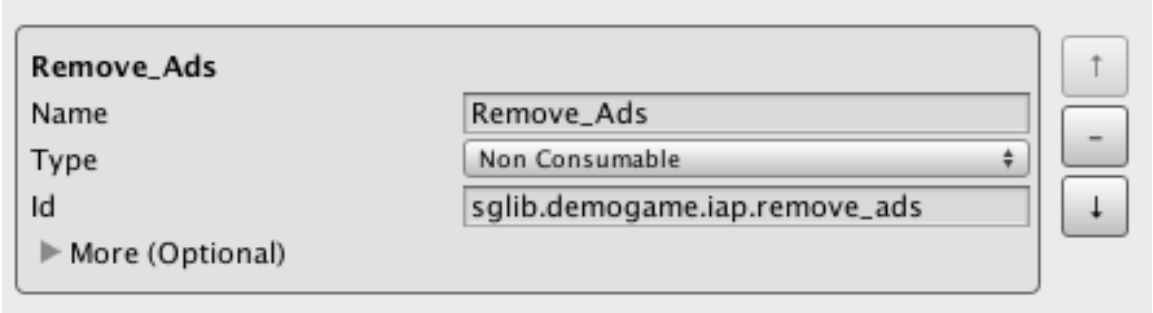
Setting up the In-App Purchasing module of Easy Mobile includes the following steps. Please see the In-App Purchasing section in Easy Mobile's user guide for detailed instructions on each step.

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.



Note that the products declared with Easy Mobile must have names that match with the ones you have in the aforementioned *InAppPurchaser* component. Also note that *Remove Ads* should be a non-consumable product.



5.3.3 Create the products for targeted stores

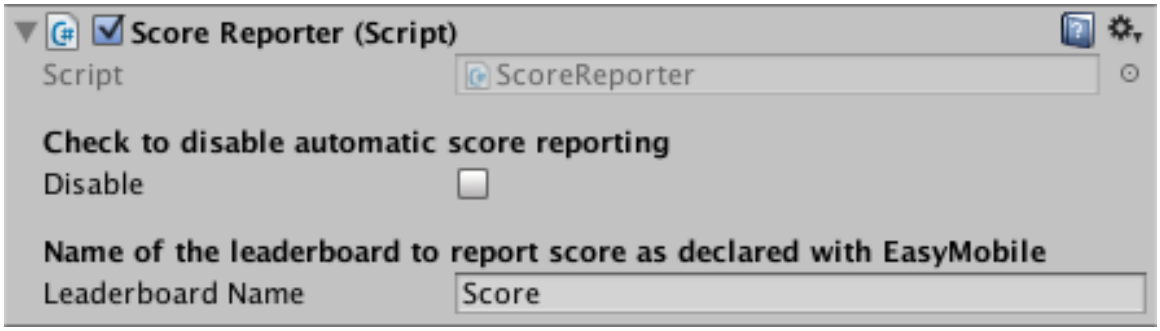
That last step in configuring the in-app purchasing feature is to create products for your targeted stores (e.g. Google Play and Apple App Store). Make sure the product ID, product type and price match the ones you have in your game.

5.4 Game Service

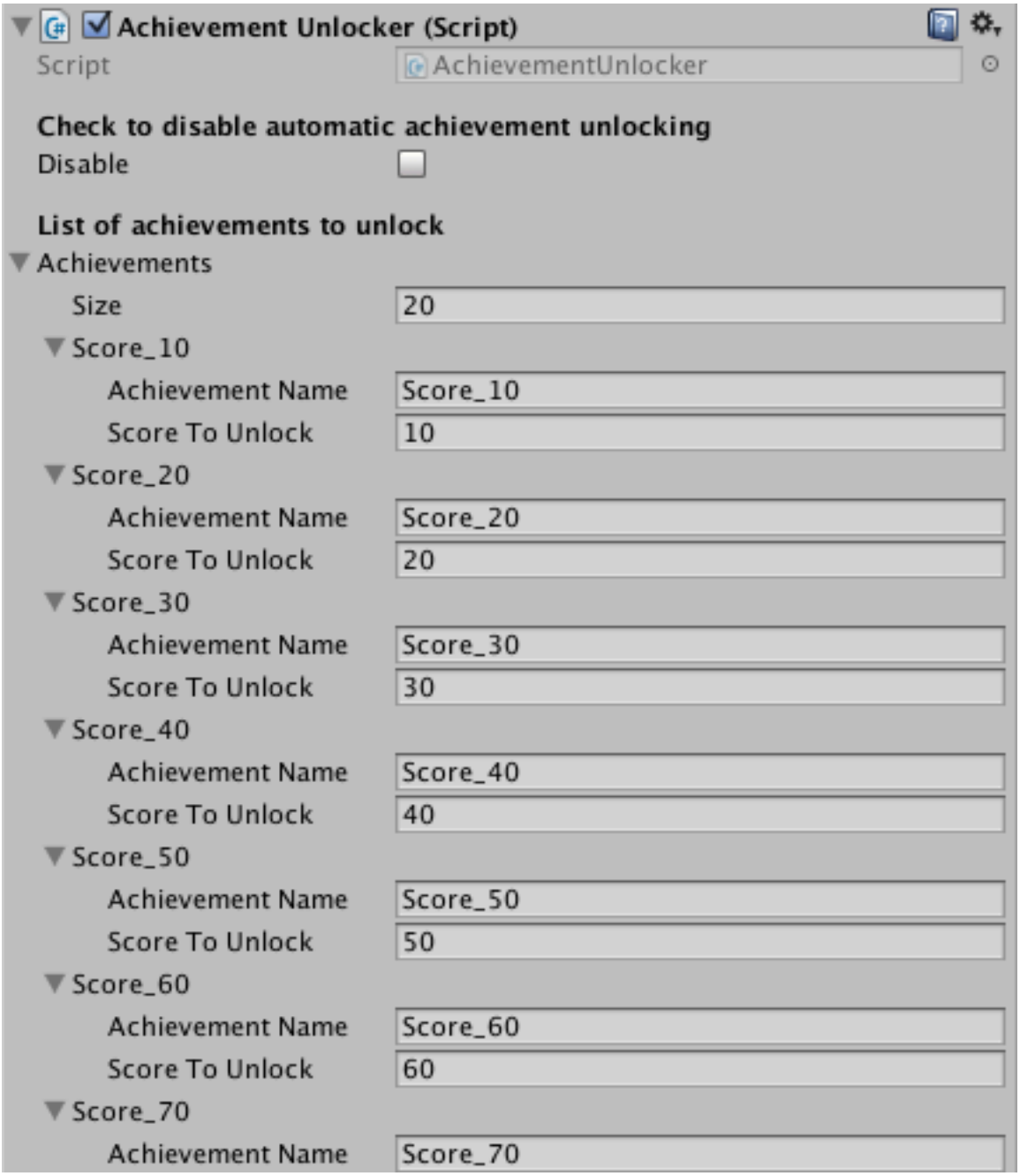
5.4.1 Template-specific setup

This template has a built-in leaderboard for ranking users' scores in Endless mode, 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 a component named *ScoreReporter*, which is also attached to PremiumFeaturesManager object. 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 the component named *AutoAchievementUnlocker*. In this component, you can modify existing achievements and add or remove achievements. You can also disable the automatic achievement unlocking feature if you wish.



5.4.2 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.

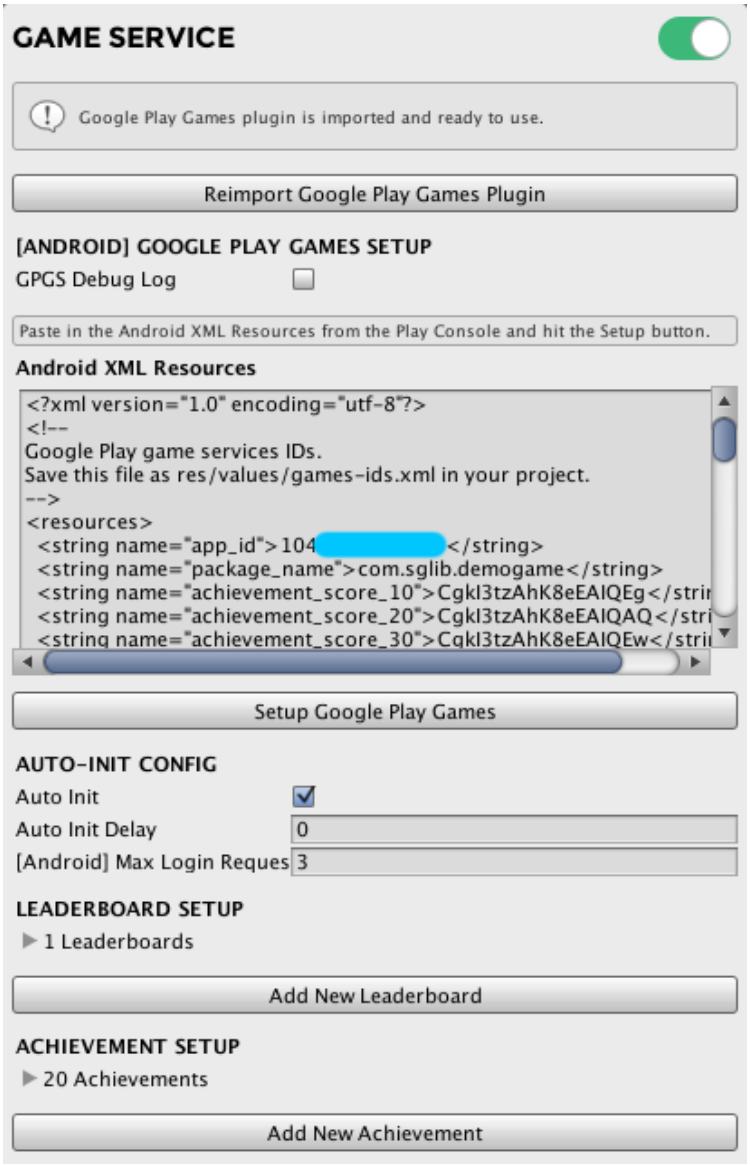
5.4.3 Easy Mobile setup

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. Import Google Play Games plugin for Unity and setup it 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.

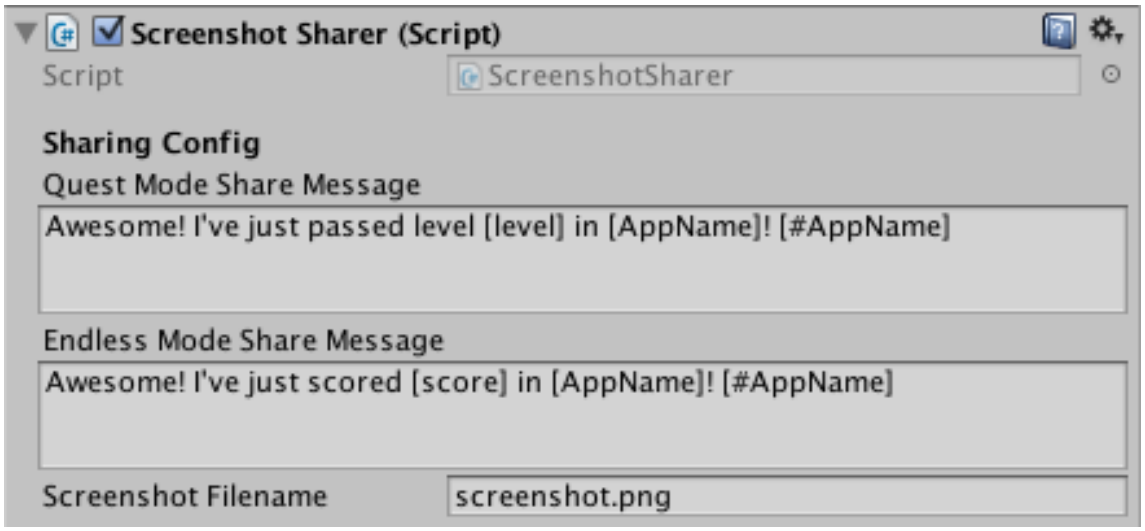


Note that you must declare the leaderboard and achievements with the same names as the ones you have in the *AutoScoreReporter* and *AutoAchievementUnlocker* components. Also their IDs must match the ones you

created in iTunes Connect and Google Play Developer Console.

5.5 Native Sharing

This game has a *Share* button that allows the player to share the game screenshot to social networks using the native sharing functionality. This activity is managed by a component named *ScreenshotSharer*, which is also attached to the *PremiumFeaturesManager* object.



Here you can configure the sharing feature.

- *Quest Mode Share Message*: the sharing message when finishing a level, note that [level] will be replaced by the actual level, [AppName] will be replaced by the actual app name declared in AppInfo
- *Endless Mode Share Message*: the sharing message in Endless mode, note that [score] will be replaced by the 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.

5.6 Push notifications

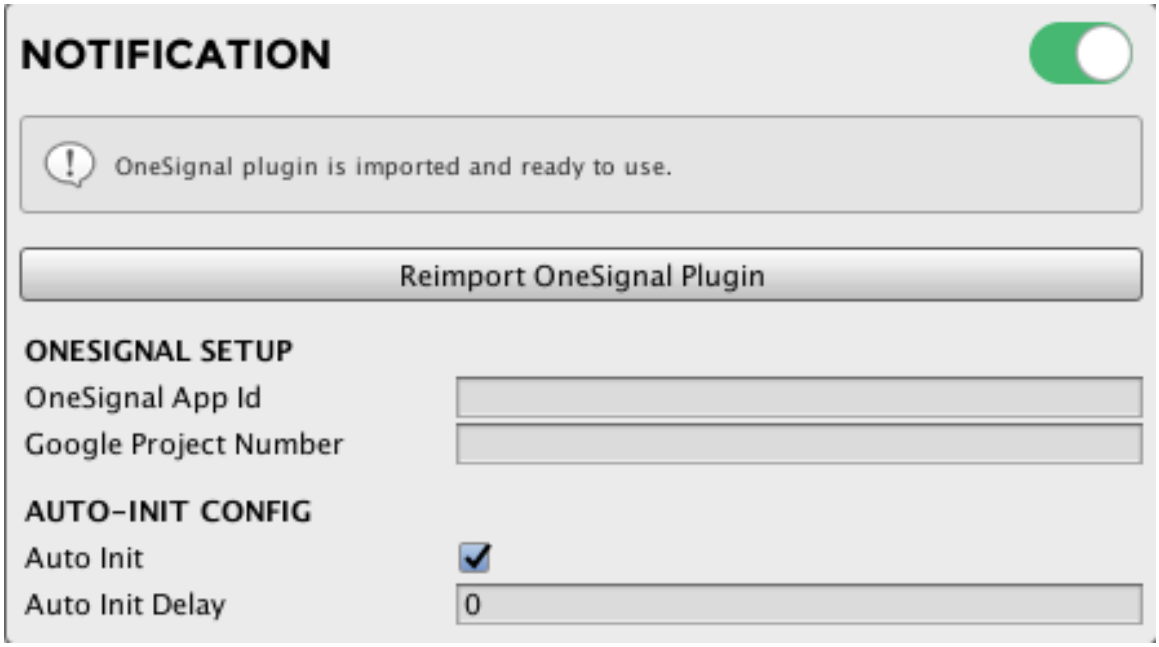
Enabling push notifications for your app using OneSignal service includes following steps. Please see the Notification section in Easy Mobile user guide for detailed instructions on each step.

- 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!