



DRAW.GAME TEMPLATE

USER GUIDE

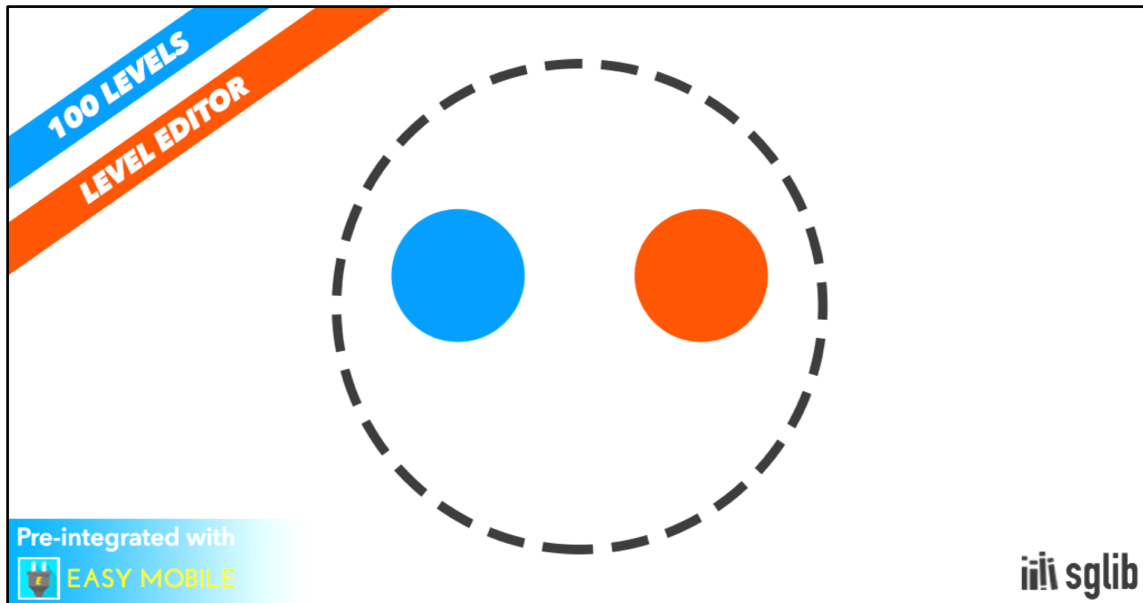
We strive to provide the best service as we can, if you have any questions or suggestions, please contact us!
Thank you!

SgLib Games

Table of Contents

1	INTRODUCTION.....	3
2	GETTING STARTED.....	4
2.1	ENTER APP INFORMATION	4
2.2	LINK THE GAME TO YOUR UNITY PROJECT.....	4
2.3	TESTING & BUILDING NOTE.....	6
3	TEMPLATE CUSTOMIZATION	6
3.1	SPLASH SCREEN	6
3.2	GAMEMANAGER.....	6
3.3	CUSTOMIZING UI	7
3.4	SOUNDS.....	8
4	LEVEL EDITOR.....	9
4.1	OVERVIEW	9
4.2	CREATING A NEW LEVEL	12
4.3	EDITING AN EXISTING LEVEL	14
4.4	ADDING MORE PROPS.....	15
4.4.1	<i>Adding simple props</i>	<i>15</i>
4.4.2	<i>Adding advanced props</i>	<i>15</i>
5	ENABLING PREMIUM FEATURES.....	16
5.1	BEFORE YOU BEGIN	16
5.1.1	<i>Template-specific setup.....</i>	<i>16</i>
5.1.2	<i>Easy Mobile setup.....</i>	<i>17</i>
5.2	IN-APP PURCHASING.....	19
5.2.1	<i>Template-specific setup.....</i>	<i>19</i>
5.2.2	<i>Easy Mobile setup.....</i>	<i>21</i>
5.2.3	<i>Create the products for targeted stores</i>	<i>23</i>
5.3	NATIVE SHARING.....	23
5.4	PUSH NOTIFICATIONS	24

1 INTRODUCTION



In **Draw.Game**, to clear a stage the player needs to draw ingenious lines and shapes to help two little separated dots overcome all the obstacles and meet each other. This unique physics-based puzzler 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:

- Interesting and addictive gameplay
- **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 (not used in this template)
- 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 template does not 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.

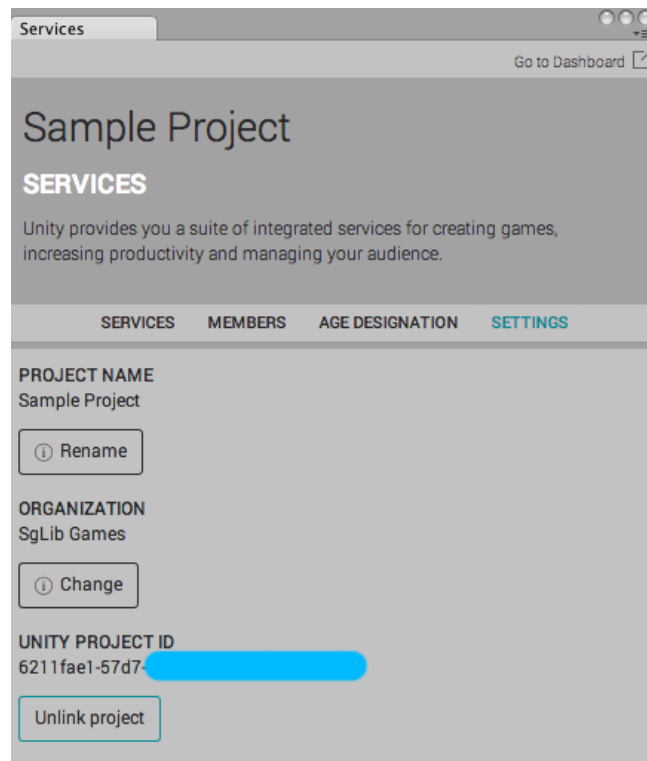


Script	AppInfo
APP_NAME	[YOUR_APP_NAME]
APPSTORE_ID	[YOUR_APPSTORE_ID]
BUNDLE_ID	[YOUR_BUNDLE_ID]
APPSTORE_HOMEPAGE	[YOUR_APPSTORE_PUBLISHER_LINK]
PLAYSTORE_HOMEPAGE	[YOUR_GOOGLEPLAY_PUBLISHER_NAME]
FACEBOOK_ID	[YOUR_FACEBOOK_PAGE_ID]
TWITTER_NAME	[YOUR_TWITTER_PAGE_NAME]
SUPPORT_EMAIL	[YOUR_SUPPORT_EMAIL]

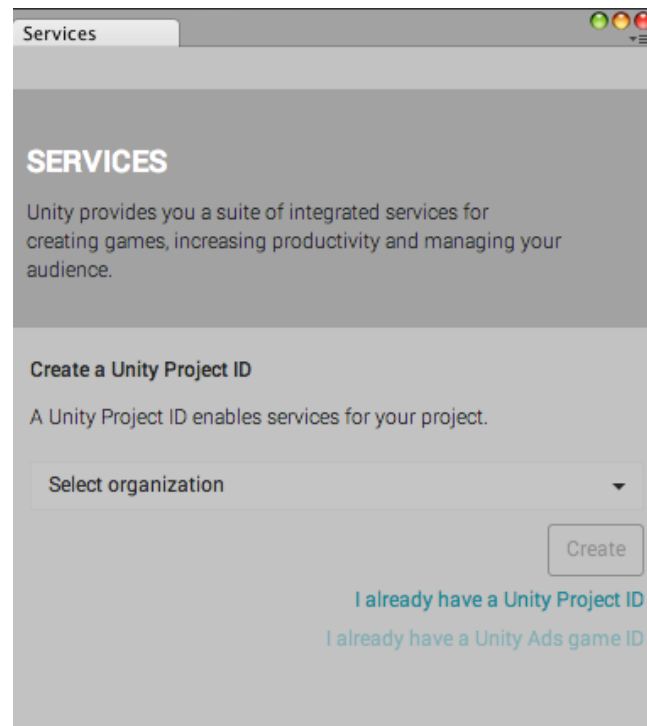
2.2 Link the game to your Unity project

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

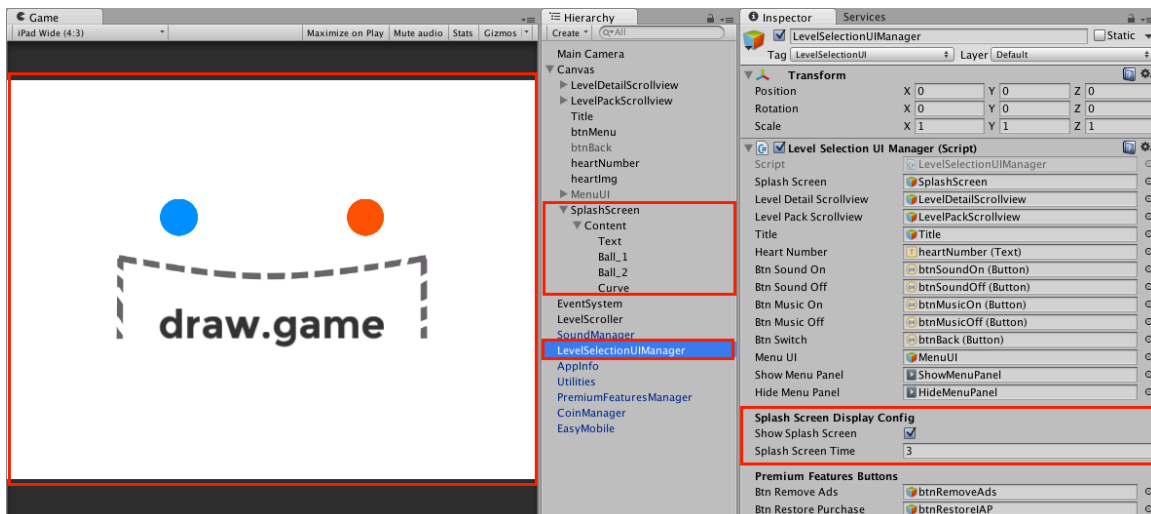
2.3 Testing & Building Note

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 Splash screen

This template has a custom splash screen which is displayed when the game starts (this one is different from Unity splash screen and is displayed after that splash screen). This splash screen is actually put inside the canvas of the *FirstScene* scene: in the hierarchy, you'll find a *SplashScreen* object under the Canvas object. Feel free to edit this splash screen to suit your needs. Additionally, you can also control the display of this splash screen from the *LevelSelectionUIManager* object in the hierarchy.

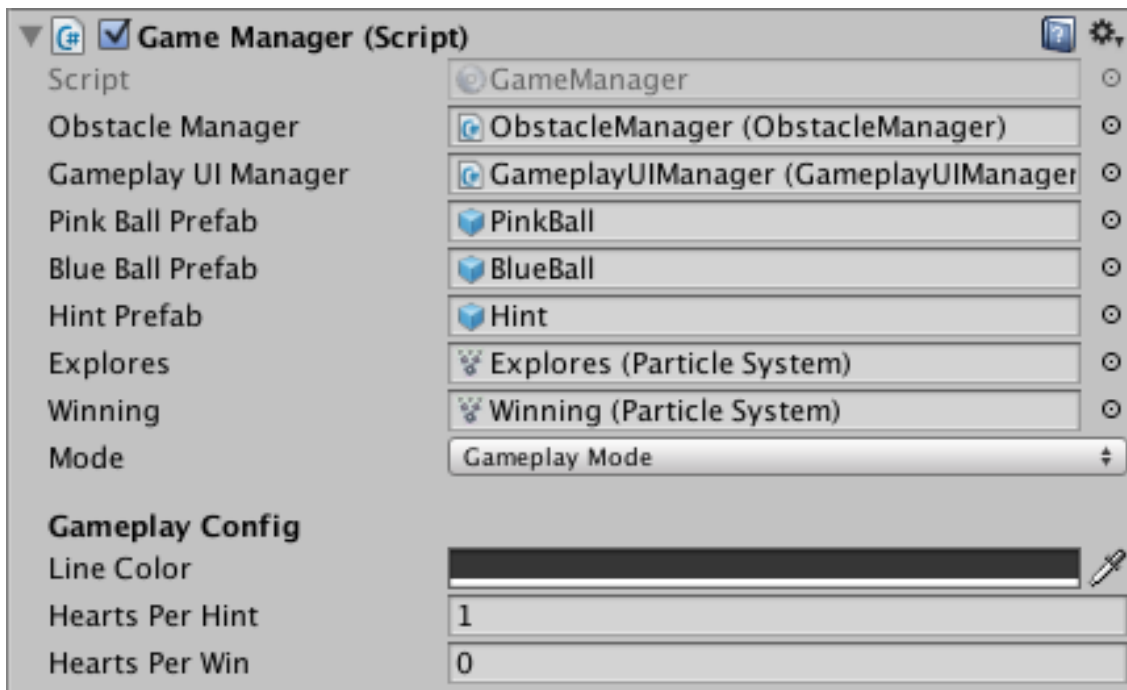


In the **Splash Screen Display Config** there're two variables:

- *Show Splash Screen*: whether to show the splash screen at all
- *Splash Screen Time*: how long the splash screen should be displayed

3.2 GameManager

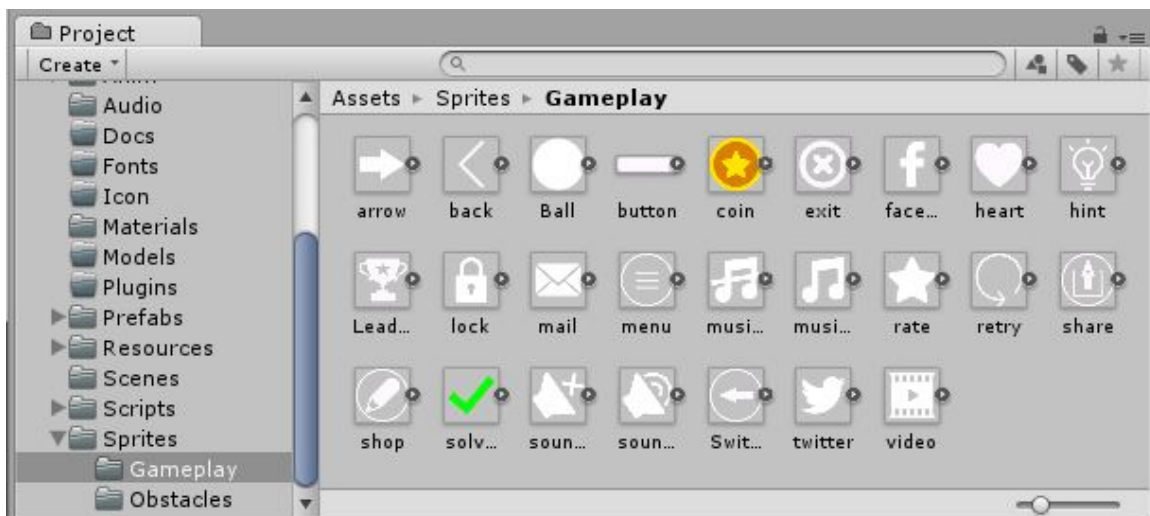
In the scene *GameScene* there's an object named *GameManager* where you can control some important variables of the game.



- *Line Color*: color the the drawn lines
- *Hearts Per Hint*: how many hearts must be spent to view a hint
- *Hearts Per Win*: how many hearts should be rewarded when the player clears a stage, put 0 to disable this feature

3.3 Customizing UI

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.



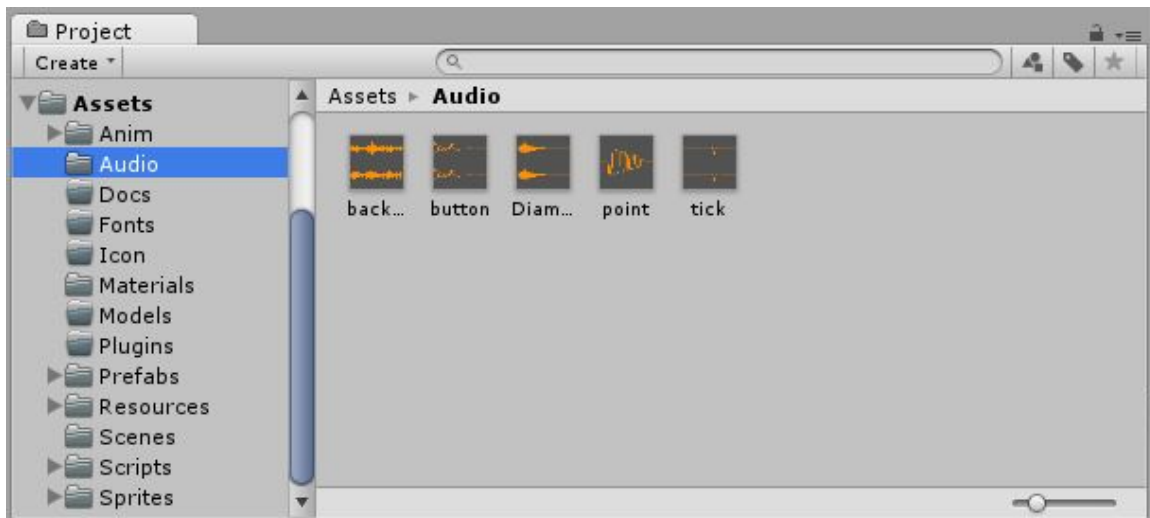
All fonts used in this game are free-to-use in commercial projects. Fonts are

located under the *Fonts* folder together with appropriate license files.

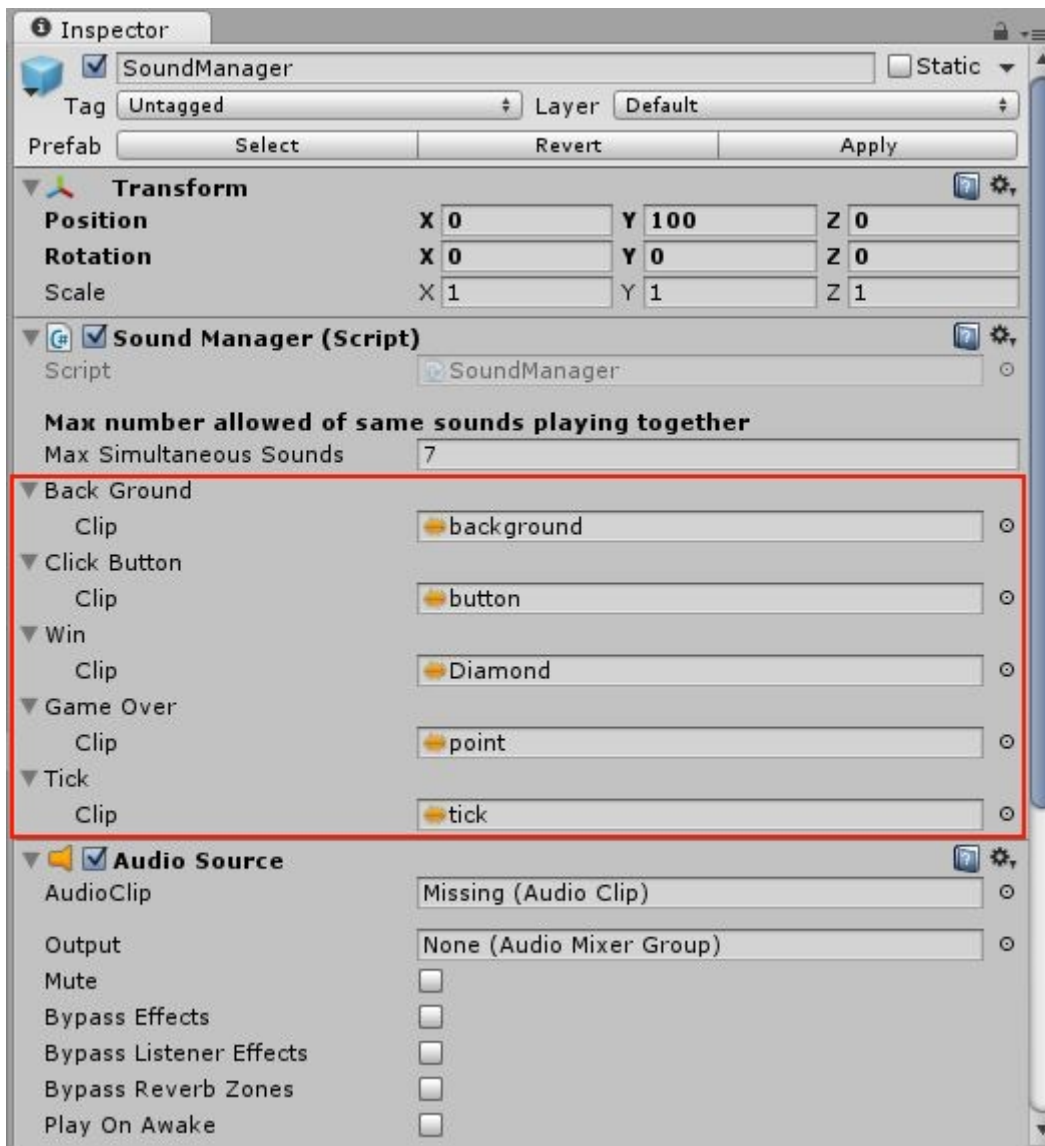


3.4 Sounds

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.



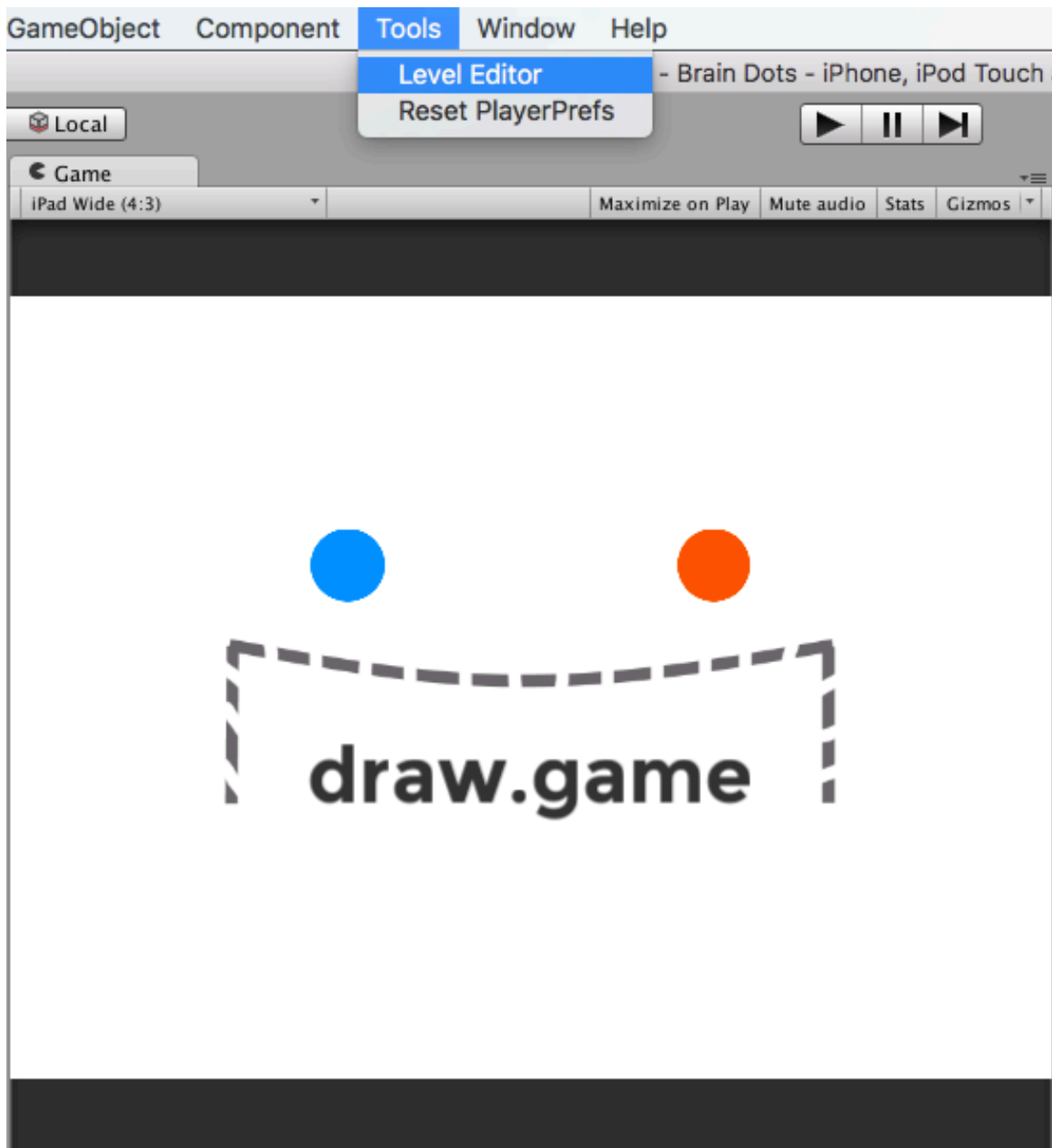
4 LEVEL EDITOR

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.

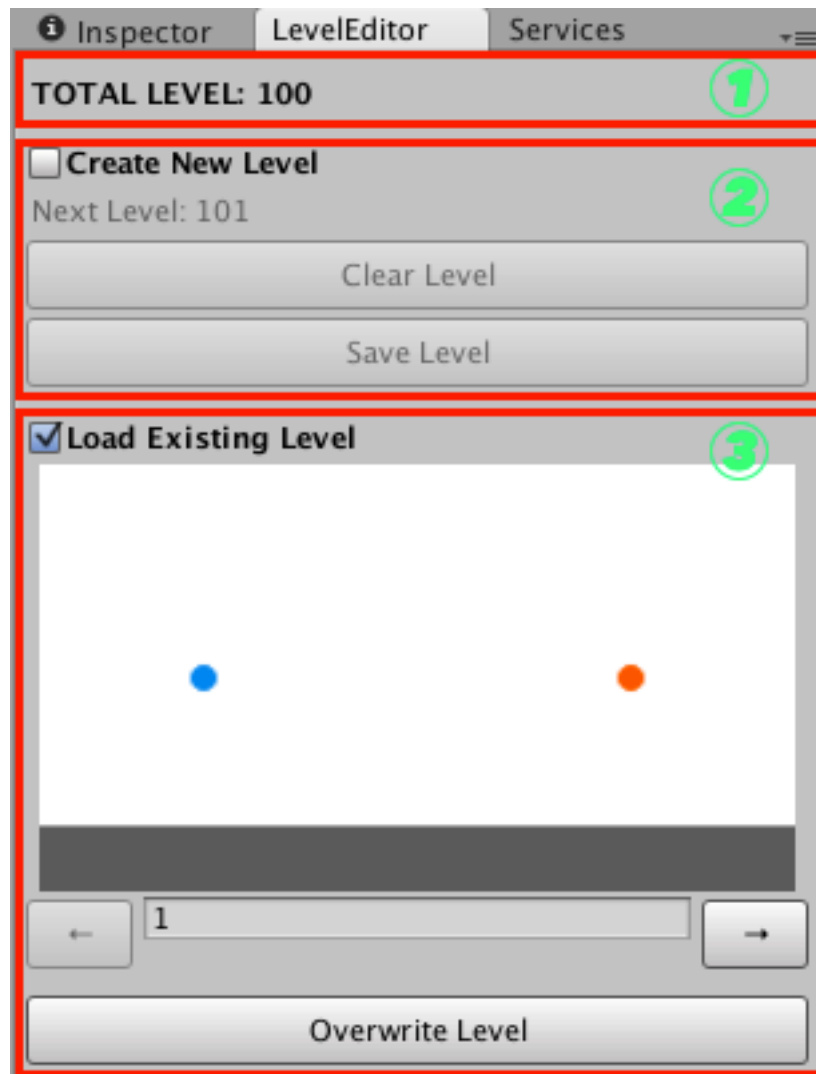
If you prefer a video tutorial, we have one at <https://youtu.be/s2J-fYHbFZ4>

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.



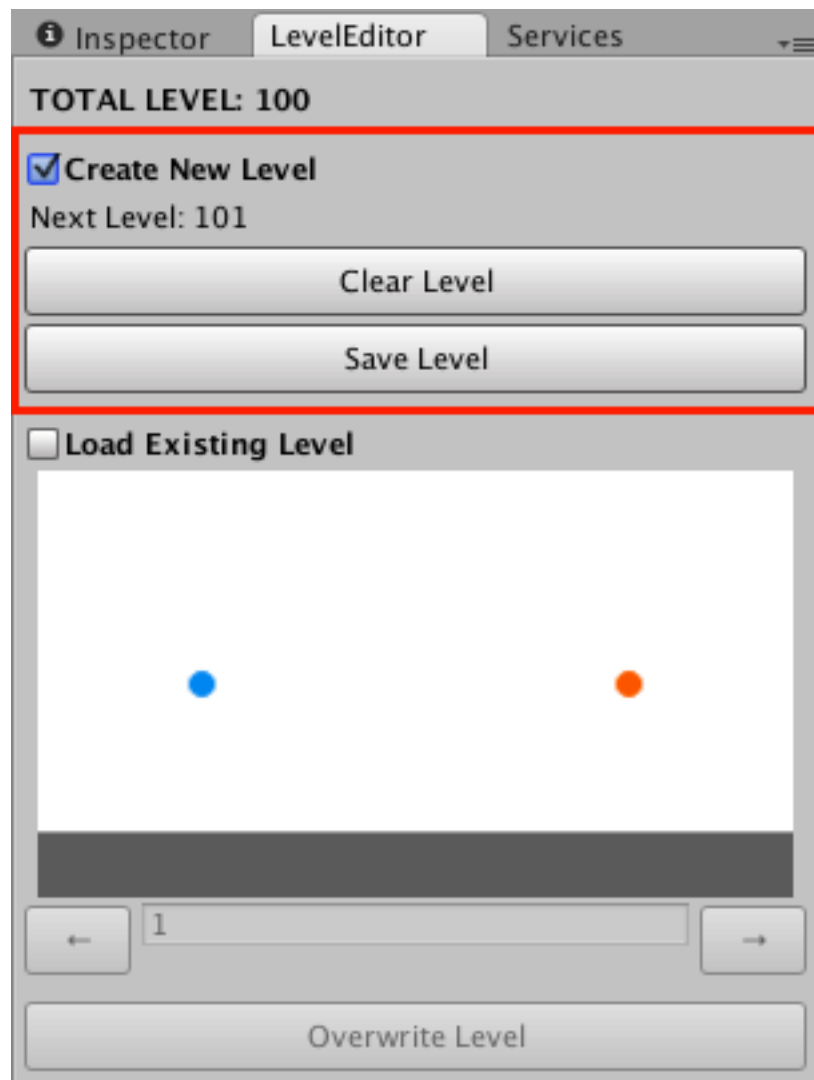
Below is the interface of the level editor.



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

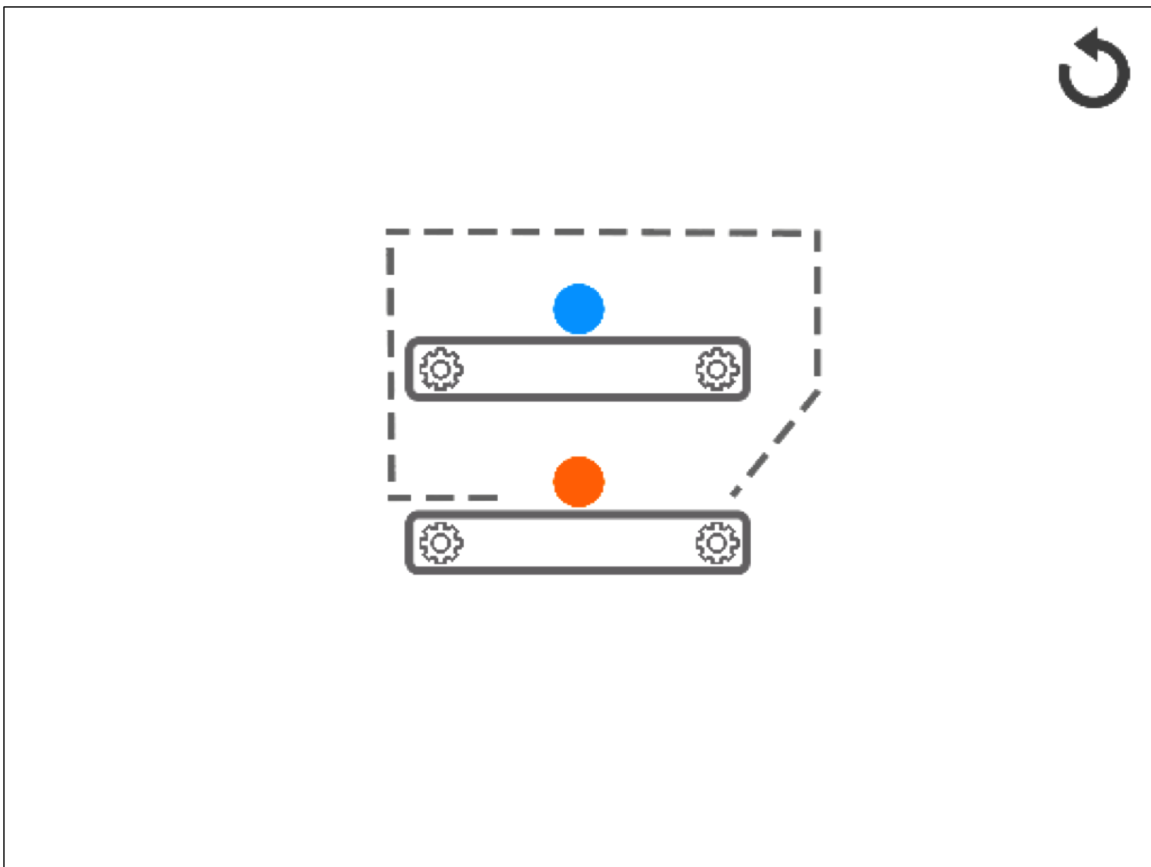
4.2 Creating a new level



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 (dots/balls and any other props and hints)
4. Drag the two ball prefabs from folder *Prefabs/Game/Balls* to the hierarchy, note that you won't be able to save a level unless these two balls exist in the scene
5. Add more props: drag appropriate prefabs from folder *Prefabs/Game/Obstacles* to the scene hierarchy
6. Arrange the two balls and the props appropriately in the scene to form your desired level. *Note that you should select the resolution of the Game view to 4:3 (or iPad Wide), and use that as the standard when designing levels as it is arguably the "shortest" landscape resolution, to prevent possible*

- displaying issues (e.g. if you designed on an iPhone 5 Wide resolution (16:9), and placed the props very near the screen edges, they might appear outside the viewable area of an iPad screen)*
7. Run the scene and test your solution
 8. Add the hint:
 - a. Draw the hint using your image-editing software
 - b. Save it to folder *Resources/Hints* of the game under the name *hint[level_number].png*, e.g. if the new level is 101 then the hint name should be *hint101.png*
 - c. Drag the *Hint* prefab from folder *Prefabs/Game/Hint* to the scene
 - d. Set the sprite of the *SpriteRenderer* component of this hint to the hint image saved in previous step
 - e. Scale and position the hint appropriately
 9. Click *Save Level* button in the level editor when the level looks good to you

Below is an example premade level with a hint.

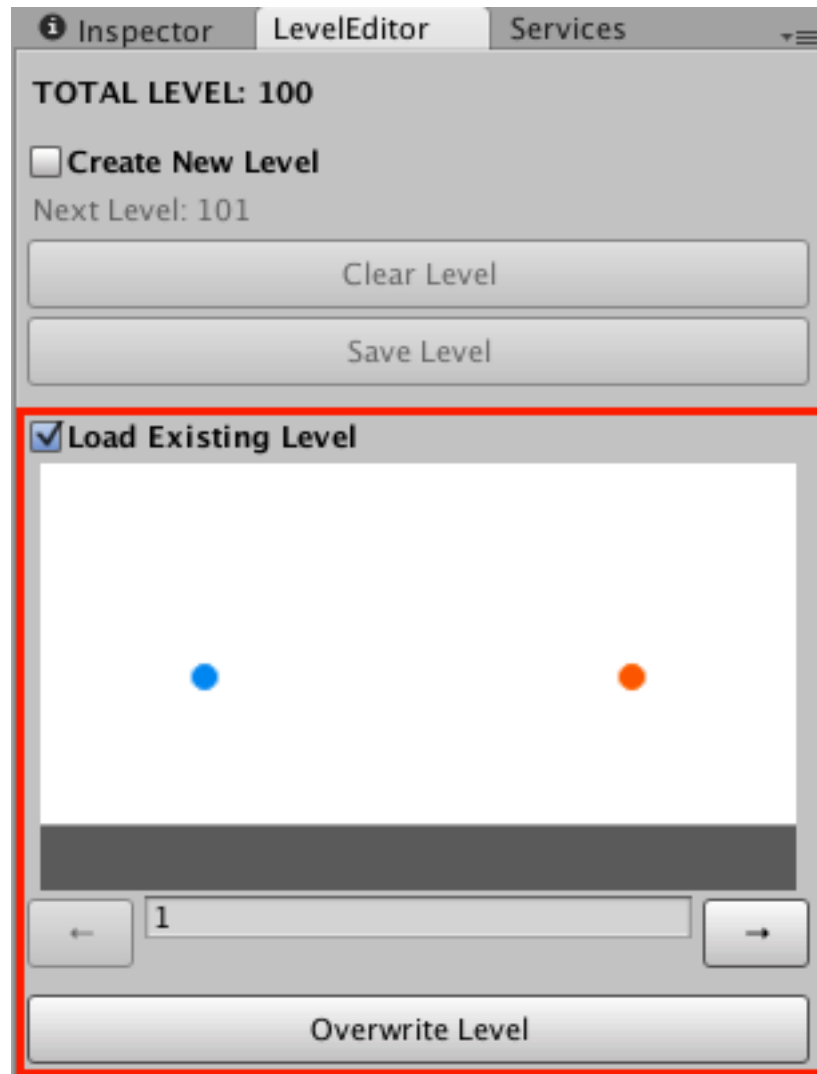


There're two things happen when a new level is saved:

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/Json* 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, or repositioning the props as needed in the scene.
 - a. Note that the props are stored at folder *Prefabs/Game/Obstacles*.
 - b. Also note that you should select the resolution of the Game view to 4:3 (or iPad Wide), and use that as the standard when designing levels as it is arguably the “shortest” landscape resolution, to prevent possible displaying issues (e.g. if you designed on an iPhone 5 Wide resolution (16:9), and placed the props very near the screen edges,

they might appear outside the viewable area of an iPad screen)

4. Adjust the hint if needed, if you need to replace a hint image with a new one, simply save the new image with the same name to overwrite the old one in *Resources/Hint* folder
5. 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.

4.4 Adding more props

Beside the existing props (obstacles), you can add more if you wish.

4.4.1 Adding simple props

Simple props are the static props without any special effects, for example wall and rock. To add a simple static props:

1. Duplicate an existing prefab in the *Prefabs/Game/Obstacles/FixedObstacles* folder
2. Give the new prefab an appropriate name which must be different from all existing prefab names in the *Prefab/Game/Obstacles* folder
3. Replace the *Sprite* of the *SpriteRenderer* component of the prefab to the sprite of your new prop
4. Reset the *Polygon Collider 2D* component to match the new sprite
5. In *LevelEditor* scene, enlarge the *Obstacles* array of the *ObstacleManager* object, then drag your new prefab to the end of this array. Hit *Apply* to save the changes to the prefab
6. Now you can use the new prop with the level editor

4.4.2 Adding advanced props

Advanced props can have special movements or special effects, for example the existing conveyor props can be considered as an advanced prop.

To add an advanced prop:

1. Create the prefab of the prop and store it in the *Prefabs/Game/Obstacles* folder. You would need to add required components as well as creating required script, animation, etc. for the prop to implement its special movement or effect.
2. In *LevelEditor* scene, enlarge the *Obstacles* array of the *ObstacleManager* object, then drag your new prefab to the end of this array. Hit *Apply* to save the changes to the prefab
3. Attach a new *MonoBehavior* script to the prop prefab (if needed) and implement an *OnTriggerEnter2D* or *OnCollisionEnter2D* method (depending on the collider type of your new prop) to do whatever actions

- required when a ball collides with the prop (you can detect the collision with the balls by checking the tag “Ball”)
4. Now you can use the prop with the level editor

5 ENABLING PREMIUM FEATURES

Premium features of this template include:

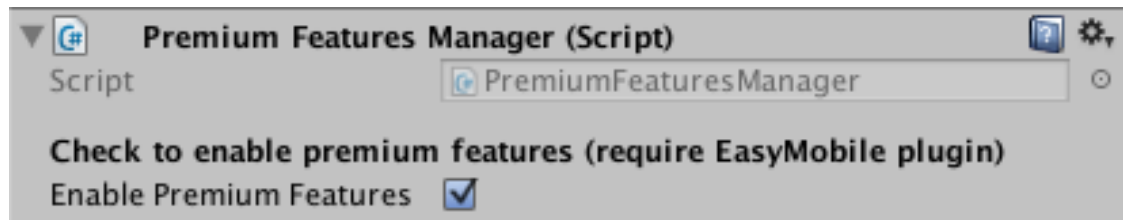
- Advertising
- In-app purchasing
- Sharing
- Push notifications

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 *PremiumFeaturesManager* object is checked.

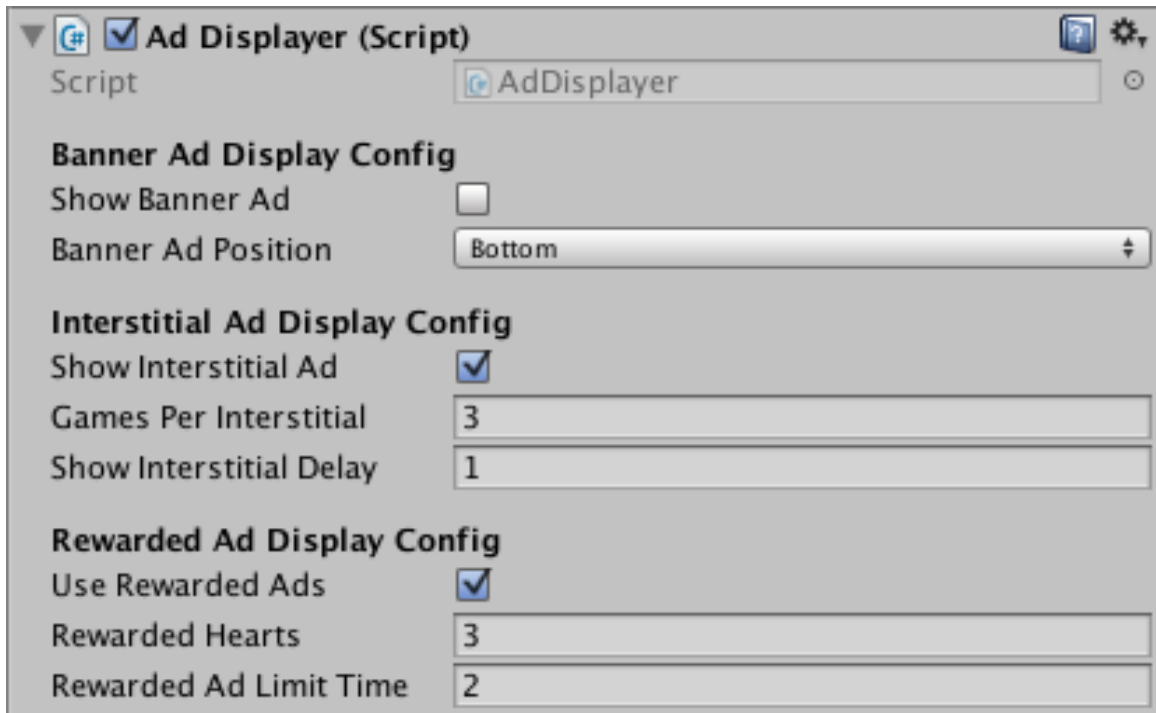


- Make sure there's an EasyMobile object in the *FirstScene* scene's hierarchy, otherwise you can add it using the EasyMobile prefab at folder *Assets/EasyMobile*. It is necessary for the plugin to function properly.
- The settings interface of Easy Mobile can be opened via menu *Window > Easy Mobile > 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).

5.1.1 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 earn hearts, which can then be used to view hints
- *Rewarded Hearts*: how many hearts should be awarded after watching an ad
- *Rewarded Ad Limit Time*: minimum time (minutes) that the player needs to wait until another rewarded ad can be watched, this is to limit the number of ads (and effective earned hearts) in a game session

5.1.2 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.

ADVERTISING

ADMOB SETUP

Google Mobile Ads (AdMob) plugin was imported.

Reimport Google Mobile Ads Plugin

► [iOS] AdMob Ids
► [Android] AdMob Ids

CHARTBOOST SETUP

Chartboost plugin not found. Please download and import it to show ads from Chartboost.

Download Chartboost Plugin

HEYZAP SETUP

Heyzap plugin not found. Please download and import it to show ads from Heyzap.

Download Heyzap Plugin

UNITY ADS SETUP

Unity Ads service is enabled.

AUTO AD-LOADING CONFIG

Auto-Load Default Ads ☒

Ad Checking Interval 10

Ad Loading Interval 20

DEFAULT AD NETWORKS

▼ [iOS] Default Ad Networks

Banner Ad Network Ad Mob

Interstitial Ad Network Ad Mob

Rewarded Ad Network Unity Ads

▼ [Android] Default Ad Networks

Banner Ad Network Ad Mob

Interstitial Ad Network Ad Mob

Rewarded Ad Network Unity Ads

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.

- 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
- 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
- Select default ad networks for each platform: choose your preferred

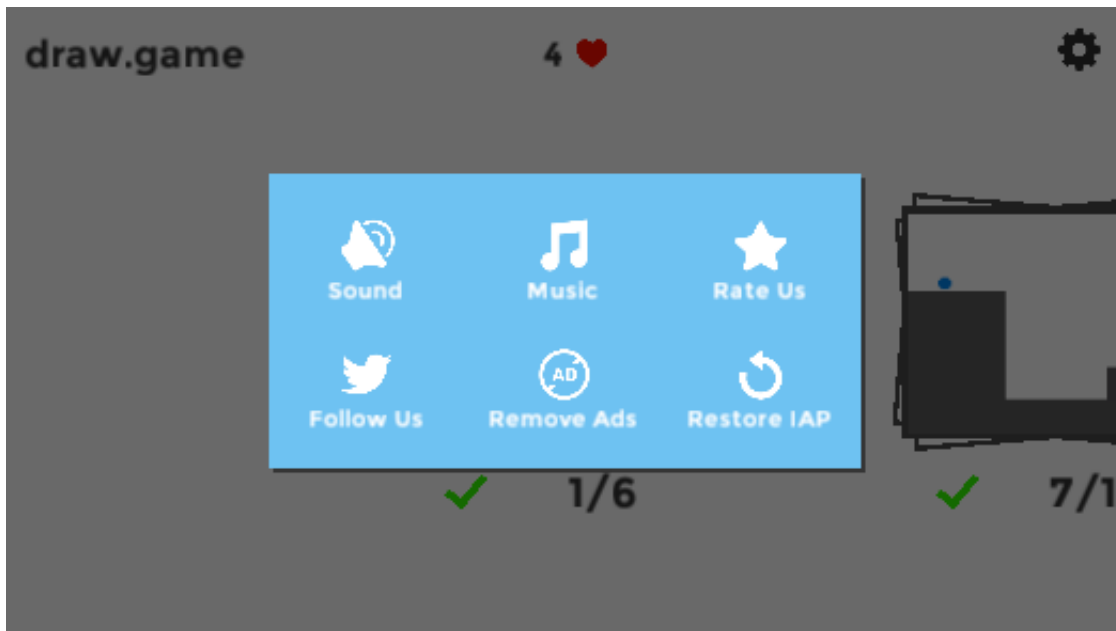
network for each type of ad on each platform

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

5.2 In-App Purchasing

5.2.1 Template-specific setup

The built-in in-app purchases of this template include a *Remove Ads* button, and several heart packs. You can modify existing products and add more packs if you like. 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.

In App Purchaser (Script)

Script

Name of Remove Ads products

Remove Ads

Name of coin pack products

▼ Heart Packs

Size

▼ Pack_5

Product Name

Price String

Value

▼ Pack_10

Product Name

Price String

Value

▼ Pack_20

Product Name

Price String

Value

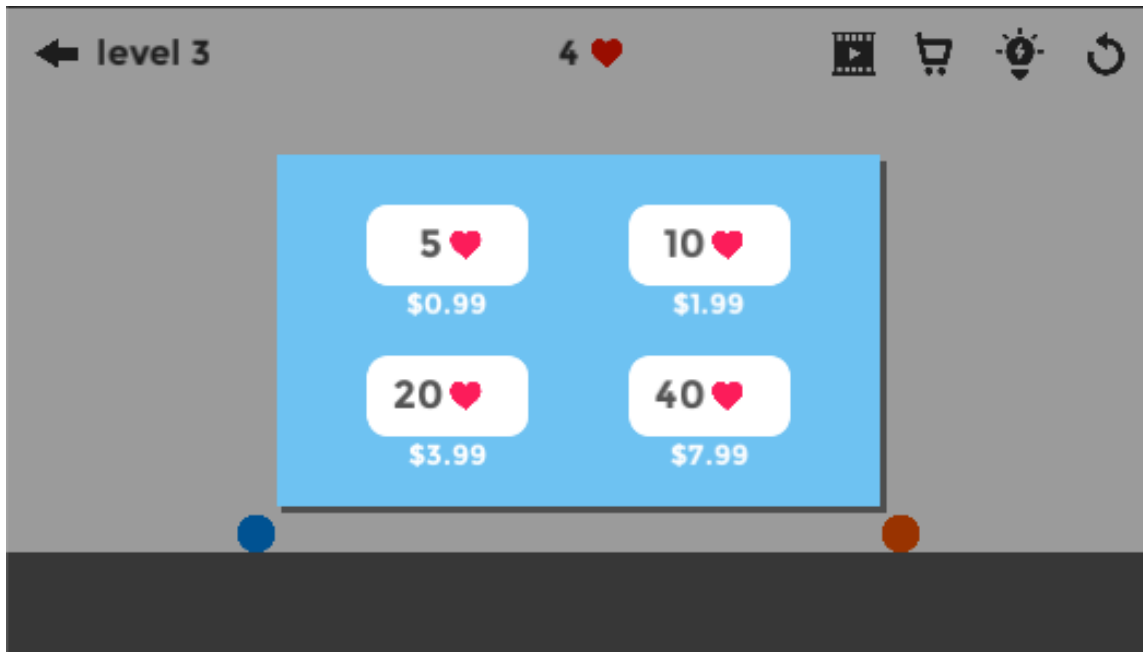
▼ Pack_40

Product Name

Price String

Value

Here you can modify the product definitions including the name, price or value of the packs. To add more packs, simply increase the size of *HeartPacks* array 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.



5.2.2 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* component. Also note that *Remove Ads* is a non-consumable product, while the heart packs must be consumable.

The screenshot shows a configuration interface for in-app purchases. It contains two main sections, each with a title, a 'Name' field, a 'Type' dropdown menu, an 'Id' field, and a 'More (Optional)' link. To the right of each section are three buttons: an up arrow, a minus sign, and a down arrow.

Product Name	Type	Id
Remove_Ads	Non Consumable	sglib.demogame.iap.remove_ads
Pack_5	Consumable	sglib.demogame.iap.coins_500

5.2.3 Create the products for targeted stores

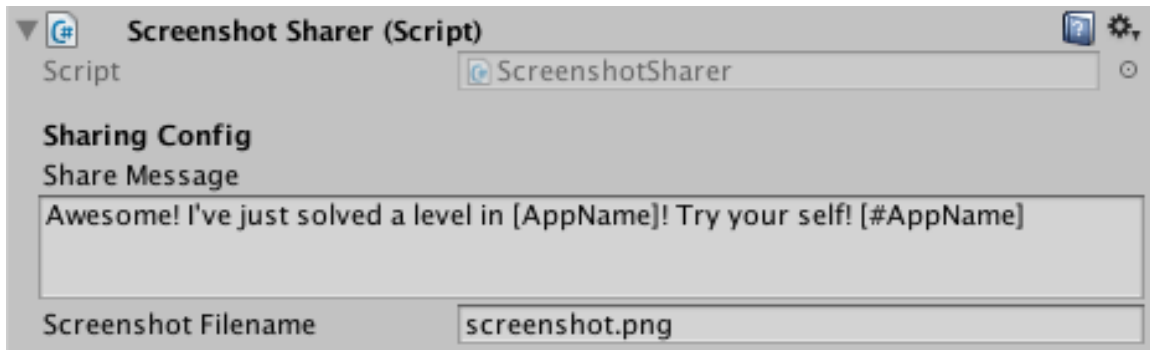
That last step in configuring the in-app purchasing feature is to create products for your targeted stores (e.g. Google Play and Apple App Store). Make sure the product ID, product type and price match the ones you have in your game.

5.3 Native Sharing

This game has a *Share* button that allows the player to share the level screenshot to social networks using the native sharing functionality after he/she solved (or failed to solve) it.



This activity is managed by a component named *ScreenshotSharer*, which is also attached to the *PremiumFeaturesManager* object.



Here you can configure the sharing feature.

- *Share Message*: the default sharing message, note that [AppName] will be replaced by the actual app name declared in AppInfo
- *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.


5.4 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.

NOTIFICATION

 OneSignal plugin is imported and ready to use.

Reimport OneSignal Plugin

ONESIGNAL SETUP

OneSignal App Id

Google Project Number

AUTO-INIT CONFIG

Auto Init☒

Auto Init Delay

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

THANK YOU AND GOOD LUCK WITH YOUR GAMES!