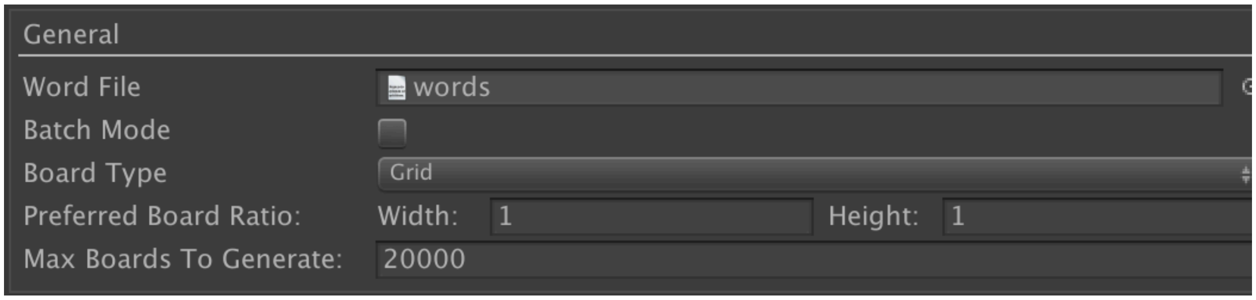
# Word Connect Documentation

# Level Builder Window Tool

The Level Builder window tool is used to create the level files that are loaded when the game runs. To open the window, select the menu item **Word Connect -> Level Builder**.

## General Settings



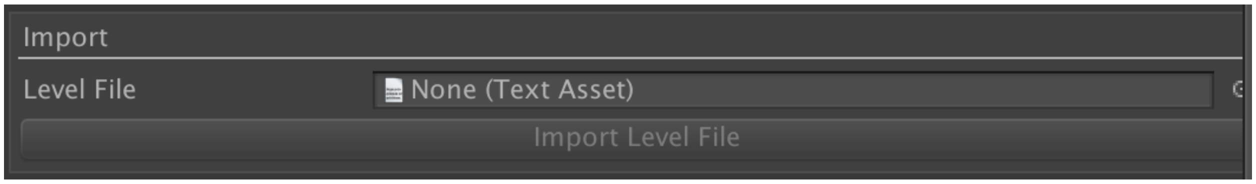
General

| | |
|---|---|
| Word File | words |
| Batch Mode | ☐ |
| Board Type | Grid |
| Preferred Board Ratio: | Width: 1 Height: 1 |
| Max Boards To Generate: | 20000 |

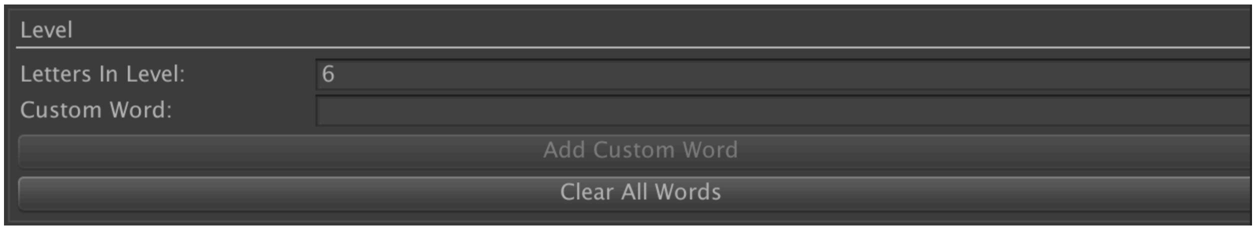
| **Fields** | |
|---|---|
| **Word File** | The file that contains all the words in the game. This is the games dictionary and should contain all possible words, each on a newline. |
| **Batch Mode** | Enables batch generation mode. Batch mode is used to created multiple randomly generated levels. |
| **Board Type** | The type of board this level is for. Grid will generate/display a grid of words, List will simply display the list of words (No grid). |
| **Preferred Board Ratio** | The ratio (rows by columns) that the grid should attempt to match. This is used to generate grids that best match the ratio to be displayed in. (Example 1x1 would create mostly squared grids where as 1x2 would created more portrait looking grids) |
| **Max Boards To Generate** | The way the grid is generate is by generating a bunch of random boards then picking the best one based on the Preferred Board Ratio. This is the maximum number of boards that the algorithm will generate before terminating. |

## Import Settings

This section of the window is used to import already created level files in order to edit them. To use, drag a level file into the **Level File** field and click **Import Level File**.

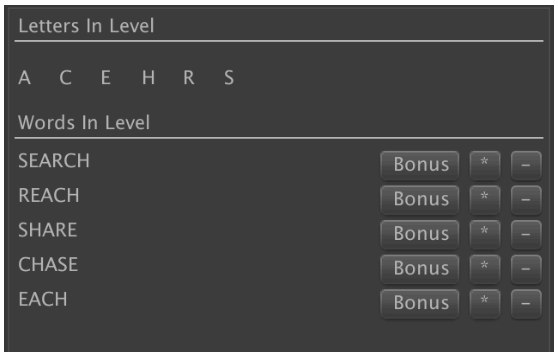


## Level Settings



| **Fields** | |
|---|---|
| **Letters In Level** | Sets the preferred number of letters for the level. This is the number of letters that will appear in the letter circle in the game. Words will be suggested in the section below based on this number. |
| **Custom Word** | Used to enter a word manually for the level instead of selecting one in the “Possible Words To Add” section. |

## Letters/Words In Level

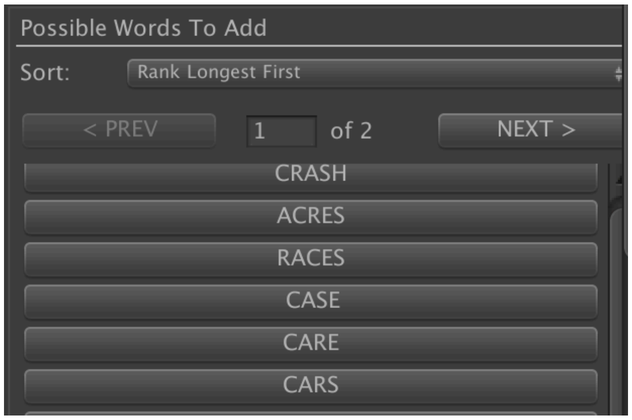


The **Letters In Level** section shows a preview of the letters that will appear in the letter circle when the game runs. The **Words In Level** section shows all the words that have been selected and will appear in the level. There are three buttons beside each word:

- **Bonus** : Clicking this button will assign the words as the bonus word. The bonus word will appear outside the grid for Grid type levels (This button does nothing for List type levels)

- *** [star]** : Clicking the star button will add coins to the word which when found in the game will award those coins to the player.
- **- [minus]** : Clicking the minus buttons will remove the word from the level.

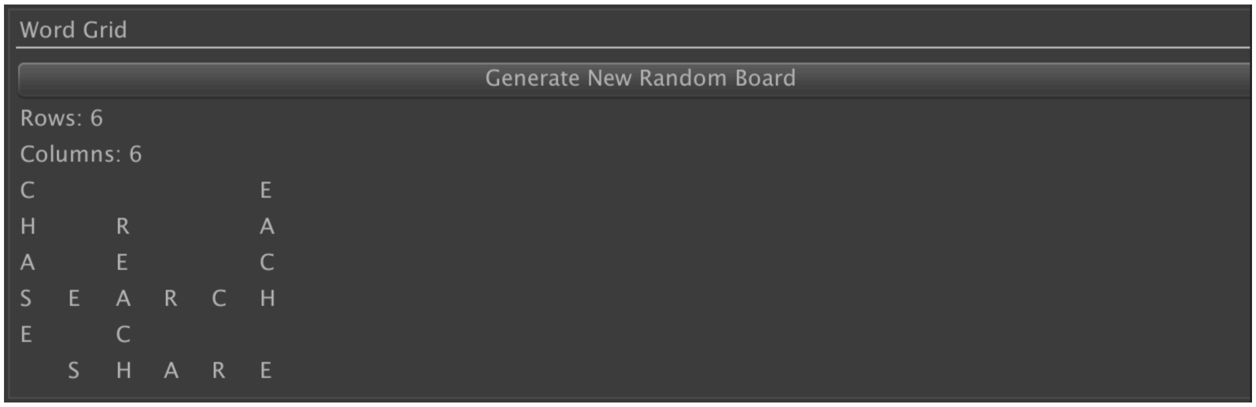
## Possible Words To Add



This section shows a list of words that can be added to the level that  wont go over the Letters In Level amount. This list can be sorted a number of ways to help find words of interest. **Rank** is the where the word appeared in the word file. The word file that comes with the asset is sorted by most used words in the English dictionary. The sort types are as follows:

- **Rank** : Sorts the words by rank, smallest to largest (Words that appear earlier in the word file)
- **Rank Shortest First** : Sorts the words by length then by rank so the shortest words appear first.
- **Rank Longest First** : Sorts the words by length then by rank so the longest words appear first.
- **Alphabetically** : Sorts the words alphabetically.
- **Alphabetically Shortest First** : Sorts the words by length then alphabetically so the shortest words appear first.
- **Alphabetically Longest First** : Sorts the words by length then alphabetically so the longest words appear first.

## Word Grid



Word Grid

Generate New Random Board

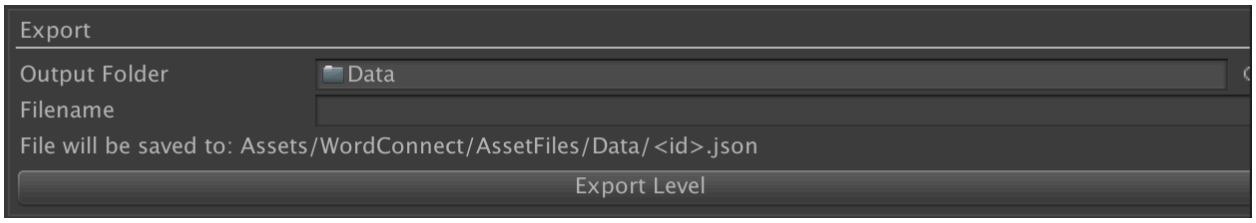
Rows: 6

Columns: 6

```
C                   E
H       R           A
A       E           C
S   E   A   R   C   H
E       C
    S   H   A   R   E
```

This section shows a preview of the grid if Grid type is selected. Clicking **Generate New Random Board** will generate a new board but it might not be different depending on the number of possible boards that could be generated since it is trying to find the most optimal board based on the settings in the General section.

## Export



Export

Output Folder: Data

Filename:

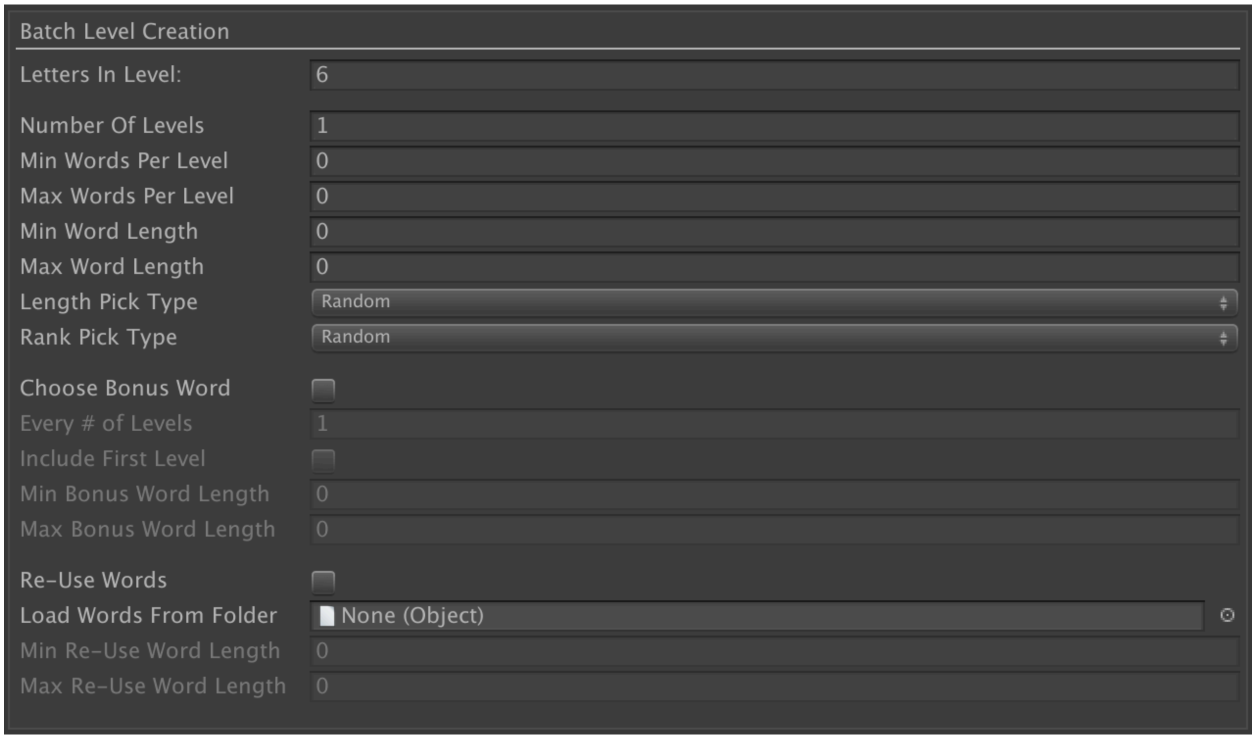
File will be saved to: Assets/WordConnect/AssetFiles/Data/<id>.json

Export Level

This section is used to export the level when all the desired words have been added. To set the **Output Folder** simply drag a folder from the Project window in Unity into the field. If a **Filename** has not been set the the id of the level will be used.

## Batch Mode

Selecting **Batch Mode** in the General settings will switch the window over to batch generation mode. Batch mode is used to create a multiple of randomly generated levels at once. The table below will describe all the settings for batch generation mode.



Batch Level Creation

| | |
|---|---|
| Letters In Level: | 6 |
| Number Of Levels | 1 |
| Min Words Per Level | 0 |
| Max Words Per Level | 0 |
| Min Word Length | 0 |
| Max Word Length | 0 |
| Length Pick Type | Random |
| Rank Pick Type | Random |
| Choose Bonus Word | ☐ |
| Every # of Levels | 1 |
| Include First Level | ☐ |
| Min Bonus Word Length | 0 |
| Max Bonus Word Length | 0 |
| Re-Use Words | ☐ |
| Load Words From Folder | None (Object) |
| Min Re-Use Word Length | 0 |
| Max Re-Use Word Length | 0 |

| **Fields** | |
|---|---|
| **Letters In Level** | Sets the number of letters for each level. This is the number of letters that will appear in the letter circle in the game. |
| **Number Of Levels** | The number of levels to generate. |
| **Min Words Per Level** | The minimum number of words each level must have. |
| **Max Words Per Level** | The maximum number of words each level can have. Words will continue to be chosen until this number has been reached. Setting this to 0 means there is no maximum, the algorithm will continue choosing words until there are no more valid words for the level. |
| **Min Word Length** | The minimum number of letters a words must have in order to be considered for a level. |
| **Max Word Length** | The maximum number of letters a words can have in order to be considered for a level. |

| **Fields** | |
|---|---|
| **Length Pick Type** | Determines the behavior when choosing words for a level.<br><br>- **Random** means length is not taking into consideration when choosing a letter.<br>- **Longest First** mean the longest possible word will be chosen first.<br>- **Shortest First** means the shortest possible word will be chosen first.<br>- **Distributed Longest First** means the longest word will be chosen first, followed by the second longest, then the third, etc<br>- **Distributed Shortest First** is the opposite of Distributed Longest First meaning the shortest word is chosen first then the second shortest, all the way to the longest word at which point it wraps back to the shortest. |
| **Rank Pick Type** | Determines the behavior when choosing words for a level. First **Length Pick Type** is used to get a list of words by length to choose from then **Rank Pick Type** is used to choose which word from that list is chosen.<br><br>- **Random** means a random word is chosen form the list.<br>- **Most Common First** means the word with the least Rank value (Appears first in the word file) will be chosen.<br>- **Least Common First** means the word with the highest Rank value (Appears last in the word file) will be chosen. |
| **Choose Bonus Word** | If selected then a bonus word will be chosen for the level. The bonus word in batch generation mode will also be starred meaning it will have coins on it when the level is played in the game. |
| **Every # of Levels** | Determines which levels get the bonus word. For example, setting to 1 means every level will choose a bonus word, setting to 8 means every 8th level will choose a bonus word (Level 8, level 16, etc) |
| **Include First Level** | If selected then the first level generate will get a bonus word. |
| **Min Bonus Word Length** | The minimum word length for a bonus word. |
| **Max Bonus Word Length** | The maximum word length for a bonus word. |
| **Re-Use Words** | If selected then words will be re-used in levels, if false then words that have been used in previous levels will not be re-used again. |
| **Load Words From Folder** | Drag a folder with already generated level files into this field and when the batch generation starts all level files in this folder will be loaded and their words will be considered used when generating new levels. |

| Fields | |
|---|---|
| **Min Re-Use Word Length** | The minimum length a word must be in order to be re-used. For example, setting to 4 means all words of length 2 and 3 will not be re-used only words greater than or equal to 4. |
| **Max Re-Use Word Length** | The maximum length a word can be in order to be re-used. For example, setting to 6 means all words or length greater than 6 will not be re-used. |

## Sounds

Sounds in the game are controlled using the SoundManager. On the SoundManager's inspector you will find a number of Sounds Infos already created and used in the game.

### Sound Info fields

**Id** - The Id used to play the sound in the game.
**Audio Clip** - The sound file from your project.
**Type** - The type of sound (Sound Effect or Music), this is used to turn on/off all sounds of a particular type.
**Play And Loop On Start** - If selected the Audio Clip will play when the game starts and will loop forever unless it is stopped.
**Clip Volume** - Sets the volume of the sound when it is played.

### Playing Sounds

Sounds can be played by calling the **Play** method on the SoundManager like so:

```
SoundManager.Instance.Play(string id);
```

You can easily play a sound when a Button is clicked by adding the **ButtonSound** component to a GameObject with a **Button** component. The ButtonSound will play the sound with the specified Id every time the button is clicked.

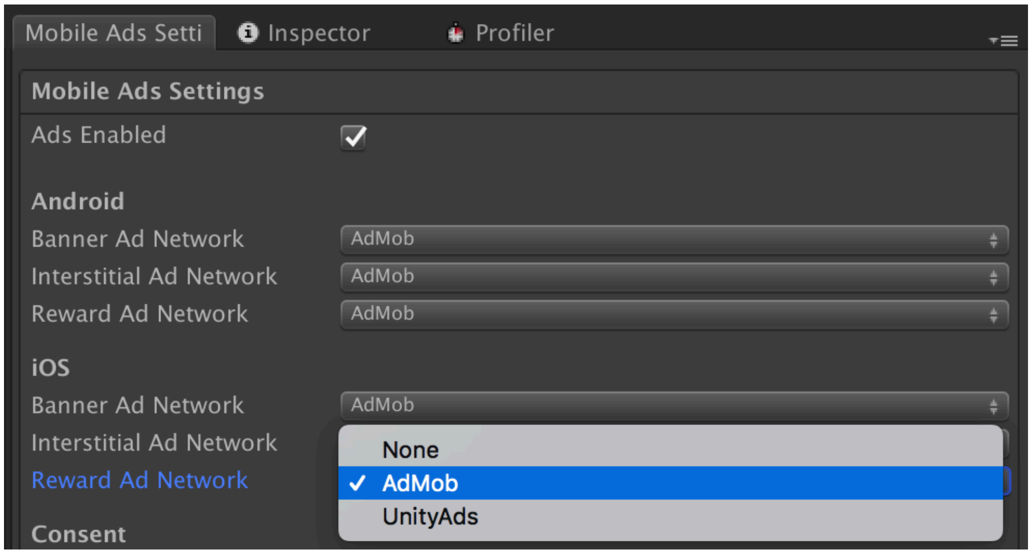
# Ads

Ads are setup using the **Mobile Ads Settings** window which can be opened by selecting the menu item **Window -> Mobile Ads Settings** (Or clicking the button on the MobileAdsManager inspector).

On the Mobile Ads Settings window you can select either **AdMob** or **Unity Ads** to be used for Banner, Interstitial, and/or Reward ads for both Android and iOS platforms. Selecting **None** will disable ads for that platform / ad type.

## AdMob Setup

**Step1.** Select AdMob in one or more of the drop downs.



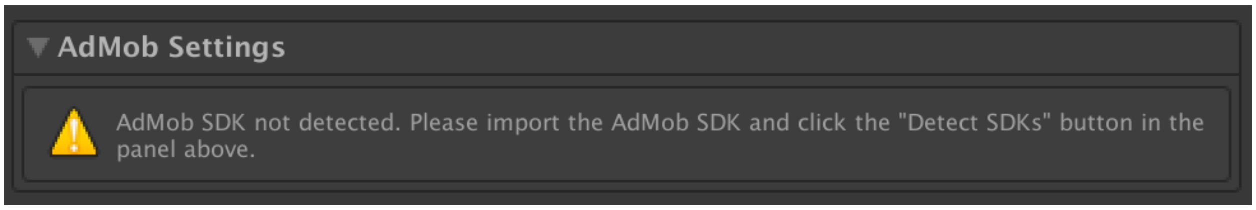
Mobile Ads Settings
Ads Enabled ✓

Android
Banner Ad Network: AdMob
Interstitial Ad Network: AdMob
Reward Ad Network: AdMob

iOS
Banner Ad Network: AdMob
Interstitial Ad Network
Reward Ad Network

Consent

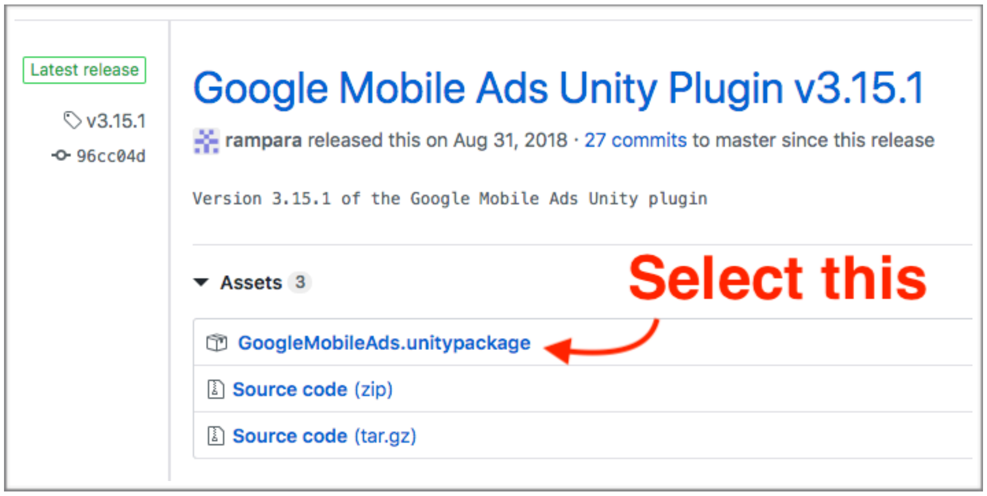
A new section will appear at the bottom of the window called **AdMob Settings**. Expanding it now will display the following warning:


AdMob Settings

AdMob SDK not detected. Please import the AdMob SDK and click the "Detect SDKs" button in the panel above.

**Step2.** Download and import the AdMob Unity SDK by clicking on this link **https://github.com/googleads/googleads-mobile-unity/releases** and clicking the GoogleMobileAds.unitypackage

Download the GoogleMobileAds.unitypackage:



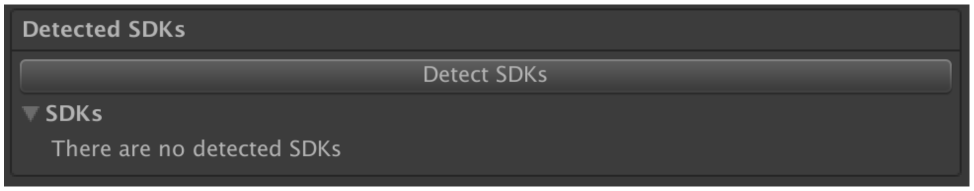
Google Mobile Ads Unity Plugin v3.15.1

rampara released this on Aug 31, 2018 · 27 commits to master since this release

```
Version 3.15.1 of the Google Mobile Ads Unity plugin
```

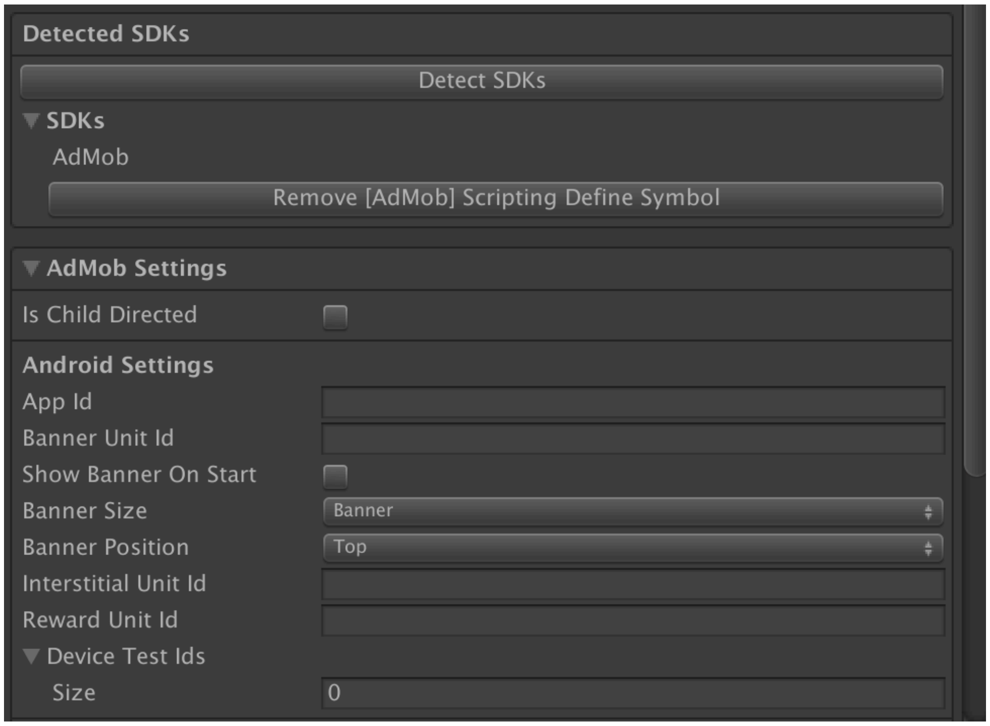
- GoogleMobileAds.unitypackage
- Source code (zip)
- Source code (tar.gz)

**Step3.** Once the GoogleMobileAds.unitypackge has finished importing into Unity click the Detect SDKs button on the Mobile Ads Settings window:



There are no detected SDKs

After Unity finishes compiling AdMob should appear under the SDKs list and the AdMob fields should appear under AdMob Settings



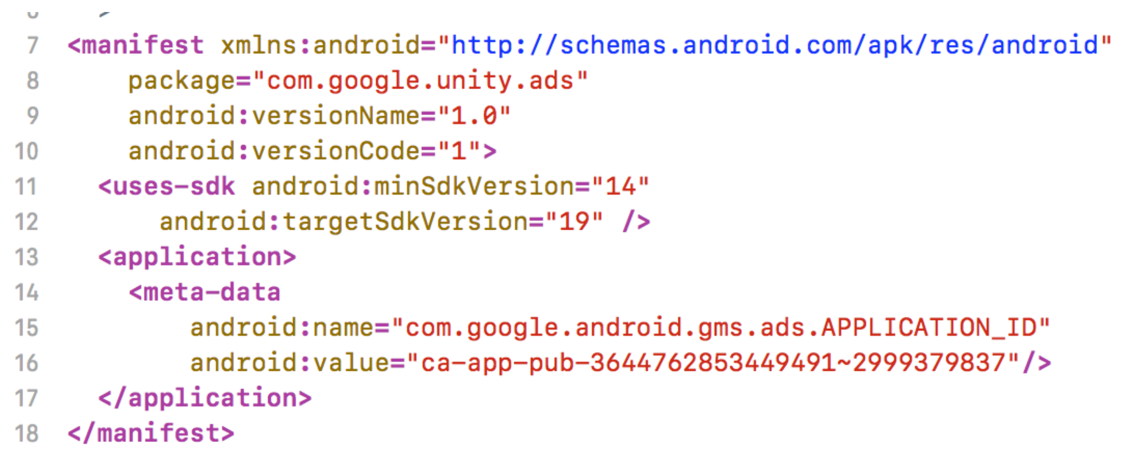
| | |
|---|---|
| Is Child Directed | ☐ |
| **Android Settings** | |
| App Id | |
| Banner Unit Id | |
| Show Banner On Start | ☐ |
| Banner Size | Banner |
| Banner Position | Top |
| Interstitial Unit Id | |
| Reward Unit Id | |
| Device Test Ids | |
| Size | 0 |

**Step4.** Add you App Id and Unit Ids to the fields under AdMob Settings

**Step5 [Android only].** Open the AndroidManifest located in the folder Plugins/Android/GoogleMobileAdsPlugin and add the following lines in the **application** element, replace ADMOB_APP_ID with your App Id:

```
<meta-data
        android:name="com.google.android.gms.ads.APPLICATION_ID"
        android:value="ADMOB_APP_ID"/>
```

Your AndroidManifest should look something like this:



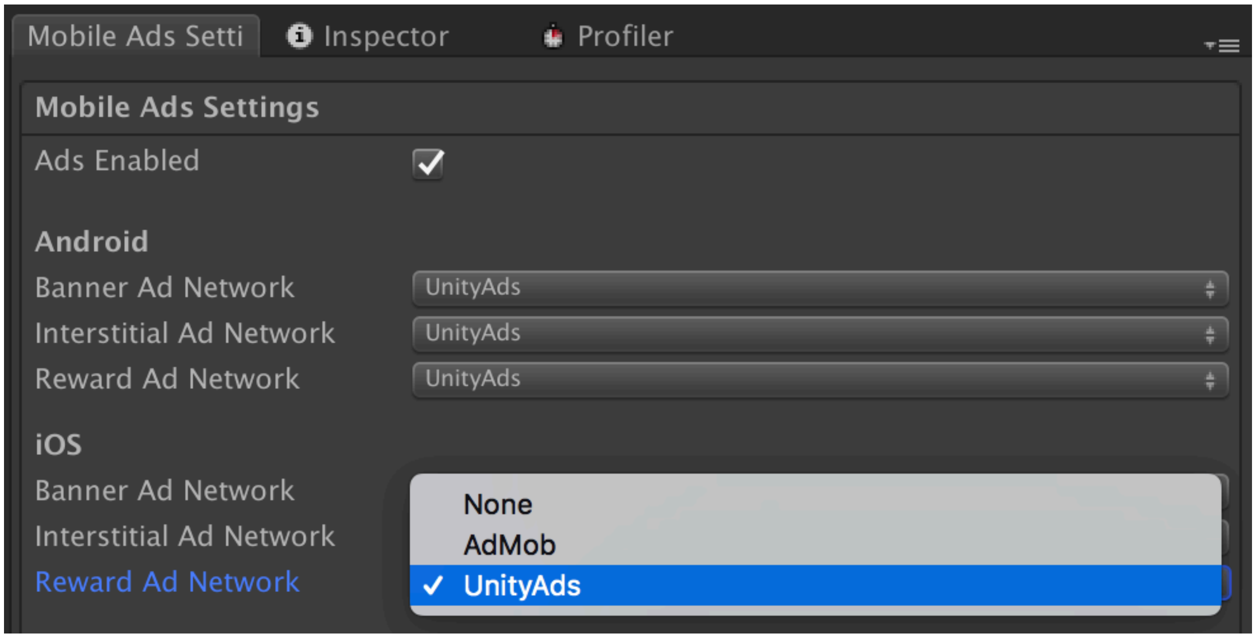
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.unity.ads"
    android:versionName="1.0"
    android:versionCode="1">
  <uses-sdk android:minSdkVersion="14"
      android:targetSdkVersion="19" />
  <application>
    <meta-data
        android:name="com.google.android.gms.ads.APPLICATION_ID"
        android:value="ca-app-pub-3644762853449491~2999379837"/>
  </application>
</manifest>
```

**Step6 [Android Only].** Make sure the play services resolver that comes with the GoogleMobileAds plugin has executed by selecting the menu item **Assets -> Play Services Resolver -> Android Resolver -> Resolve**.

Thats it! AdMob ads should appear in the game.

## Unity Ads Setup

**Step1.** Select AdMob in one or more of the drop downs.



Mobile Ads Settings

Ads Enabled ✓

Android

Banner Ad Network: UnityAds

Interstitial Ad Network: UnityAds

Reward Ad Network: UnityAds

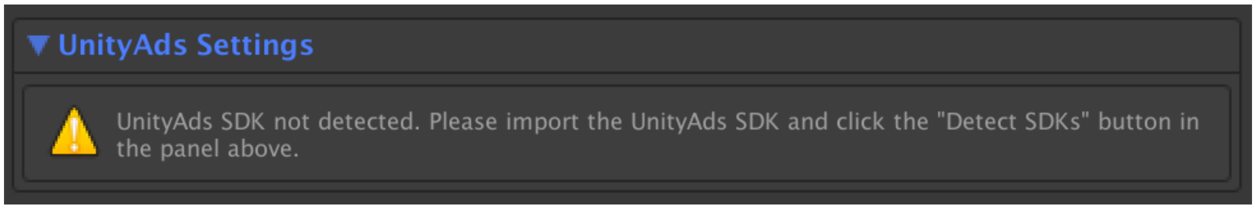
iOS

Banner Ad Network

Interstitial Ad Network

Reward Ad Network: None / AdMob / ✓ UnityAds

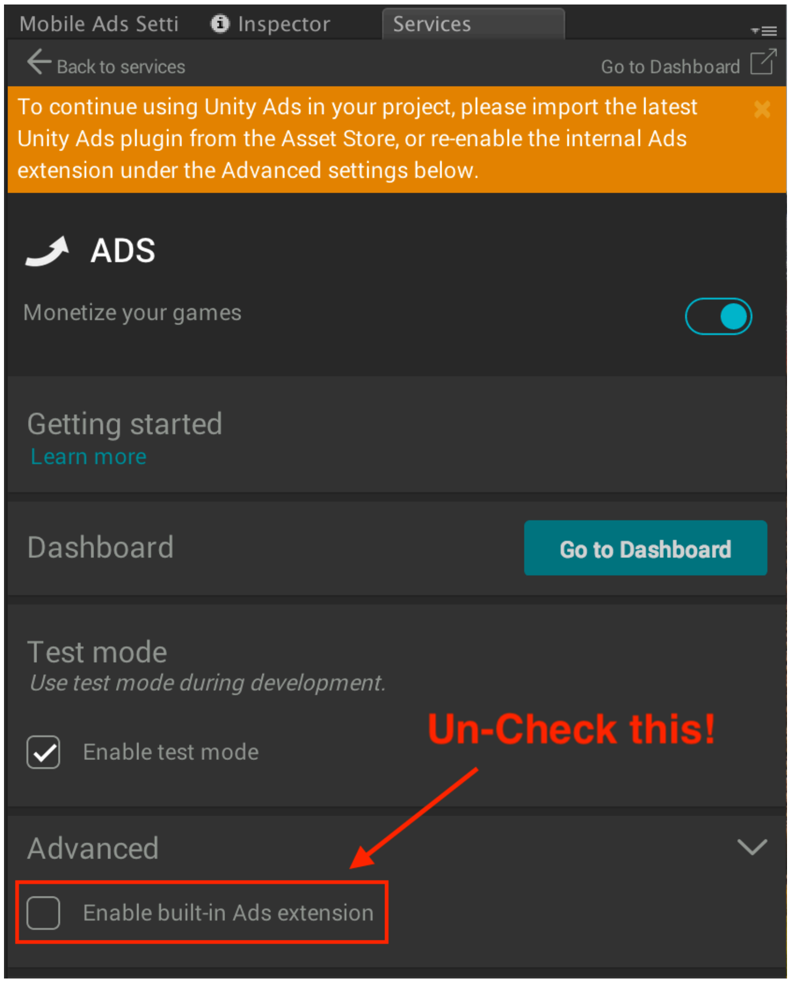
A new section will appear at the bottom of the window called **UnityAds Settings**. Expanding it now will display the following warning:
**Step2.** Enable Ads in the Services window:



UnityAds Settings

UnityAds SDK not detected. Please import the UnityAds SDK and click the "Detect SDKs" button in the panel above.

*** **IMPORTANT** *** Make sure "Enable built-in Ads Extension" is disabled or it will collide with the Monetization plugin you will import in the next step. To do so expand the **Advanced** section and un-check the field if it is checked



Mobile Ads Setti | Inspector | Services

Back to services | Go to Dashboard

To continue using Unity Ads in your project, please import the latest Unity Ads plugin from the Asset Store, or re-enable the internal Ads extension under the Advanced settings below.

ADS

Monetize your games

Getting started
Learn more

Dashboard | Go to Dashboard

Test mode
*Use test mode during development.*

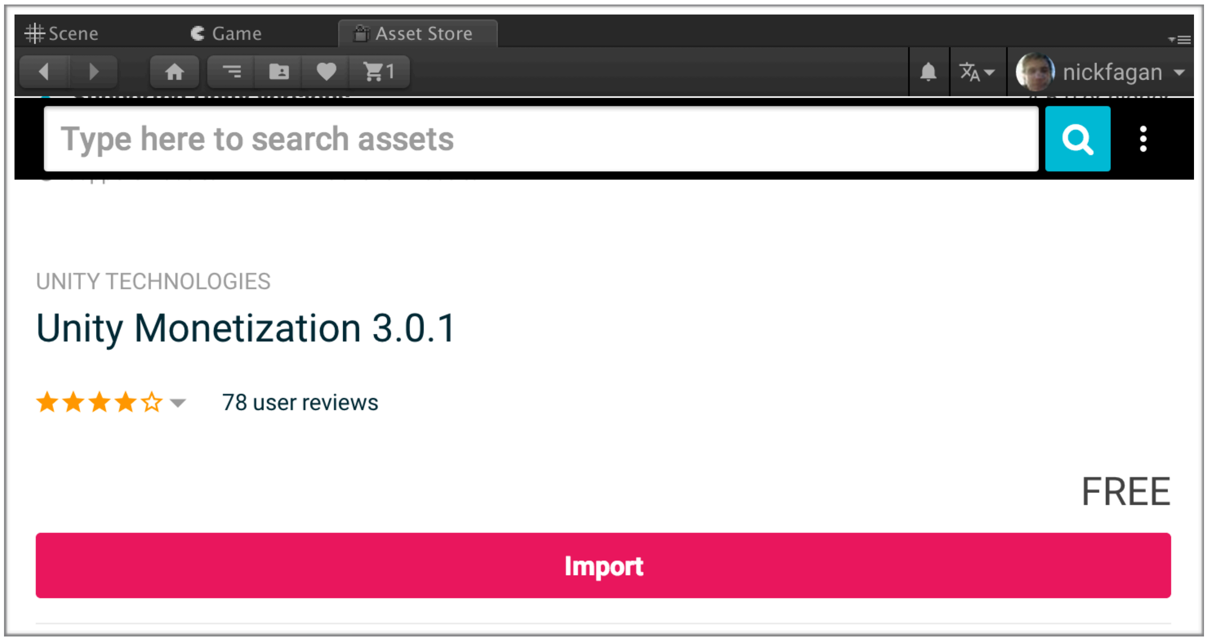
☑ Enable test mode

Un-Check this!

Advanced

☐ Enable built-in Ads extension

**Step2.** Open the Asset Store window and Download/Import the Unity Monetization asset:



Scene | Game | Asset Store

nickfagan

Type here to search assets

UNITY TECHNOLOGIES

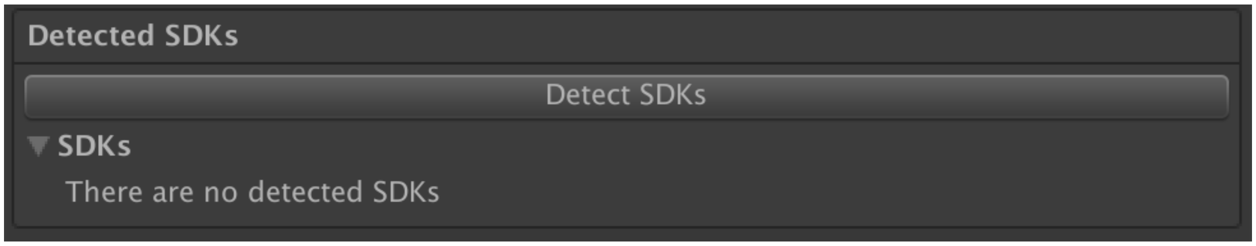
Unity Monetization 3.0.1

★★★★☆ 78 user reviews

FREE

Import

**Step3.** Click the **Detect SDKs** button on the Mobile Ads Settings window:



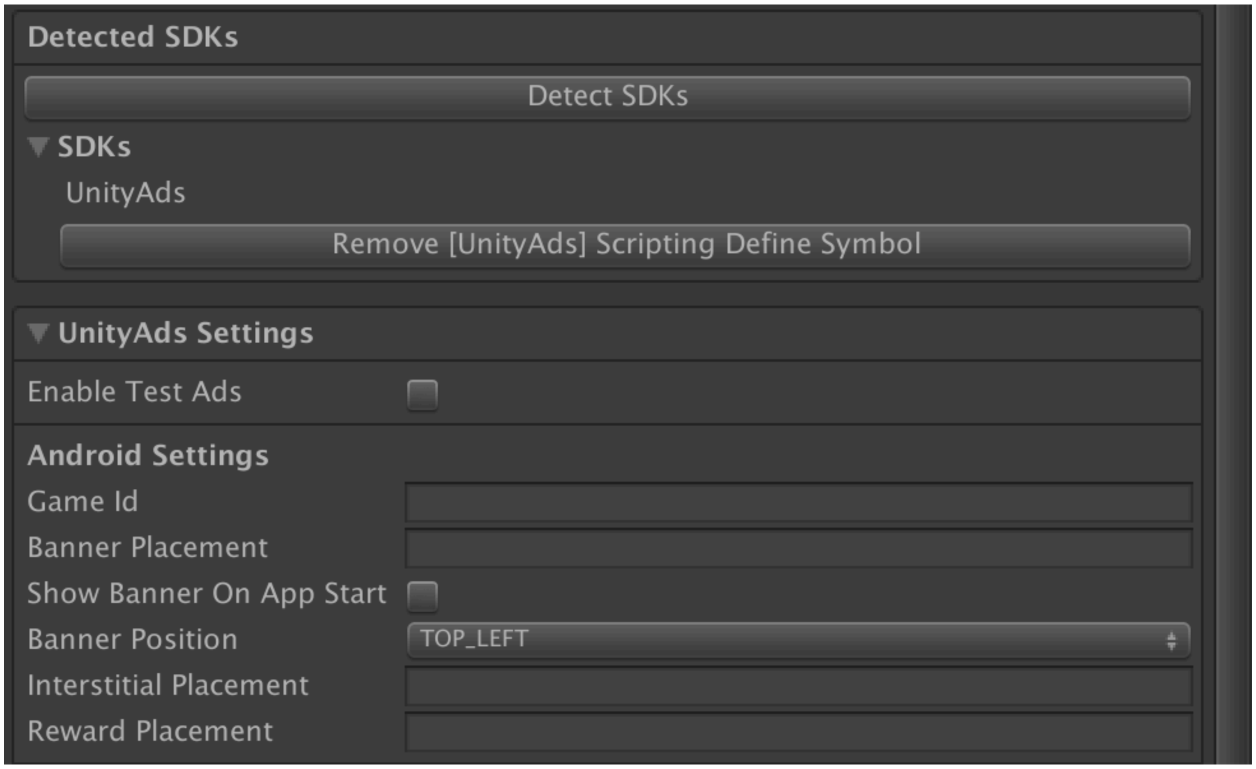
Detected SDKs

Detect SDKs

SDKs

There are no detected SDKs

After Unity finishes compiling, UnityAds should appear under the SDKs list and the UnityAds fields should appear under UnityAds Settings:



Detected SDKs

Detect SDKs

SDKs

UnityAds

Remove [UnityAds] Scripting Define Symbol

UnityAds Settings

Enable Test Ads ☐

Android Settings

| | |
|---|---|
| Game Id | |
| Banner Placement | |
| Show Banner On App Start | ☐ |
| Banner Position | TOP_LEFT |
| Interstitial Placement | |
| Reward Placement | |

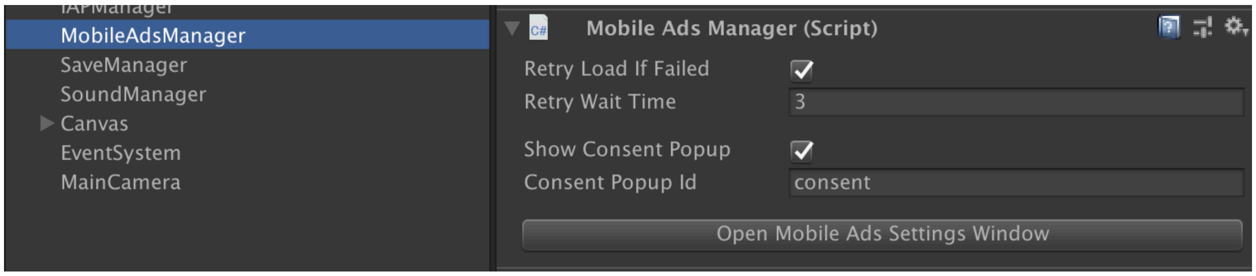
**Step4.** Add you Game Id and Placement Ids to the fields under UnityAds Settings.

Thats it! Unity Ads will now appear in your game.

## Showing Ads

The game is already setup to show Banner ads on every screen, Interstitial ads when the player starts a level, and a Reward ad button on the Game Screen. The frequency of Interstitial ads can be controlled on the GameControllers inspector.

The MobileAdsManager script is used to show ads in your game.



Retry Load If Failed ✓
Retry Wait Time 3
Show Consent Popup ✓
Consent Popup Id consent

The MobileAdsManager can be accessed anywhere in you projects scripts using the static **Instance** property like so: **MobileAdsManager.Instance**

If the **Retry Load If Failed** is checked then when an ad fails to load for any reason (no internet connection, no ad fill, etc) the the MobileAdsManager will wait the amount of seconds specified in **Retry Wait Time** then it will try and load the ad again. If Retry Load If Failed is un-check then when an ad fails to load you will have to manually call one of the load methods.

If the **Show Consent Popup** is checked then when the app starts, if consent is required for the user, it will show the popup with the given **Consent Popup Id**.

**NOTE:** If one of the show methods (ShowInterstitialAd, ShowRewardAd) is called and there is no pre-loaded ad then the MobileAdsManager will attempt to load an ad so the next time the show method is called there will be an ad ready to show.

### Showing Banner Ads

To show banner ads call the **ShowBannerAd** method on the MobileAdsManager Instance:

```
MobileAdsManager.Instance.ShowBannerAd();
```

You can hide the banner ad by calling the **HideBannerAd** method.

### Showing Interstitial Ads

To show interstitial Ads call the **ShowInterstitialAd** method on the MobileAdsManager Instance:

```
MobileAdsManager.Instance.ShowInterstitialAd();
```

The ShowInterstitialAd method can be passed a callback method that is invoked when the interstitial ad closes and is no longer displayed on the screen.

The ShowInterstitialAd method returns a boolean indicating if an interstitial ad has been pre-loaded and is going to show. If the method returns false then there was no interstitial ad loaded so no ad will be displayed to the user. If it returns true then an interstitial ad has been pre-loaded an is about to show.

### Showing Reward Ads

To show interstitial Ads call the **ShowRewardAd** method on the MobileAdsManager Instance:

```
MobileAdsManager.Instance.ShowRewardAd();
```

The ShowReward method can be passed two callback methods:

**onClosedCallback** - Invoked when the reward ad closes and is no longer displayed on the screen.
**onRewardGrantedCallback** - Invoked when the user should be granted a reward for watching the reward ad. This method is a reward id and reward amount.

The onClosedCallback will always be called after a reward ad is shown and is called when the reward ad is no longer displayed on the screen. The onRewardGrantedCallback will only be called if the user watched the ad and should be granted the reward, if the user closed the ad in any way then onRewadGrantedCallback will not be invoked. (onRewadGrantedCallback is always invoked before onClosedCallback).

The ShowReward method returns a boolean indicating if a reward ad has been pre-loaded and is going to show. If the method returns false then there was no reward ad loaded so no ad will be displayed to the user. If it returns true then an reward ad has been pre-loaded an is about to show.

## Events

There are a number of ad events you can listen to on the MobileAdsManager script. To listen for one of the events add the following code anywhere in your project and replace OnEvent with the event you wish to listen for:

```
MobileAdsManager.Instance.OnEvent += YourEventMethod;
```

### Banner Events

**OnBannerAdLoading** - Invoked when a banner ad starts loading.
**OnBannerAdLoaded** - Invoked when a banner ad has loaded successfully.
**OnBannerAdFailedToLoad** - Invoked when a banner ad fails to load.
**OnBannerAdShown** - Invoked when the banner ad is shown on the screen.
**OnBannerAdHidden** - Invoked when the banner ad is hidden from the screen.

### Interstitial Events

**OnInterstitialAdLoading** - Invoked when an interstitial ad starts loading.
**OnInterstitialAdLoaded** - Invoked when an interstitial ad has successfully loaded.
**OnInterstitialAdFailedToLoad** - Invoked when an interstitial ad failed to load.
**OnInterstitialAdShowing** - Invoked when an interstitial ad is about to show on the screen.
**OnInterstitialAdShown** - Invoked when an interstitial ad has been displayed on the screen.
**OnInterstitialAdClosed** - Invoked when an interstitial ad has closed and is no longer displayed on the screen.

### Reward Events

**OnRewardAdLoading** - Invoked when an reward ad starts loading.
**OnRewardAdLoaded** - Invoked when an reward ad has successfully loaded.
**OnRewardAdFailedToLoad** - Invoked when an reward ad failed to load.
**OnRewardAdShowing** - Invoked when an reward ad is about to show on the screen.
**OnRewardAdShown** - Invoked when an reward ad has been displayed on the screen.
**OnRewardAdClosed** - Invoked when an reward ad has closed and is no longer displayed on the screen.
**OnRewardAdGranted** - Invoked when a reward ad has been watched and the reward should be granted to the user. This event is passed the reward id and reward amount that is set on the ad networks dashboard.

## Consent

Consent can be required before any ads are loaded by setting the **Consent Setting** on the Mobile Ads Manager. There are three types you can set the consent setting to:

**Not Required** - Consent is not required for ads to be loaded.

**Required Only In EEA** - Consent is only required for users in the European Economic Area. When the app starts it first attempts to determine if the user is located in the EEA and if so ads will not be loaded until the consent status has been set to either Personalized or Non-Personalized. If the user location can not be determined for any reason then it errs on the side of caution and requires consent before ads are loaded.

**Require All** - Consent is required for all users before ads are loaded.

### Setting the Consent Status

If consent is required before ads are loaded then the **SetConsentStatus** method must be called on the MobileAdsManager to set the consent status to either Personalized or Non-Personalized ads.

To set the consent simply call the method like so:

```
MobileAdsManager.Instance.SetConsentStatus(consentStatus);
```

The consentStatus parameter is an integer value:

**1** - Indicates the user has consented to receive personalized ads
**0** - Indicates the user should only be shown non-personalized ads.

## Testing

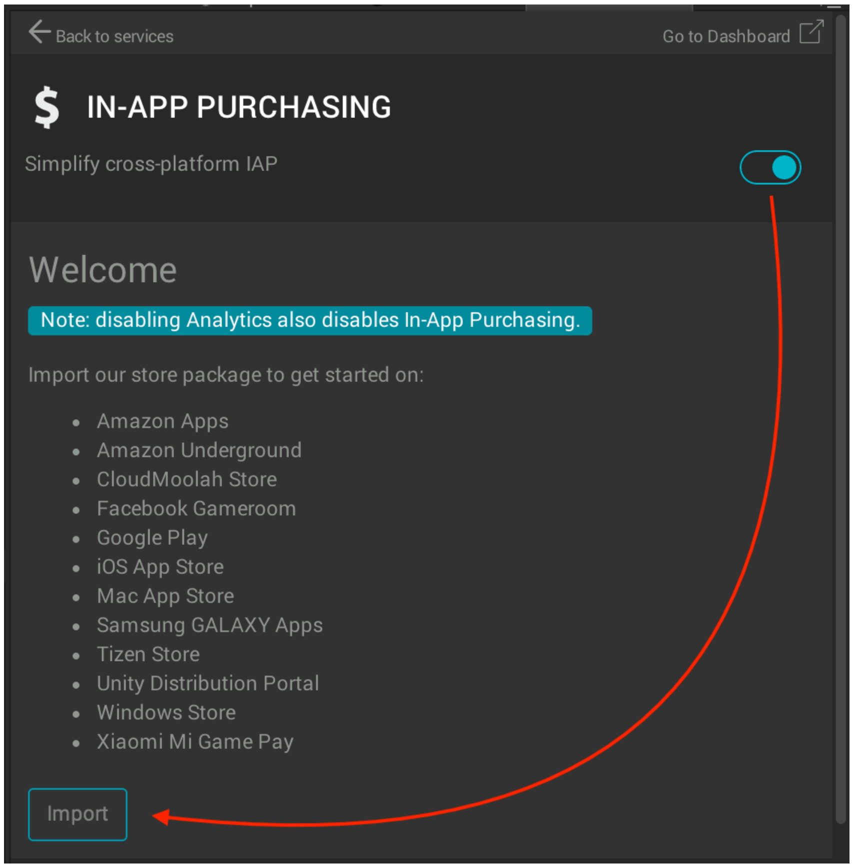
You should use the TestScene located in the Scenes folder to test if you have setup your ad networks properly. The test scene can be used to show ads and will print out log messages if there are any errors.

# IAP

IAP is setup using the **IAP Settings** window which can be opened by selecting the menu item **Window -> IAP Settings** (Or clicking the button on the IAPManagers inspector).

## Enable IAP

To enable IAP first you need to import the Unity plugin from the Services window. Open the Services window and turn on IAP then click the Import button:



Welcome

Note: disabling Analytics also disables In-App Purchasing.

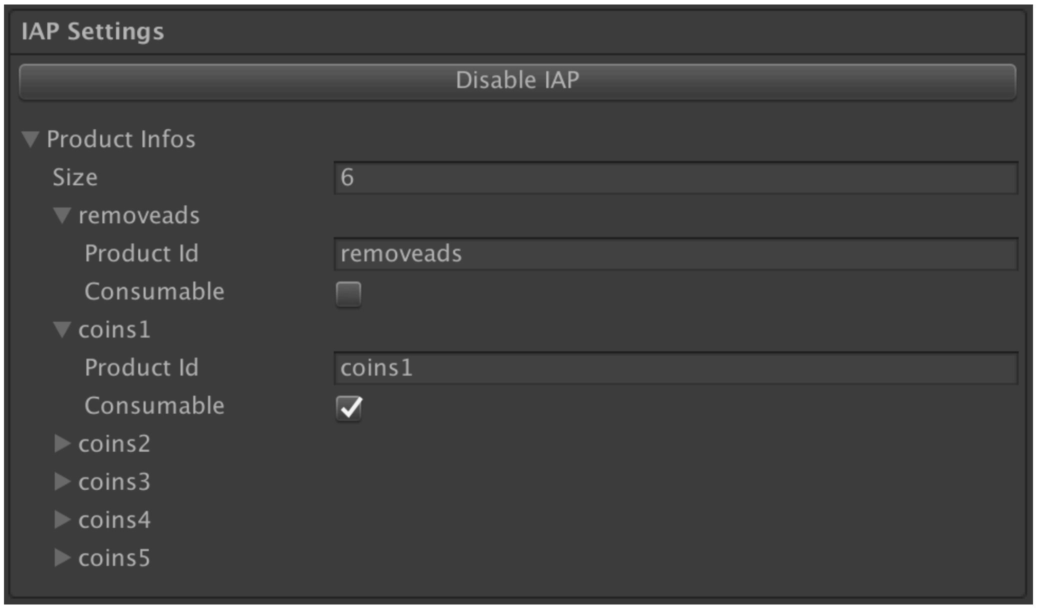
Import our store package to get started on:

- Amazon Apps
- Amazon Underground
- CloudMoolah Store
- Facebook Gameroom
- Google Play
- iOS App Store
- Mac App Store
- Samsung GALAXY Apps
- Tizen Store
- Unity Distribution Portal
- Windows Store
- Xiaomi Mi Game Pay

Once it has finished importing you can open the IAP Settings window and click the Enable IAP button which will enable the code in the project.

## Add Product Ids

To add products open the IAP Settings window and add a Product Info for each product in your game. The Word Blocks asset comes with six products already configured in the game, you can replace the assets product ids for your own or you can remove them and add new ones


IAP Settings

Disable IAP

Product Infos

Size: 6

removeads
- Product Id: removeads
- Consumable: ☐

coins1
- Product Id: coins1
- Consumable: ✓

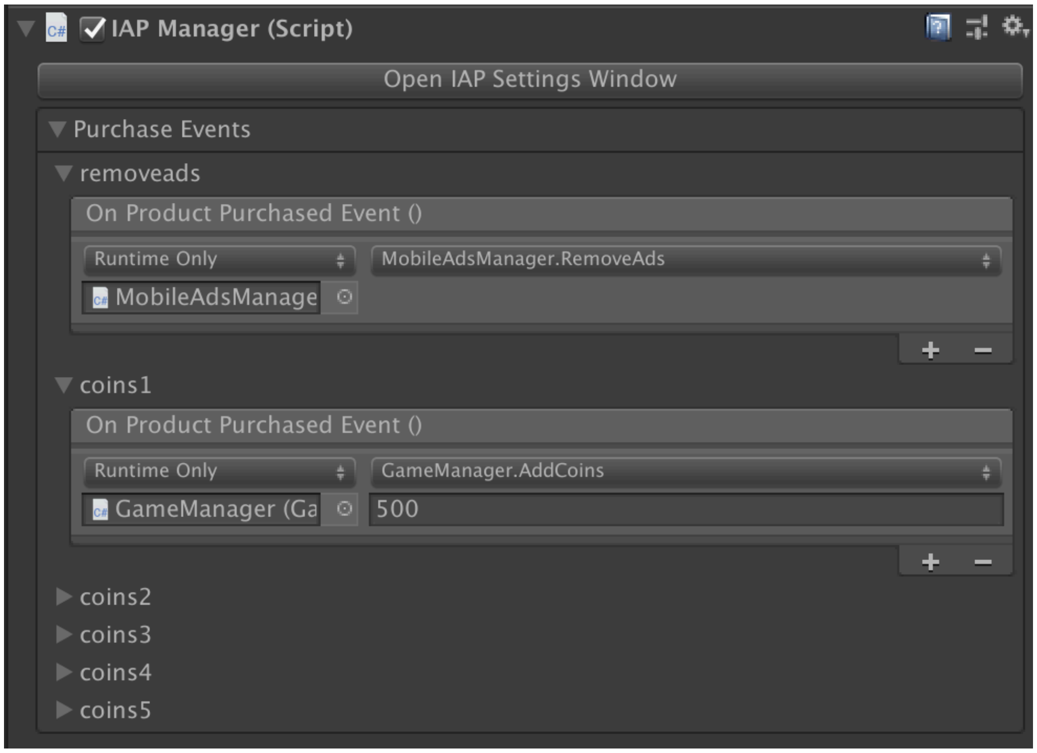
coins2

coins3

coins4

coins5

Once you add a new product an entry will appear on the IAPManager's inspector. Here you can add events for when the product is purchased:


IAP Manager (Script)

Open IAP Settings Window

Purchase Events

removeads
- On Product Purchased Event ()
- Runtime Only | MobileAdsManager.RemoveAds
- MobileAdsManage

coins1
- On Product Purchased Event ()
- Runtime Only | GameManager.AddCoins
- GameManager (Ga | 500

coins2

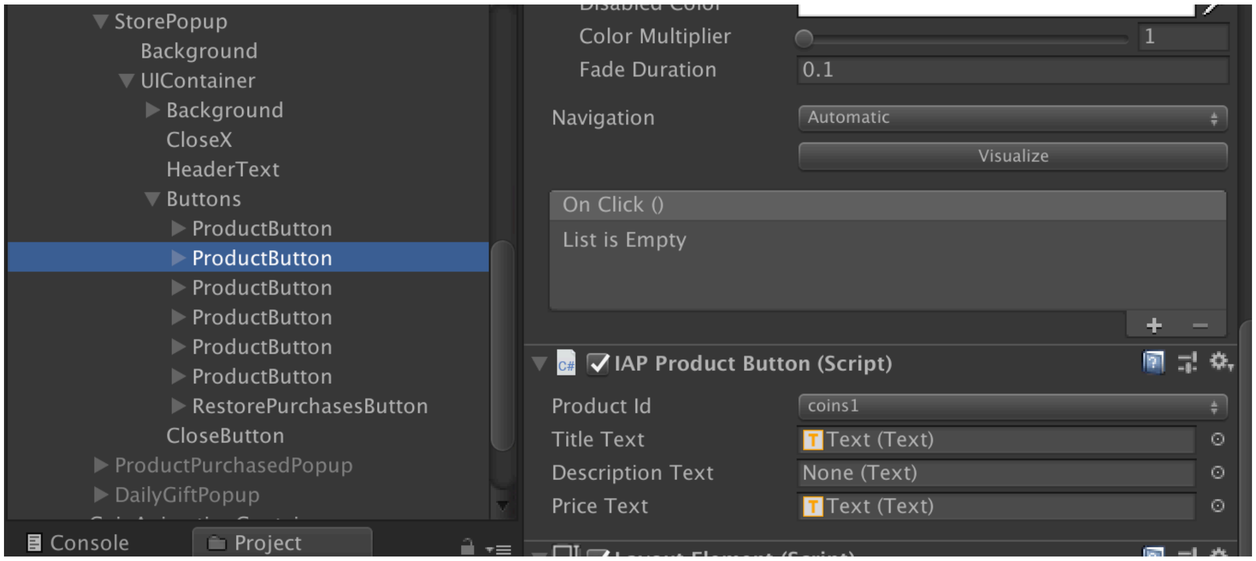
coins3

coins4

coins5

## Purchase A Product

To invoke the purchasing of a product you can use the **IAPProductButton** component.



```
▼ StorePopup
    Background
  ▼ UIContainer
    ▶ Background
      CloseX
      HeaderText
    ▼ Buttons
      ▶ ProductButton
      ▶ ProductButton
      ▶ ProductButton
      ▶ ProductButton
      ▶ ProductButton
      ▶ ProductButton
      ▶ RestorePurchasesButton
      CloseButton
▶ ProductPurchasedPopup
▶ DailyGiftPopup
```

| | |
|---|---|
| Color Multiplier | 1 |
| Fade Duration | 0.1 |
| Navigation | Automatic |

On Click ()
List is Empty

IAP Product Button (Script)

| | |
|---|---|
| Product Id | coins1 |
| Title Text | Text (Text) |
| Description Text | None (Text) |
| Price Text | Text (Text) |

The Product Id dropdown will contain all the products you created in the IAPSettings. When the button is clicked in the game the IAPManager's BuyProduct method will be called with selected Product Id.

You can also call the BuyProduct method manually like so:

**IAPManager.Instance.BuyProduct(string productId);**

You can also listen for successful product purchases with the OnProductPurchases action like so:

**IAPManager.Instance.OnProductPurchases += YourMethod;**

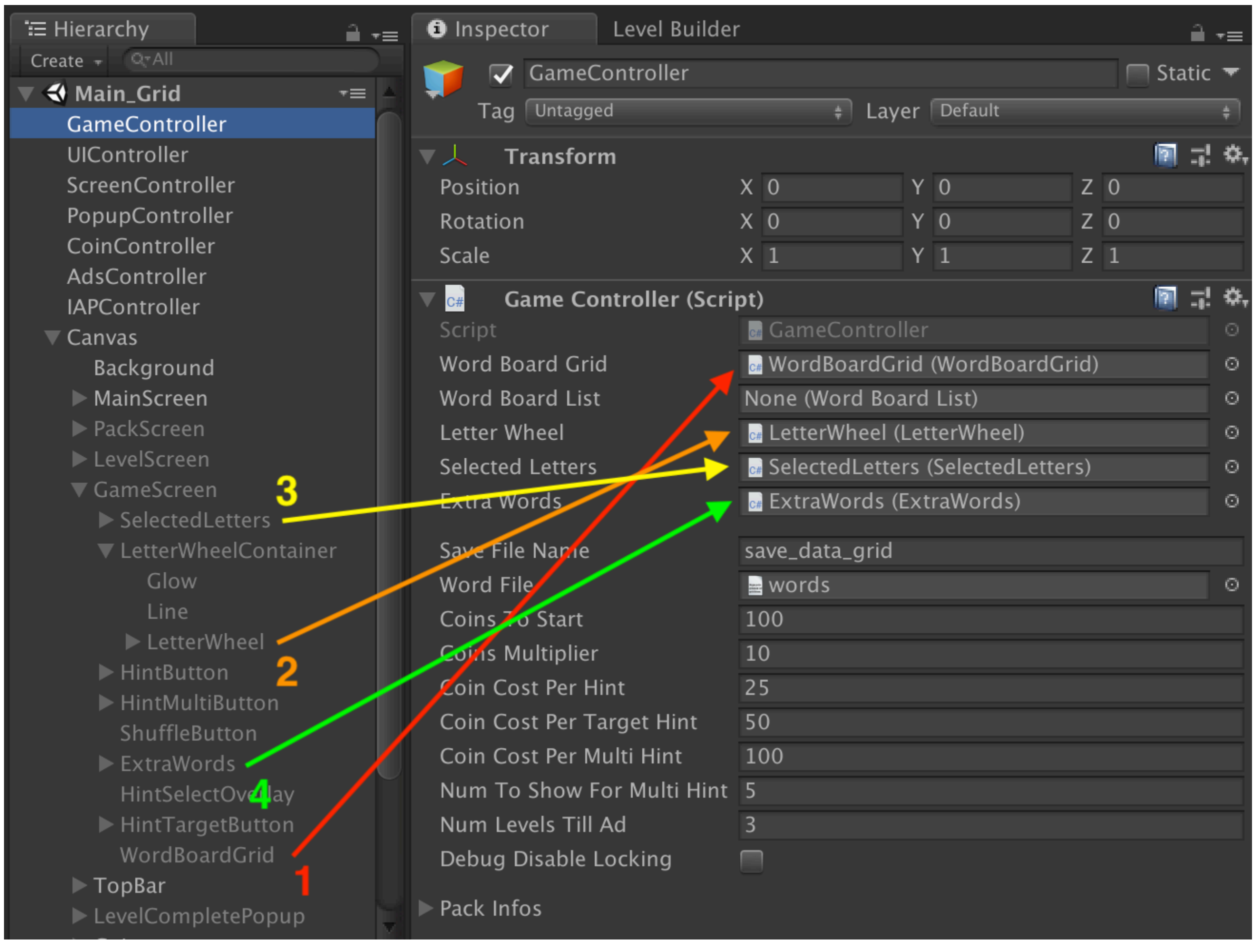
## Testing

You should use the TestScene located in the Scenes folder to test if you have setup IAP correctly. The scene will create a button for each product in the IAPSettings window which you can click to purchase the product. Logs will output on the screen to show and errors that may occur.

# Project Setup

## GameController

The GameController handles most of the core game logic. It's used to start levels and controls what happens when the user selects words or hints.



GameController — Static
Tag: Untagged — Layer: Default

| Transform | X | Y | Z |
|---|---|---|---|
| Position | 0 | 0 | 0 |
| Rotation | 0 | 0 | 0 |
| Scale | 1 | 1 | 1 |

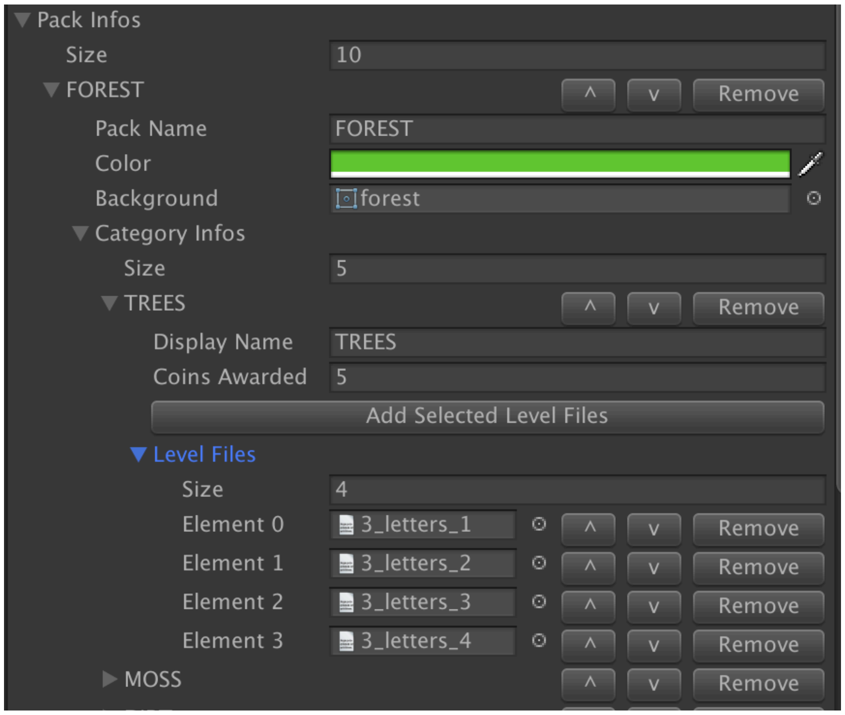
| Game Controller (Script) | |
|---|---|
| Script | GameController |
| Word Board Grid | WordBoardGrid (WordBoardGrid) |
| Word Board List | None (Word Board List) |
| Letter Wheel | LetterWheel (LetterWheel) |
| Selected Letters | SelectedLetters (SelectedLetters) |
| Extra Words | ExtraWords (ExtraWords) |
| Save File Name | save_data_grid |
| Word File | words |
| Coins To Start | 100 |
| Coins Multiplier | 10 |
| Coin Cost Per Hint | 25 |
| Coin Cost Per Target Hint | 50 |
| Coin Cost Per Multi Hint | 100 |
| Num To Show For Multi Hint | 5 |
| Num Levels Till Ad | 3 |
| Debug Disable Locking | ☐ |

**1 Word Board Grid** - The Word Board Grid handles displaying the main grid on the screen. It shows blanks cells for words that have not been found and when a word is found it will animate in the word onto the board.
**2 Letter Wheel** - The Letter Wheel component handles displaying the letters that are in the level and notifying a listener when letters are selected.
**3 Selected Letters** - The Selected Letters component handles displaying the letters that have been selected in the Letter Wheel.
**4 Extra Words** - The Extra Words component handles displaying the number of extra words that are found in the level.

**Other GameController Fields:**

**Save File Name** - The name the save file will have when the game saves.

**Word File** - The file that contains all the words in the game. This file is used when checking for extra words.
**Coins To Start** - The amount of coins that the player starts with when they first play the game.
**Coin Multiplier** - This multiple is applied whenever the player receives coins.
**Coin Cost Per Hint** - The amount of coins each hint costs.
**Coin Cost Per Target Hint** - The amount of coins each target costs.
**Coin Cost Per Multi Hint** - The amount of coins multi-hint costs.
**Num To Show For Multi Hint** - The amount of letters to show on the board when a multi-hint is used.
**Debug Disable Locking** - If this is selected then all levels will be playable on the level screen regardless of the progress. This only applies in the Unity editor, if the game is built to device this will be set to false.



```
▼ Pack Infos
    Size                10
  ▼ FOREST
      Pack Name         FOREST
      Color             🟩
      Background        forest
    ▼ Category Infos
        Size            5
      ▼ TREES
          Display Name    TREES
          Coins Awarded   5
          Add Selected Level Files
        ▼ Level Files
            Size        4
            Element 0   3_letters_1
            Element 1   3_letters_2
            Element 2   3_letters_3
            Element 3   3_letters_4
      ▶ MOSS
```

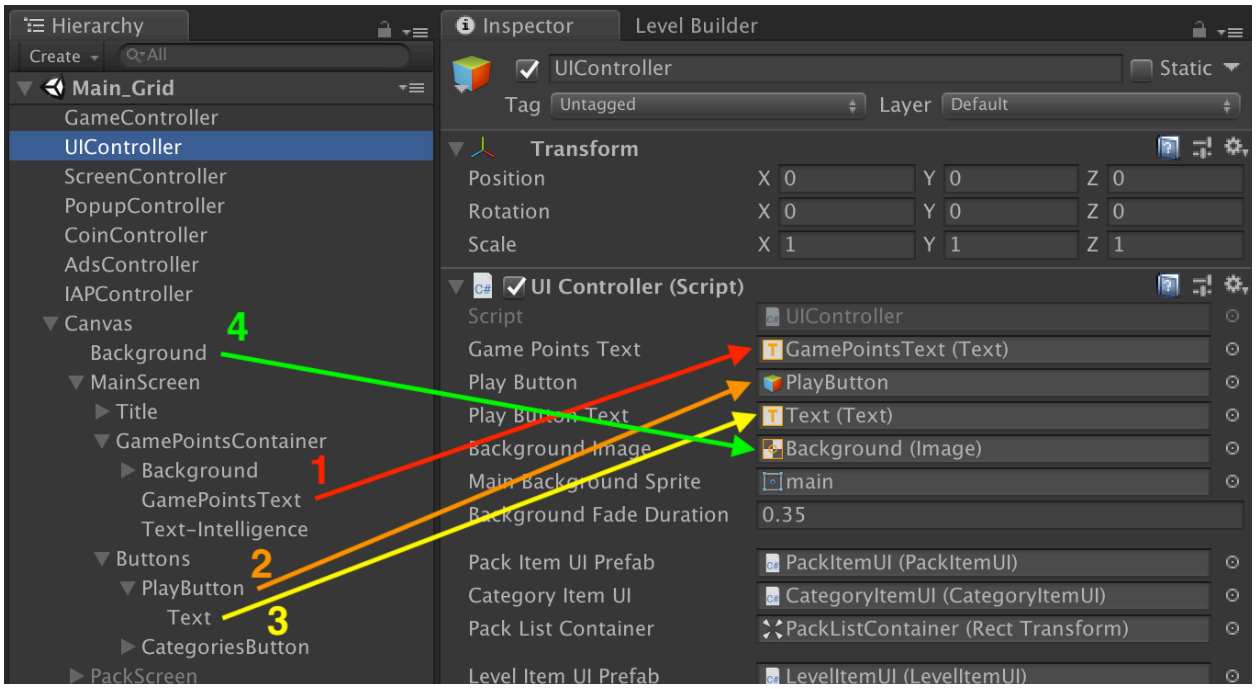
The game is split up into Packs/Categories/Levels. Packs contain any number of Categories and Categories contain any number of Levels. Levels Files are the files that are generated using the **Level Creator Window Tool**.

**<u>Pack Info:</u>**
**Pack Name** - The display name of the pack.
**Color** - The color of the pack. This color will be used for various elements on the pack screen game screen.
**Background** - This Sprite will be used as the background when a level in this pack is played.

**<u>Category Info:</u>**
**Display Name** - The display name of the category.
**Coins Awarded** - The amount of coins that are awarded when all levels in this category have been completed.

## UIController

The UIController handles updating most of the UI elements on various screens. It is also responsible for creating the Pack, Category, and Level UI list items.

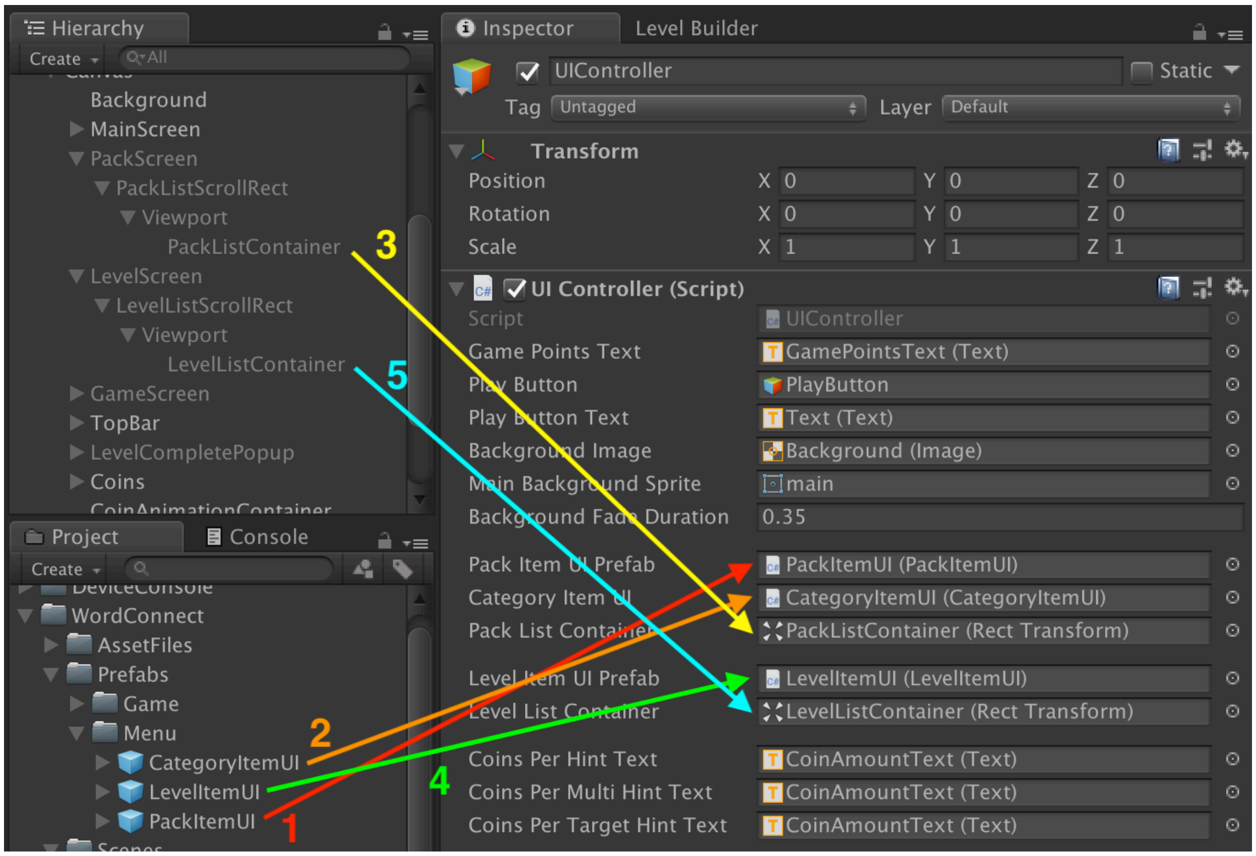


```
Main_Grid
  GameController
  UIController
  ScreenController
  PopupController
  CoinController
  AdsController
  IAPController
  Canvas
    Background
    MainScreen
      Title
      GamePointsContainer
        Background
        GamePointsText
        Text-Intelligence
      Buttons
        PlayButton
          Text
        CategoriesButton
    PackScreen
```

```
UIController                          Static
Tag  Untagged                         Layer  Default

Transform
Position      X 0    Y 0    Z 0
Rotation      X 0    Y 0    Z 0
Scale         X 1    Y 1    Z 1

UI Controller (Script)
Script                      UIController
Game Points Text            GamePointsText (Text)
Play Button                 PlayButton
Play Button Text            Text (Text)
Background Image            Background (Image)
Main Background Sprite      main
Background Fade Duration    0.35

Pack Item UI Prefab         PackItemUI (PackItemUI)
Category Item UI            CategoryItemUI (CategoryItemUI)
Pack List Container         PackListContainer (Rect Transform)

Level Item UI Prefab        LevelItemUI (LevelItemUI)
```

**1 Game Points Text** - The Text component on the Main screen to update when the player receives game points from completing levels.
**2 Play Button** - A reference to the Main screens play button, this buttons GameObject will be set to de-active when all levels in the game have been completed.
**3 Play Button Text** - The Text component that will display the next level number to play.
**4 Background** - The Image component that will display the background sprite for the current active level.

**Main Background Sprite** - The Sprite that will be used as the background on the Main screen and Pack screen.
**Background Fade Duration** - The animation duration in seconds for fading in a new background when the background Sprite changes.



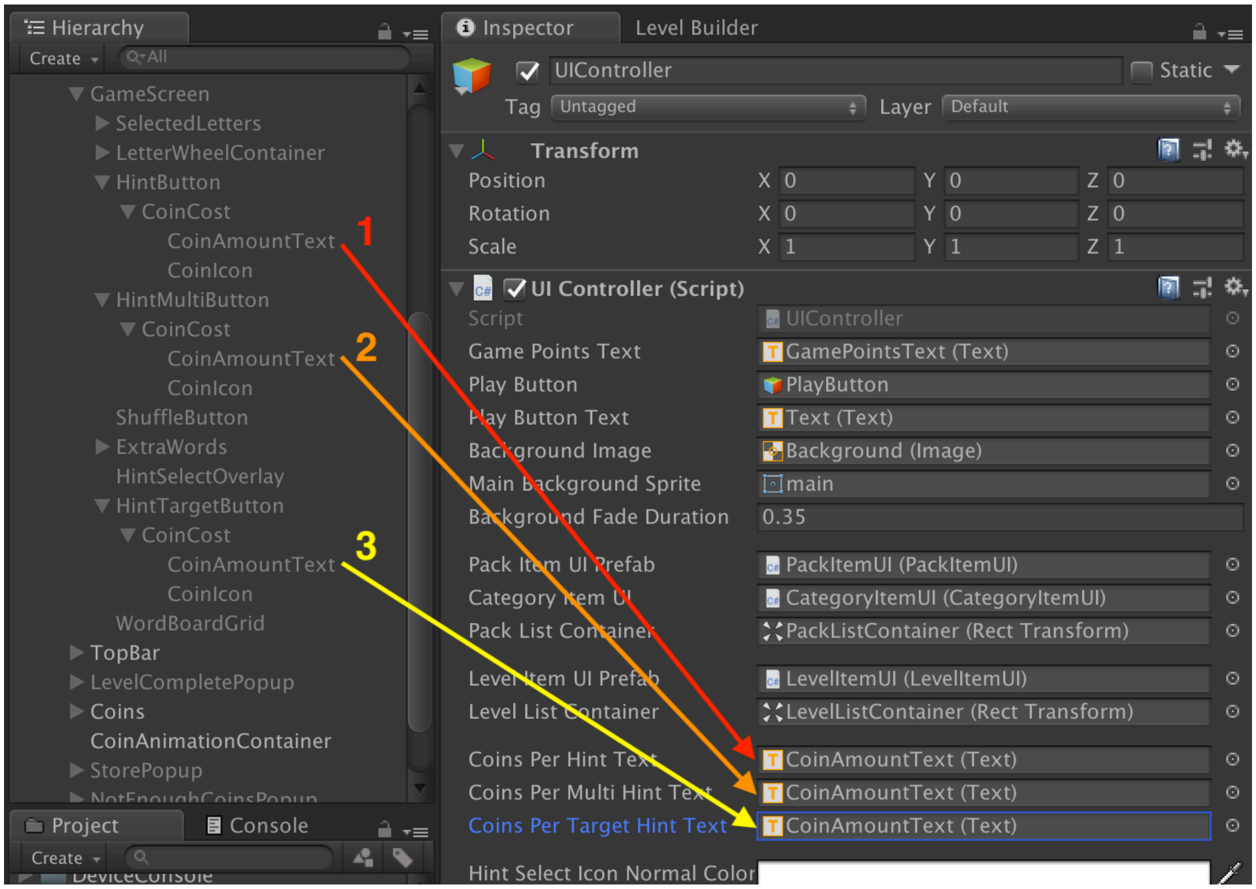
UIController
Tag Untagged
Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

UI Controller (Script)

| | |
|---|---|
| Script | UIController |
| Game Points Text | GamePointsText (Text) |
| Play Button | PlayButton |
| Play Button Text | Text (Text) |
| Background Image | Background (Image) |
| Main Background Sprite | main |
| Background Fade Duration | 0.35 |
| Pack Item UI Prefab | PackItemUI (PackItemUI) |
| Category Item UI | CategoryItemUI (CategoryItemUI) |
| Pack List Container | PackListContainer (Rect Transform) |
| Level Item UI Prefab | LevelItemUI (LevelItemUI) |
| Level List Container | LevelListContainer (Rect Transform) |
| Coins Per Hint Text | CoinAmountText (Text) |
| Coins Per Multi Hint Text | CoinAmountText (Text) |
| Coins Per Target Hint Text | CoinAmountText (Text) |

**1 Pack Item UI Prefab** - The prefab that will be used to create pack UI items for each pack in the game.
**2 Category Item UI Prefab** - The prefab that will be used to create category UI items for each category in the game.
**3 Pack List Container** - The Transform that pack UI items will be played in.
**4 Level Item UI Prefab** - The prefab that will be used to create level UI items for each level in the game.
**5 Level List Container** - The Transform that category UI items will be played in.



UIController  Static

Tag Untagged  Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

UI Controller (Script)

Script: UIController

Game Points Text: GamePointsText (Text)

Play Button: PlayButton

Play Button Text: Text (Text)

Background Image: Background (Image)

Main Background Sprite: main

Background Fade Duration: 0.35

Pack Item UI Prefab: PackItemUI (PackItemUI)

Category Item UI: CategoryItemUI (CategoryItemUI)

Pack List Container: PackListContainer (Rect Transform)

Level Item UI Prefab: LevelItemUI (LevelItemUI)

Level List Container: LevelListContainer (Rect Transform)

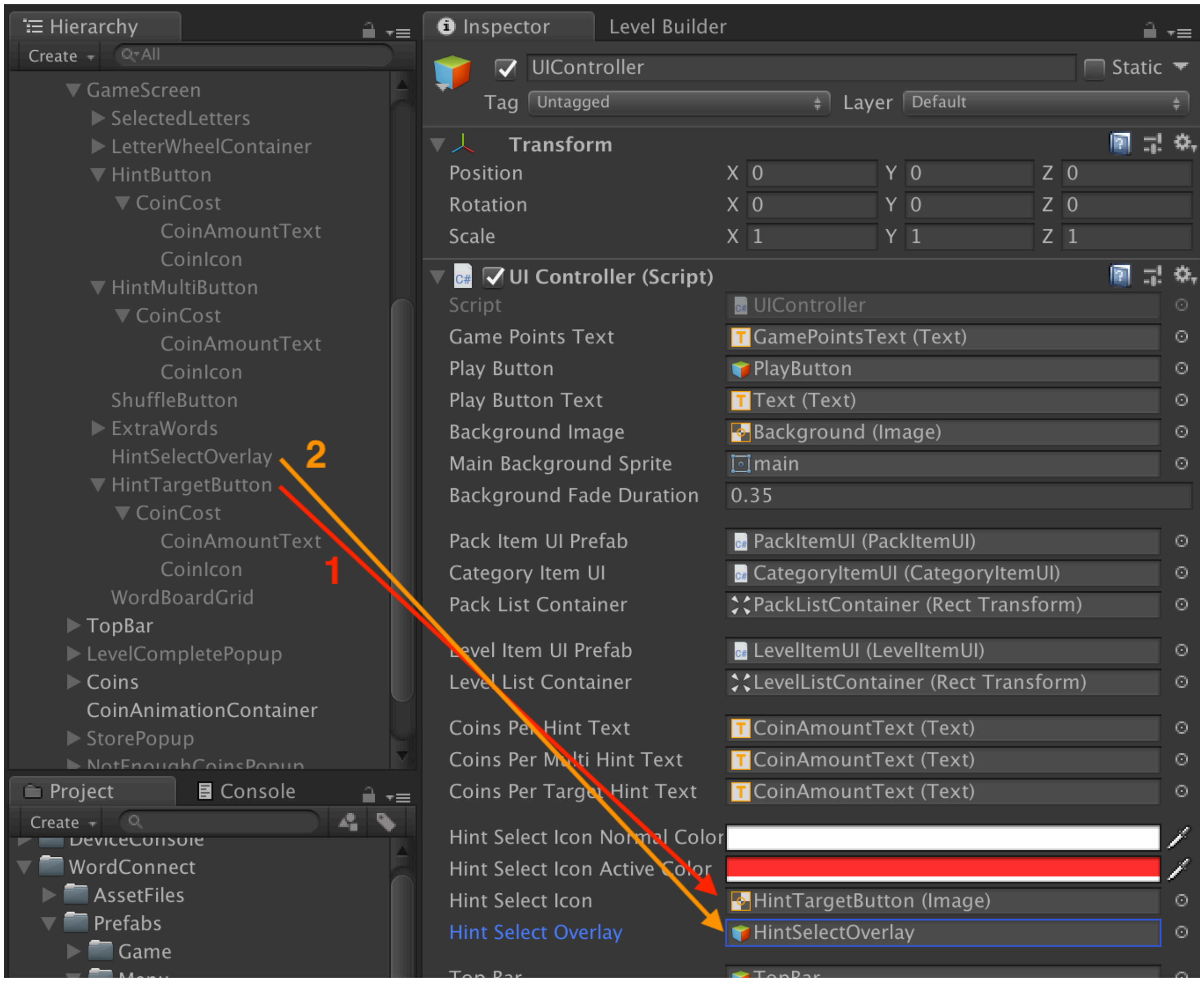
Coins Per Hint Text: CoinAmountText (Text)

Coins Per Multi Hint Text: CoinAmountText (Text)

Coins Per Target Hint Text: CoinAmountText (Text)

Hint Select Icon Normal Color

**1 Coins Per Hint Text** - The Text component that will display the amount of coins a hint costs.
**2 Coins Per Multi Hint Text** - The Text component that will display the amount of coins a multi-hint costs.
**3 Coins Per Target Hint Text** - The Text component that will display the amount of coins a target hint costs.



UIController Static

Tag Untagged Layer Default

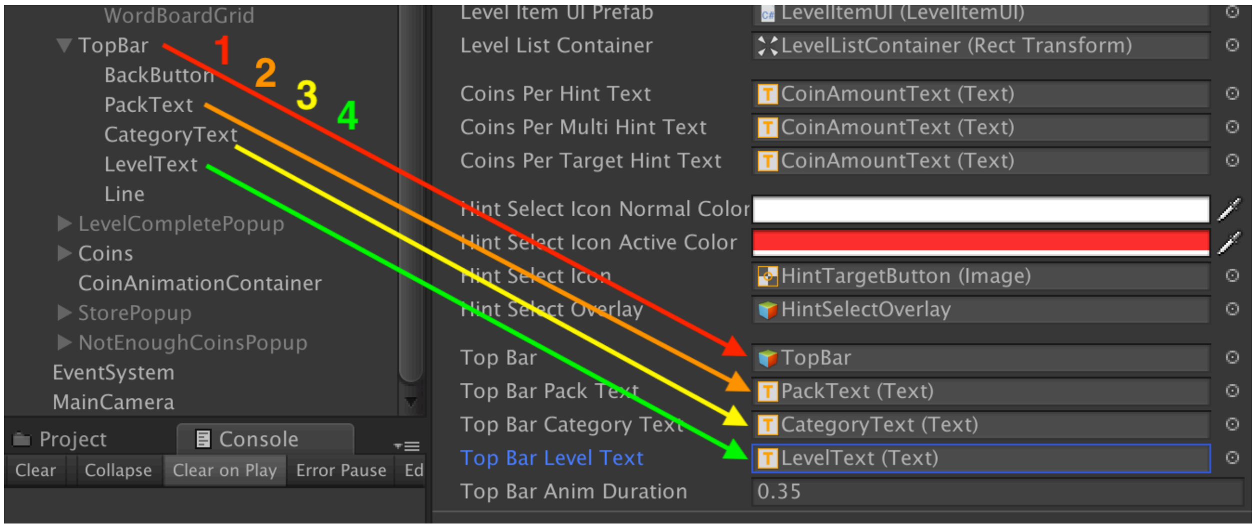
Transform

| | | | | | | |
|---|---|---|---|---|---|---|
| Position | X | 0 | Y | 0 | Z | 0 |
| Rotation | X | 0 | Y | 0 | Z | 0 |
| Scale | X | 1 | Y | 1 | Z | 1 |

UI Controller (Script)

| | |
|---|---|
| Script | UIController |
| Game Points Text | GamePointsText (Text) |
| Play Button | PlayButton |
| Play Button Text | Text (Text) |
| Background Image | Background (Image) |
| Main Background Sprite | main |
| Background Fade Duration | 0.35 |
| Pack Item UI Prefab | PackItemUI (PackItemUI) |
| Category Item UI | CategoryItemUI (CategoryItemUI) |
| Pack List Container | PackListContainer (Rect Transform) |
| Level Item UI Prefab | LevelItemUI (LevelItemUI) |
| Level List Container | LevelListContainer (Rect Transform) |
| Coins Per Hint Text | CoinAmountText (Text) |
| Coins Per Multi Hint Text | CoinAmountText (Text) |
| Coins Per Target Hint Text | CoinAmountText (Text) |
| Hint Select Icon Normal Color | ⬜ |
| Hint Select Icon Active Color | 🟥 |
| Hint Select Icon | HintTargetButton (Image) |
| Hint Select Overlay | HintSelectOverlay |

**1 Hint Select Icon** - The Image component that will have its color set when the target hint is selected.
**2 Hint Select Overlay** - The GameObject that will be set to active when the target hint is selected. This is meant to block all other UI elements other that the cells on the WordBoardGrid so the user can only select a cell on the board.



WordBoardGrid
TopBar
BackButton
PackText
CategoryText
LevelText
Line
LevelCompletePopup
Coins
CoinAnimationContainer
StorePopup
NotEnoughCoinsPopup
EventSystem
MainCamera

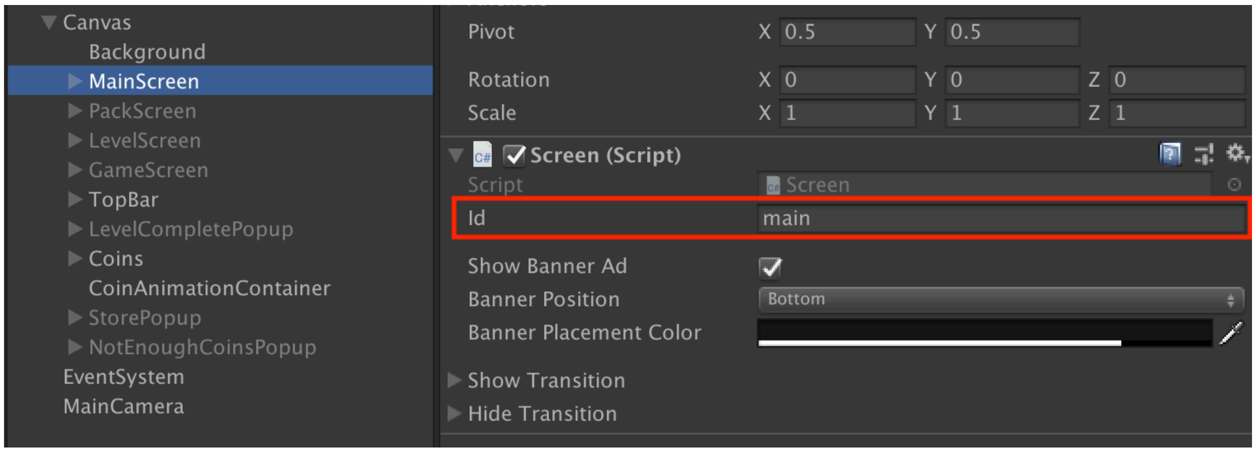
Project | Console
Clear | Collapse | Clear on Play | Error Pause | Ed

| | |
|---|---|
| Level Item UI Prefab | LevelItemUI (LevelItemUI) |
| Level List Container | LevelListContainer (Rect Transform) |
| Coins Per Hint Text | CoinAmountText (Text) |
| Coins Per Multi Hint Text | CoinAmountText (Text) |
| Coins Per Target Hint Text | CoinAmountText (Text) |
| Hint Select Icon Normal Color | ⬜ |
| Hint Select Icon Active Color | 🟥 |
| Hint Select Icon | HintTargetButton (Image) |
| Hint Select Overlay | HintSelectOverlay |
| Top Bar | TopBar |
| Top Bar Pack Text | PackText (Text) |
| Top Bar Category Text | CategoryText (Text) |
| Top Bar Level Text | LevelText (Text) |
| Top Bar Anim Duration | 0.35 |

**1 Top Bar** - The GameObject which has all top bar UI elements as its children. This GameObject is faded out when the Main screen is presented and faded in when the user navigates to any other screen.
**2 Top Bar Pack Text** - The Text component which will display the name of the selected pack on the Level screen.
**3 Top Bar Category Text** - The Text component which will display the name of the category for the active level on the Game screen.
**4 Top Bar Level Text** - The Text component which will display the level number for the active level on the Game screen.

## ScreenController

The screen controller handles transitioning between the screens in the game. A screen can be transition to using **ScreenController.Instance.Show(screenId).** The screenId is the Id that is set on the screen:



Canvas
Background
MainScreen
PackScreen
LevelScreen
GameScreen
TopBar
LevelCompletePopup
Coins
CoinAnimationContainer
StorePopup
NotEnoughCoinsPopup
EventSystem
MainCamera

| | | | |
|---|---|---|---|
| Pivot | X 0.5 | Y 0.5 | |
| Rotation | X 0 | Y 0 | Z 0 |
| Scale | X 1 | Y 1 | Z 1 |

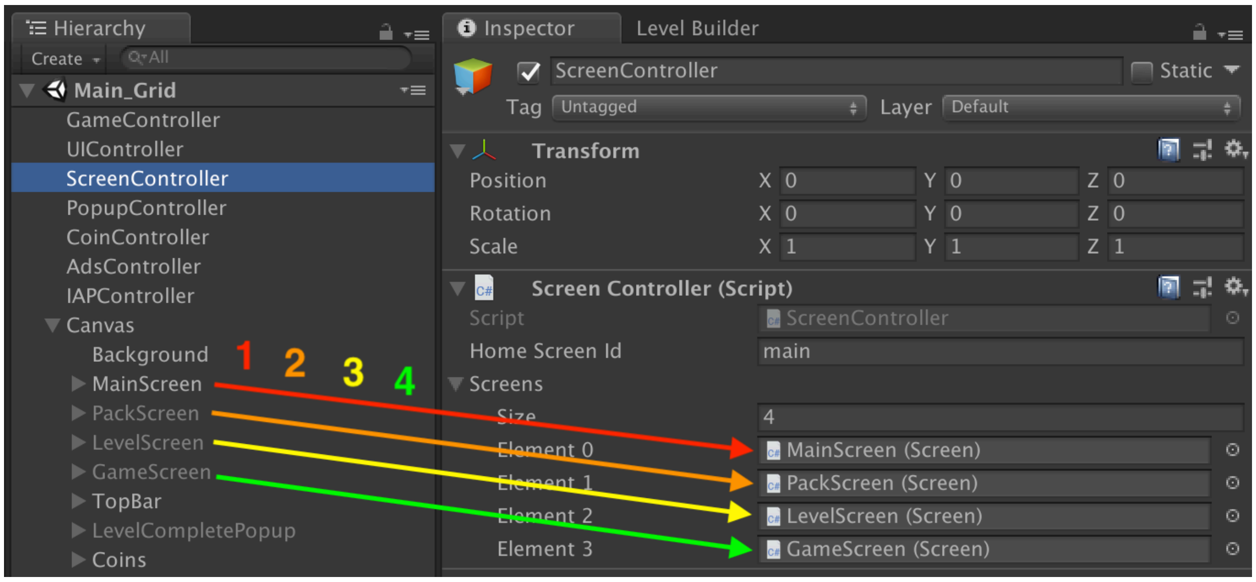
Screen (Script)

| | |
|---|---|
| Script | Screen |
| Id | main |
| Show Banner Ad | ✓ |
| Banner Position | Bottom |
| Banner Placement Color | ⬛ |

Show Transition
Hide Transition

The ScreenController keeps track of the screens that have been shown, calling the method **ScreenController.Instance.Back()** will transition from the current screen to the last shown screen.

There are four screens in the Word Connect Game asset:



ScreenController

Tag Untagged Layer Default

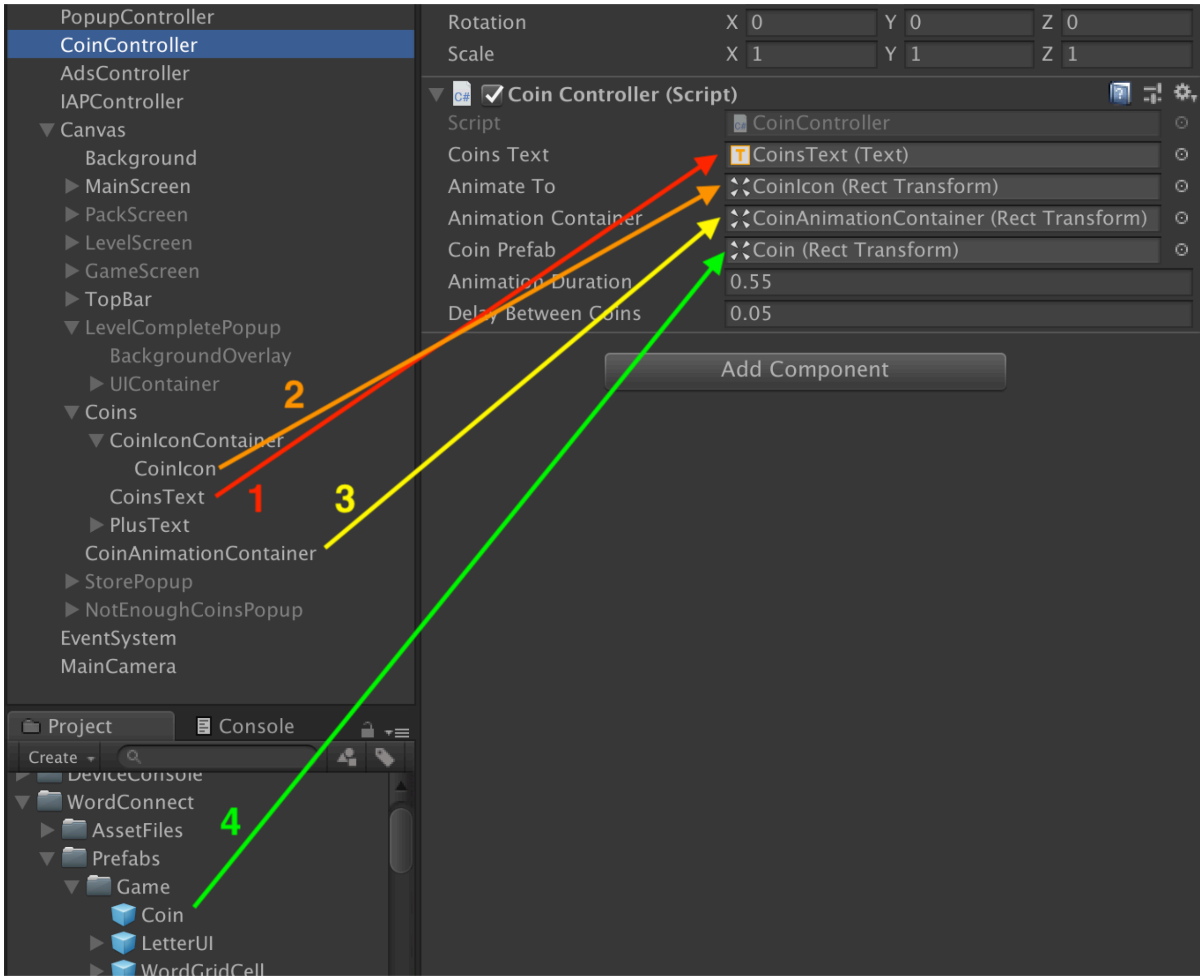
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Script ScreenController
Home Screen Id main
Screens
Size 4
Element 0 MainScreen (Screen)
Element 1 PackScreen (Screen)
Element 2 LevelScreen (Screen)
Element 3 GameScreen (Screen)

**1 MainScreen** - The first screen that the player sees when the app launches.
**2 PackScreen** - The screen that displays the list of packs and categories. Selecting a category transitions to the Level Screen,
**3 LevelScreen** - The screen that displays the levels for the selected category.
**4 GameScreen** - The screen that displays when a level is being played. Selecting a level or clicking play on the Main screen transitions to this screen.

## CoinController

The CoinController handles updating the text that shows the amount of coins the player currently has and animating coin objects when the player receives coins.



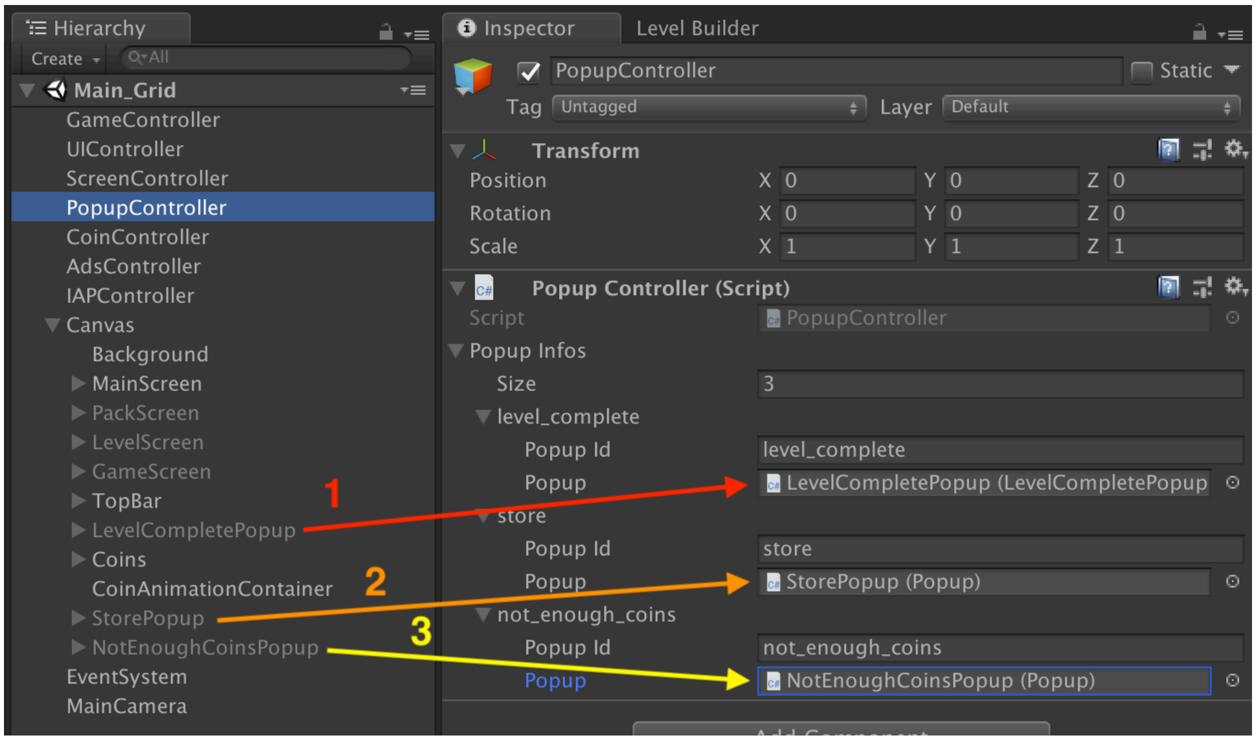
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Coin Controller (Script)
Script: CoinController
Coins Text: CoinsText (Text)
Animate To: CoinIcon (Rect Transform)
Animation Container: CoinAnimationContainer (Rect Transform)
Coin Prefab: Coin (Rect Transform)
Animation Duration: 0.55
Delay Between Coins: 0.05

**1 Coins Text** - The Text component to update when the player receives coins.
**2 Animate To** - The RectTransform that coin objects will animate to when the player receives coins.
**3 Animation Container** - The Transform that instantiated coin objects will be placed in.
**4 Coin Prefab** - The prefab that will be instantiated when coins need to animate.

**Animation Duration** - The amount of time in seconds the coin animation takes.
**Delay Between Coins** - The amount of time in seconds each coin will wait until starting it’s animation.

## PopupController

The PopupController handles displaying popups. A popup can be displayed by calling **PopupController.Instance.Show(id, data, onPopupClosed)**. The id is the Popup Id set on the PopupController, data is an object array containing the data that the popup is expecting, and onPopupClosed is a callback which is invoked when the popup closes.



Main_Grid
- GameController
- UIController
- ScreenController
- PopupController
- CoinController
- AdsController
- IAPController
- Canvas
  - Background
  - MainScreen
  - PackScreen
  - LevelScreen
  - GameScreen
  - TopBar
  - LevelCompletePopup
  - Coins
  - CoinAnimationContainer
  - StorePopup
  - NotEnoughCoinsPopup
- EventSystem
- MainCamera

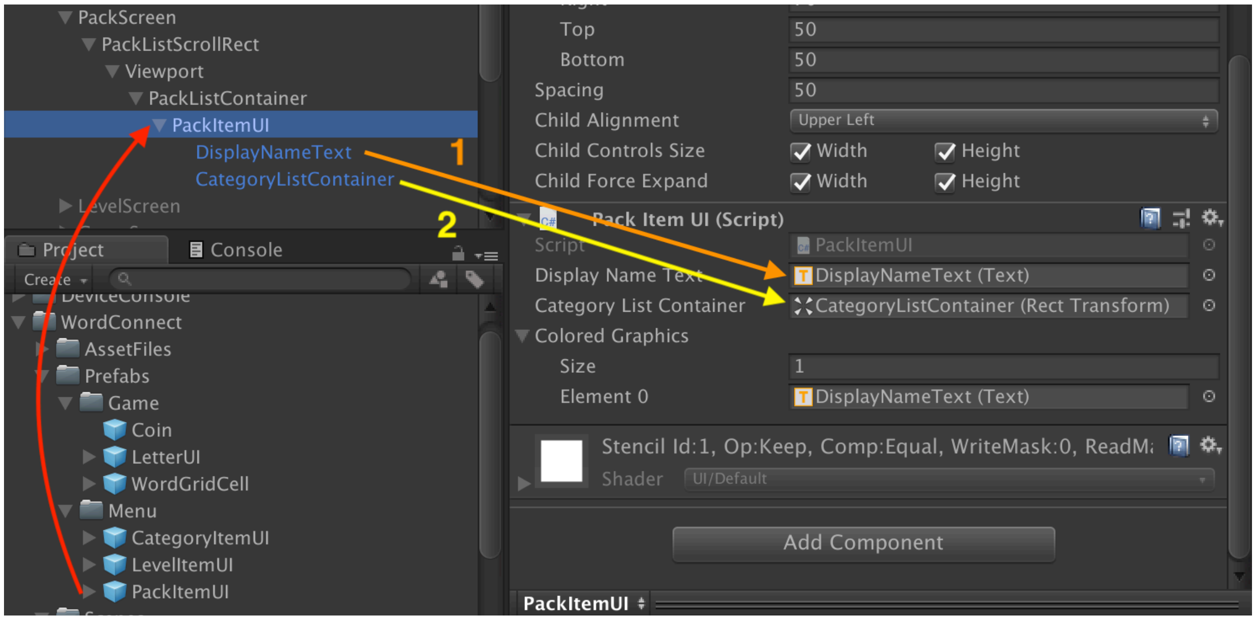
```
PopupController                              Static
Tag Untagged                    Layer Default
Transform
Position      X 0      Y 0      Z 0
Rotation      X 0      Y 0      Z 0
Scale         X 1      Y 1      Z 1
Popup Controller (Script)
Script        PopupController
Popup Infos
  Size        3
  level_complete
    Popup Id  level_complete
    Popup     LevelCompletePopup (LevelCompletePopup
  store
    Popup Id  store
    Popup     StorePopup (Popup)
  not_enough_coins
    Popup Id  not_enough_coins
    Popup     NotEnoughCoinsPopup (Popup)
```

**1 LevelCompletePopup** - Displayed when a level is completed.
**2 StorePopup** - Displayed when the player coins is clicked.
**3 NotEnoughCoinsPopup** - Displayed when the player attempts to use a hint but does not have enough coins.

## PackItemUI

The PackItemUI component handles how a pack should be displayed in the pack list. A PackItemUI prefab exists in the Prefabs/Menu folder and is used by the UIController when instantiating new pack UI list items to be placed in the pack list.

To edit/change the look of a pack UI item drag the PackItemUI prefab into the PackListContainer found in the PackScreen:



| | |
|---|---|
| Top | 50 |
| Bottom | 50 |
| Spacing | 50 |
| Child Alignment | Upper Left |
| Child Controls Size | ✓ Width ✓ Height |
| Child Force Expand | ✓ Width ✓ Height |
| Pack Item UI (Script) | |
| Script | PackItemUI |
| Display Name Text | DisplayNameText (Text) |
| Category List Container | CategoryListContainer (Rect Transform) |
| Colored Graphics | |
| Size | 1 |
| Element 0 | DisplayNameText (Text) |

Stencil Id:1, Op:Keep, Comp:Equal, WriteMask:0, ReadM:
Shader UI/Default

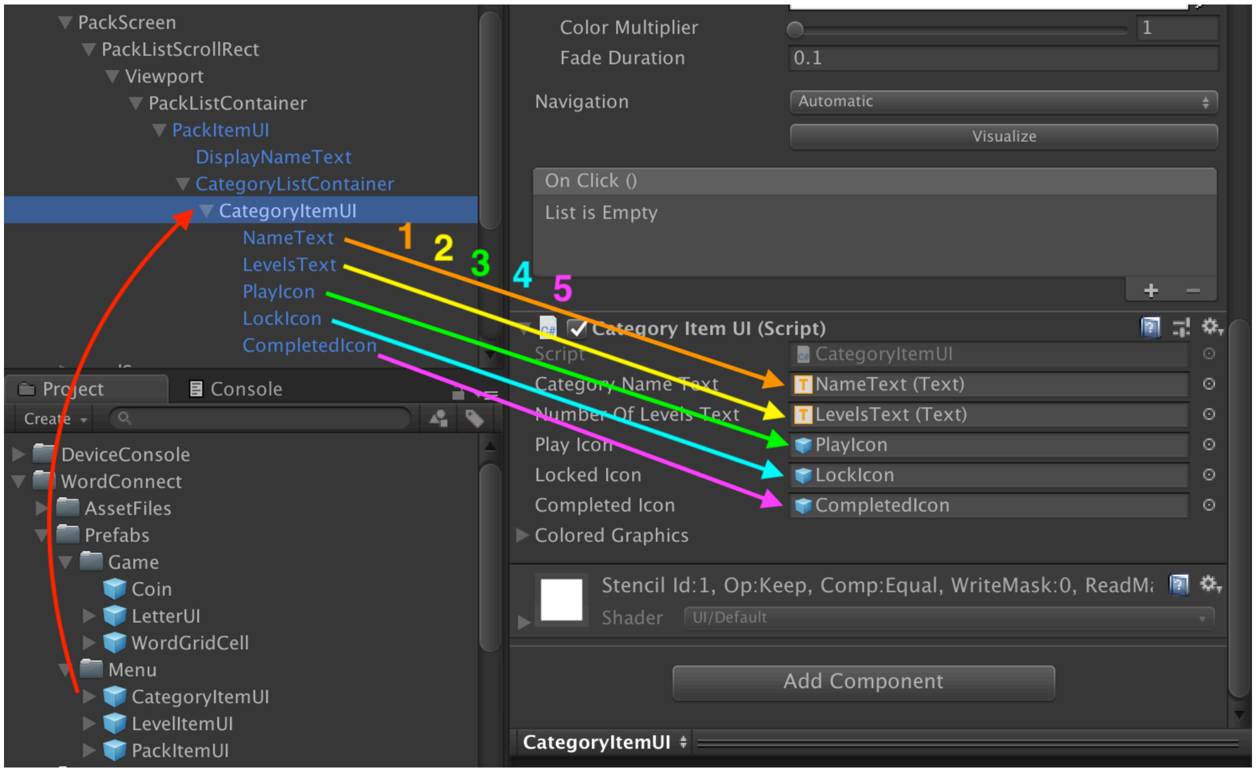
**1 Display Name Text** - The Text component that will display the name of the pack.
**2 Category List Container** - The Transform that CategoryItemUIs will be instantiated and plaid in.

**Colored Graphics** - The list of Graphic components (Text, Image, etc) who's color field will be set the the color of the pack.

## CategoryItemUI

The CategoryItemUI component handles how a category list item will be displayed. A CategoryItemUI prefab exists in the Prefabs/Menu folder and is used by the UIController when instantiating new category list items to be placed in a packs category list.

To edit/change the look of a category item first drag a PackItemUI prefab into the PackListContainer found in the PackScreen. Next drag a CategoryItemUI prefab into the PackItemUIs CategoryListContainer:



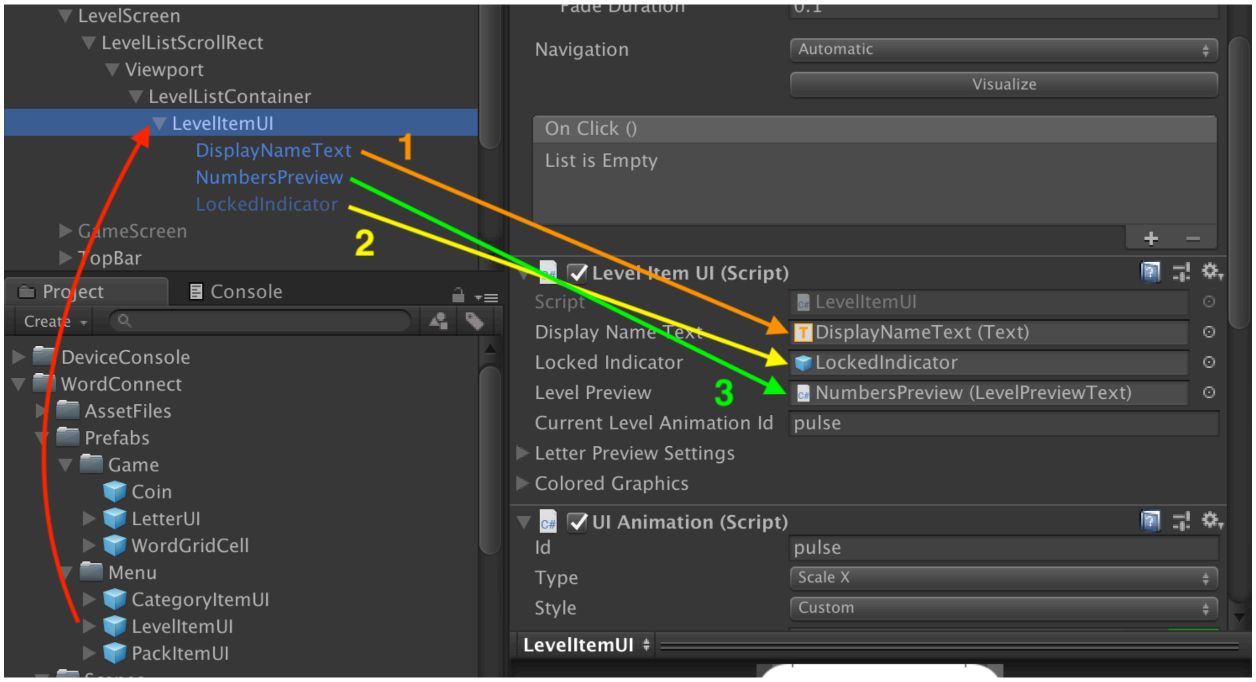
**1 Category Name Text** - The Text component that will display the name of the category.
**2 Number Of Levels Text** - The Text component that will display the to/from level numbers.
**3 Play Icon** - The GameObject that will be set to active if this category is the next playable category.
**4 Locked Icon** The GameObject that will be set to active if this category is locked.
**5 Completed Icon** - The GameObject that will be set to active if all levels in the category have been completed.

**Colored Graphics** - The list of Graphic components (Text, Image, etc) who's color field will be set the the color of the pack that this category belongs to.

## LevelItemUI

The LevelItemUI component handles how a level list item will be displayed. A LevelItemUI prefab exists in the Prefabs/Menu folder and is used by the UIController when instantiating new level list items to be placed on the Level Screen.

To edit/change the look of a level item drag the LevelItemUI prefab into the LevelListContainer found in the LevelScreen:



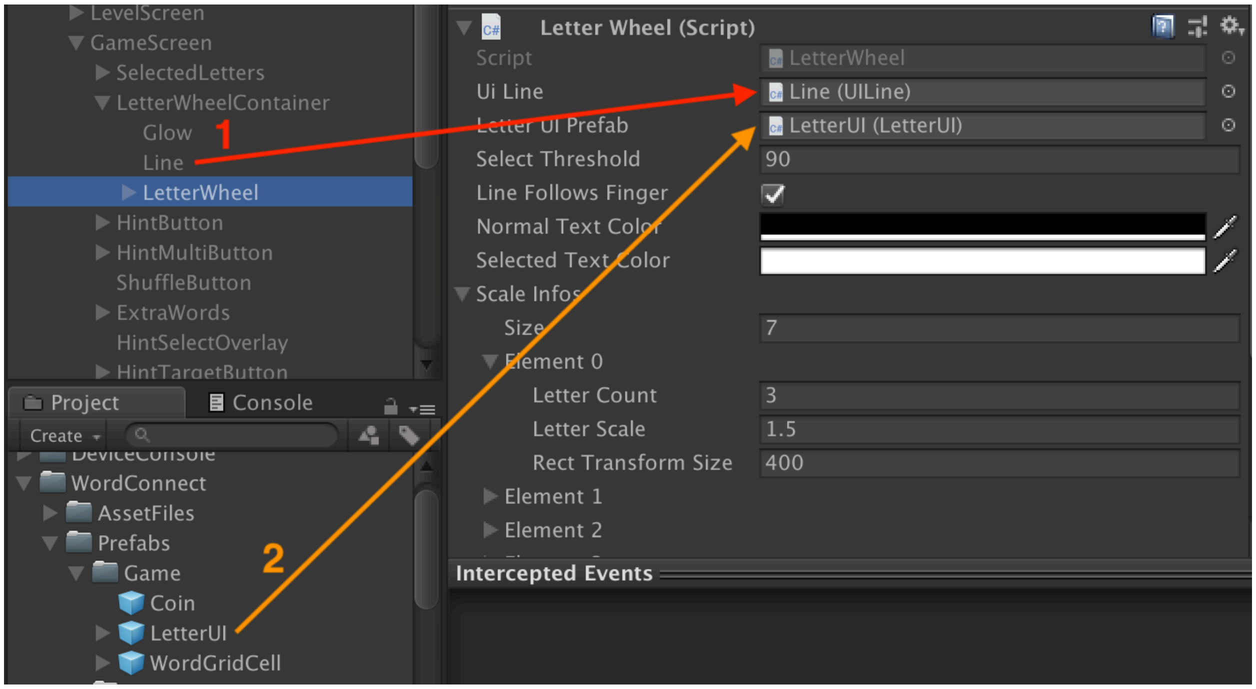
Navigation: Automatic
Script: LevelItemUI
Display Name Text: DisplayNameText (Text)
Locked Indicator: LockedIndicator
Level Preview: NumbersPreview (LevelPreviewText)
Current Level Animation Id: pulse
Id: pulse
Type: Scale X
Style: Custom

**1 Display Name Text** - The Text component that will display levels number.
**2 Locked Indicator** - The GameObject that will be set to active if the level is locked.
**3 Level Preview** - The LevelPreview component that displays the levels letters in a circle.

**Current Level Animation Id** - The UI Animation Id that should play when the LevelItemUI is the active/next level to be played.
**Letter Preview Settings** - List of settings to set on the LevelPreview object. Letter Count is the number of letters in the level for this setting to be used.
**Colored Graphics** - The list of Graphic components (Text, Image, etc) who's color field will be set the the color of the pack that this level belongs to.

## LetterWheel

The LetterWheel component handles displaying the letters in the active level in a circle and invoking an event when the user selected letters.



```
▶ LevelScreen
▼ GameScreen
  ▶ SelectedLetters
  ▼ LetterWheelContainer
      Glow
      Line
    ▶ LetterWheel
  ▶ HintButton
  ▶ HintMultiButton
    ShuffleButton
  ▶ ExtraWords
    HintSelectOverlay
  ▶ HintTargetButton

Project   Console
Create
  DeviceConsole
▼ WordConnect
  ▶ AssetFiles
  ▼ Prefabs
    ▼ Game
        Coin
      ▶ LetterUI
      ▶ WordGridCell
```

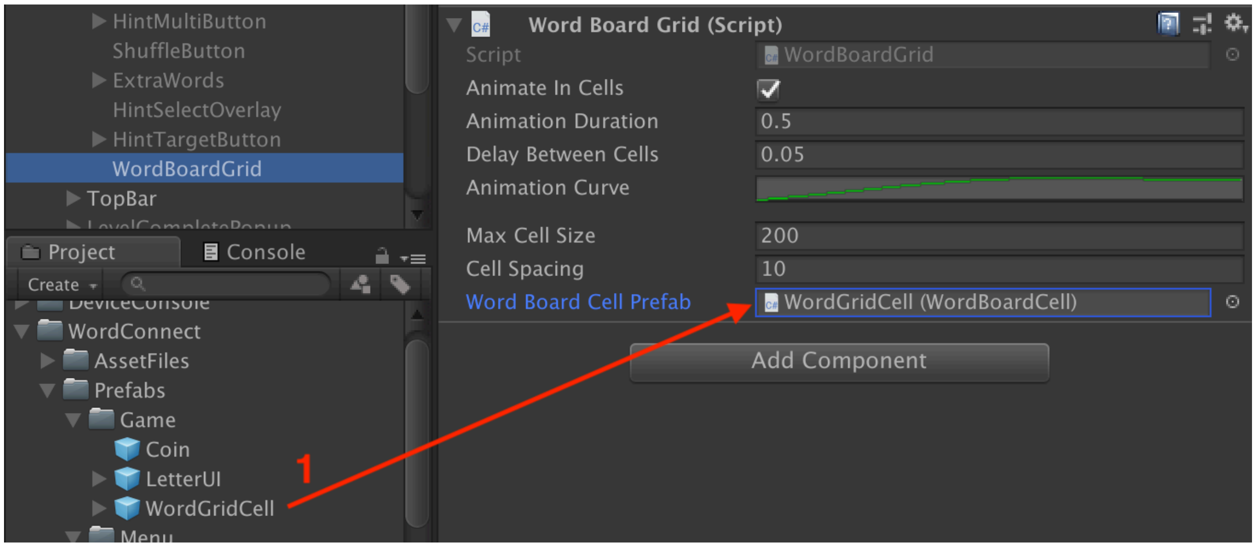
```
▼ Letter Wheel (Script)
Script                  LetterWheel
Ui Line                 Line (UILine)
Letter UI Prefab        LetterUI (LetterUI)
Select Threshold        90
Line Follows Finger     ✓
Normal Text Color       ■
Selected Text Color     □
▼ Scale Infos
    Size                7
  ▼ Element 0
      Letter Count      3
      Letter Scale      1.5
      Rect Transform Size 400
  ▶ Element 1
  ▶ Element 2
Intercepted Events
```

**1 Ui Line** - The UILine component that draws the line between selected letters.
**2 Letter UI Prefab** - The LetterUI prefab that will be instantiated for each letter in a level.

**Select Threshold** - The distance the mouse/finger needs to be from a letter in order to selected it.
**Line Follows Finger** - If selected then the line will be drawn from the last selected letter to the location of the mouse/finger.
**Normal Text Color** - The text color for un-selected letters.
**Selected Text Color** - The text color for selected letters.
**Scale Infos** - List of settings to apply to the LetterWheel and it's letters. Letter Count is the number of letters in the level for this setting to be used. Letter Scale is a scale that is applied to each letter object, Rect Transform Size is the size to set the LetterWheels RectTransform.

## WordBoardGrid

The WordBoardGrid component handles displaying a Grid level. It creates the objects required to display the levels grid and animates in the letters as they are found.



Word Board Grid (Script)

| | |
|---|---|
| Script | WordBoardGrid |
| Animate In Cells | ✓ |
| Animation Duration | 0.5 |
| Delay Between Cells | 0.05 |
| Animation Curve | |
| Max Cell Size | 200 |
| Cell Spacing | 10 |
| Word Board Cell Prefab | WordGridCell (WordBoardCell) |

**1 Word Board Cell Prefab** - The WordBoardCell prefab to use when instantiating cells in the grid.

**Animate In Cells** - If selected then the letters will animate in when a word is found.
**Animation Duration** - The duration of a single cells animation.
**Delay Between Cells** - The amount of time to wait before starting each cells animation when a word has been found.
**Animation Curve** - The curve to use when animating in a cell.
**Mac Cell Size** - The maximum size a single cell can be on the screen when scaling the grid to fit the bounds of the WordBoardGrids RectTransform.
**Cell Spacing** - The amount of spacing between cells in the grid.