Coin Frenzy Game

Game documentation and HowTo guide.

This document contains:

Package Description and features

Coin Frenzy is an action packed game full of challenge and fun. The game is ready to release straight out of the box, and it can also be easily customized to make it even more engaging to your players. The game supports PC/Mac, iOS, Android. It can be played with the mouse, keyboard, gamepad, or touch controls!

How to Play?

Move with the arrow keys or by clicking on the spot you want to go to. Collect seashells and avoid enemies!

Features:

- Game ready for release straight out of the box, just build and play!
- Works on all platforms, PC, Mac, iOS, Android, etc
- Supports multiple resolutions and aspect ratios, automatically.
- Supports Mouse, Keyboard, Gamepad, and Touch controls.
- Easily customizable with lots of options to control game difficulty.
- Great learning resource with commented scripts and documentation.
- All assets included: graphics, sounds, and code.
- UnityAds support with integration guide.

Current version 1.02

Update history

1.04 (04.11.2016)

- Support for Unity 5.5, 5.6, and 2017

1.02 (10.07.2016)

- Support for Unity 5.3 and higher versions.
- Better support for UnityAds 5.2 and above.

1.0 (17.01.2016)

- Added powerups you pick up: Super Fast, and Double Bonus!
- Redesigned UI and made it animated.
- Added a bomb enemy that explodes after a few seconds.

0.9 (09.01.2016)

- Initial version

Credits

The font used is Riffic by Nini Prower

The sounds are courtesy of the free sound project.

Music is Waterford by Kevin MacLeod (Public Domain)

Credits go to these authors for their great sound samples: **mattj99, freefire66, eaglestealthteam, harris85, oddworld, fins**

Please rate my file, I'd appreciate it 🙂

Overview of the game's library contents

Let's take a look inside the game files. Open the main CFGAssets folder using Unity3D 4.6.9 or newer. Take a look at the project library, usually placed on the right or bottom side of the screen. Here are the various folders inside:

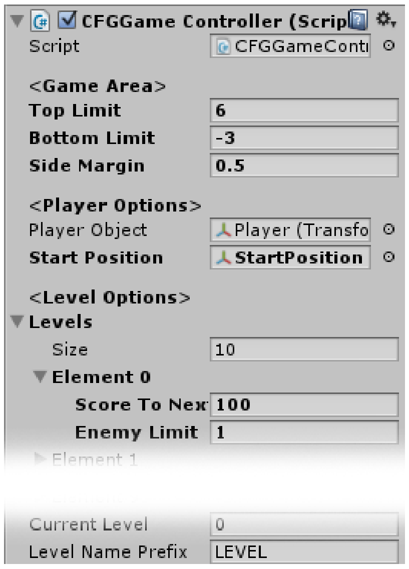
- **Animations:** Holds the animation clips made with Unity’s built-in animation system.
- **FLA:** Holds the object graphics made with Flash CS3. These are vector graphics than can be easily scaled without loss of quality and then exported as PNG to be used in Unity.
- **Fonts:** Holds the font used in the game.
- **Prefabs:** Holds all the prefabs used in the game. These are distributed to various folders for easier access, Buttons, Enemies, Objects, etc. It also holds all the canvases in the game which are used to hold buttons and other UI elements.
- **Scenes:** The first scene that runs in the game is MainMenu. From this scene you can get to the Game scene.
- **Scripts:** Holds all the scripts used in the game. Each prefab contains one or more of these scripts.
- **Sounds:** Holds all the sounds used in the game. Jump,Item, etc
- **Textures:** Holds all the textures used in the game which are used as sprites in Unity.

Getting started

Coin Frenzy Game Template (CFG) is considered a complete project, and as such is supposed to work as the starting point of your planned game, rather than an addition to an existing project. That said, you may of course pick and choose some of the scripts/models to import into your existing project, but CFG works best as a starter kit which you can customize any part of to your liking.

The Game Controller

The Game Controller is the main prefab that controls all the progress of the game from start to finish. It controls the UI of the game, creates enemies and items and checks the level up condition.



CFGGame Controller (Scrip	
Script	CFGGameConti
<Game Area>	
Top Limit	6
Bottom Limit	-3
Side Margin	0.5
<Player Options>	
Player Object	Player (Transfo
Start Position	StartPosition
<Level Options>	
Levels	
Size	10
Element 0	
Score To Nex	100
Enemy Limit	1
Element 1	
Current Level	0
Level Name Prefix	LEVEL

Top/Bottom Limits – You can set the top and bottom limits of the game area. This is where the items and enemies drop and the player moves.

Side Margin – This is the margin from the left/right edges of the screen. The screen width can vary based on the device running the game.

Player Object – This is the player that you control. It must be assigned from the project view.

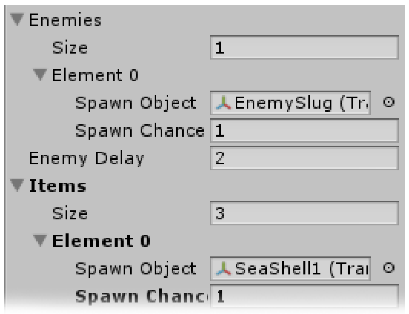
Start Position – The position at which the player object is spawned at the start of the level.

Levels – A list of levels in the game, including the number of points needed to win the game and the limit of enemies.

- **Score To Next Level –** The score needed to win this level.
- **Enemy Limit –** The maximum number of the enemies in this level.

Current Level – The index number of the current level we are on.

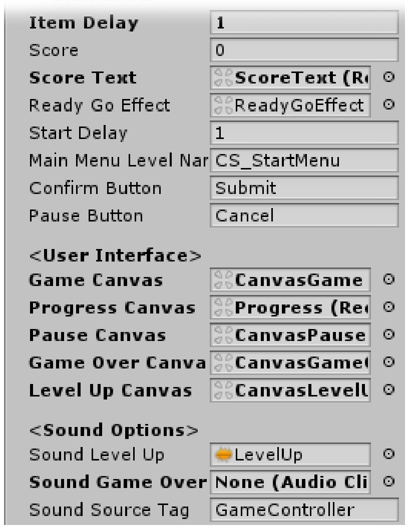
Level Name Prefix – The text that appears before the number of the level.



▼ Enemies	
Size	1
▼ Element 0	
Spawn Object	EnemySlug (Tr
Spawn Chance	1
Enemy Delay	2
▼ **Items**	
Size	3
▼ **Element 0**	
Spawn Object	SeaShell1 (Trai
Spawn Chanc	**1**

Enemies/Items – A list of enemies/items in the game, which are spawned based on their spawn chance.

- **Spawn Object –** The prefab of the enemy/item that can be spawned.
- **Spawn Chance –** The chance of appearance for this enemy/item. This is relevant to other objects in the list.



Item Delay	**1**
Score	0
Score Text	**ScoreText (R**
Ready Go Effect	ReadyGoEffect
Start Delay	1
Main Menu Level Nar	CS_StartMenu
Confirm Button	Submit
Pause Button	Cancel
<User Interface>	
Game Canvas	**CanvasGame**
Progress Canvas	**Progress (Re**
Pause Canvas	**CanvasPause**
Game Over Canva	**CanvasGame**
Level Up Canvas	**CanvasLevelU**
<Sound Options>	
Sound Level Up	LevelUp
Sound Game Over	**None (Audio Cli**
Sound Source Tag	GameController

Enemy/Item Delay – How many seconds to wait before spawning an enemy/item.

Score – The score of the game. Score is earned by shooting enemies and getting streaks.

Score Text – The text object that displays the score, assigned from the scene.

ReadyGoEffect – The effect displayed before starting the game.

Game Speed – The overall game speed. This affects the entire game (Time.timeScale).

Start Delay – How many seconds to wait before the player control starts.

Main Menu Level Name – The level of the main menu that can be loaded after the game ends.

Confirm Button – The keyboard/gamepad button that will restart the game after game over.

Pause Button – The keyboard/gamepad button that pauses the game.

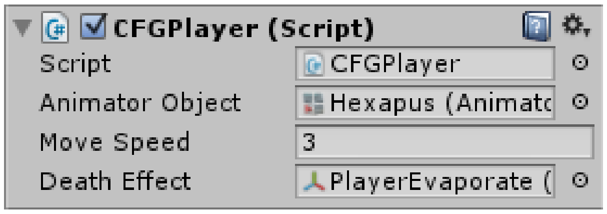
User Interface – Various canvases for the UI, assign them from the scene.

Sounds – Various sounds that play during the game.

Sound Source Tag – The audio source from which the Game Over sound plays.

Editing the Player

The player can move within the game area limits, and is controlled using the GameController component. Here is what you can change in the player component:



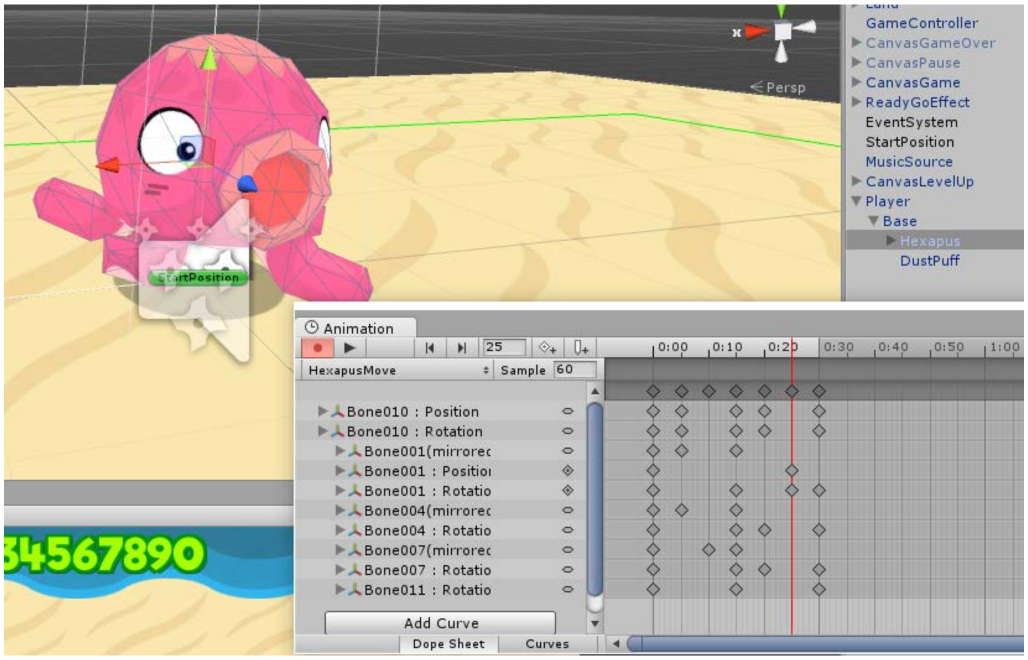
CFGPlayer (Script)

Script	CFGPlayer
Animator Object	Hexapus (Animatc
Move Speed	3
Death Effect	PlayerEvaporate (

Animator Object – This is the animated model that contains an Animator component. The Animator has all the animations of the player (Spawn,Idle,Move).

Move Speed - The movement speed of the player.

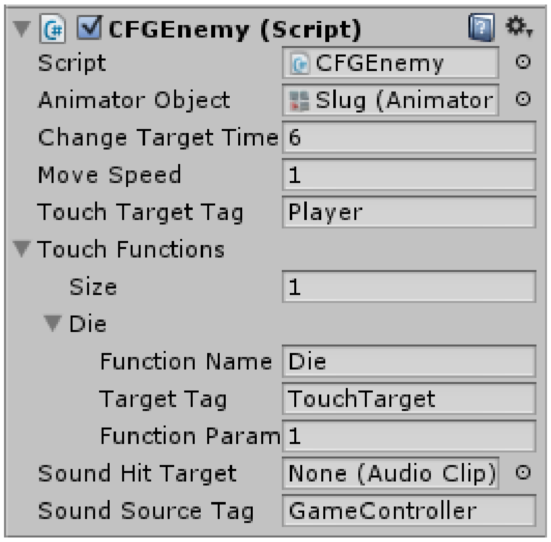
Death Effect - The effect that is created at the location of this object when it is destroyed.



You can edit the animations of the player by selecting the Animator object and pressing **Ctrl+6** to access the animation tab.

Editing the Enemy

The enemy is spawned within the game area limits randomly. It rotates and moves towards a target position within the same area and can kill the player when it hits it. Here is what you can change in the enemy component:



CFGEnemy (Script)	
Script	CFGEnemy
Animator Object	Slug (Animator
Change Target Time	6
Move Speed	1
Touch Target Tag	Player
Touch Functions	
Size	1
Die	
Function Name	Die
Target Tag	TouchTarget
Function Param	1
Sound Hit Target	None (Audio Clip)
Sound Source Tag	GameController

Animator Object – This is the animated model that contains an Animator component. The Animator has all the animations of the player (Spawn,Idle,Move).

Change Target Time – How any seconds to wait before choosing a new target for this enemy.

Move Speed - The movement speed of the enemy.

Touch Target Tag - The tag of the object that this enemy can touch.

Touch Function - A list of functions that run when this enemy touches the target.

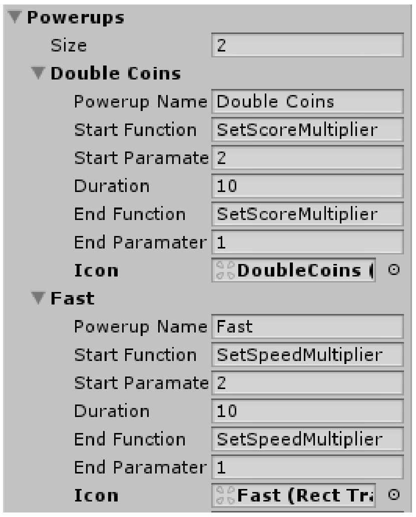
In our case we set **Touch Target Tag** to **Player** so that when the enemy collides with the player it will trigger a **Touch Function**. The function we call is **Die()**, and we call it on the player ,which makes the player die.

Sound Hit Target – The sound produced when the enemy touches the player.

Sound Source Tag – The source from which sounds are played.

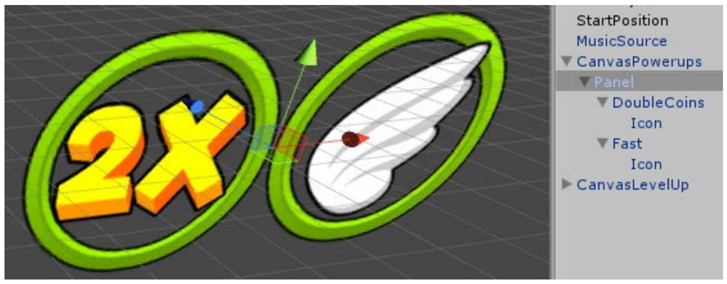
Editing Powerups

Powerups are special items you can pick up in the game. When a powerup is activated it runs a function in the gamecontroller, using SendMessage. The list of powerups you have is defined in the game controller.



```
▼ Powerups
    Size              2
  ▼ Double Coins
      Powerup Name    Double Coins
      Start Function  SetScoreMultiplier
      Start Paramate  2
      Duration        10
      End Function    SetScoreMultiplier
      End Paramater   1
      Icon            DoubleCoins (
  ▼ Fast
      Powerup Name    Fast
      Start Function  SetSpeedMultiplier
      Start Paramate  2
      Duration        10
      End Function    SetSpeedMultiplier
      End Paramater   1
      Icon            Fast (Rect Tr
```

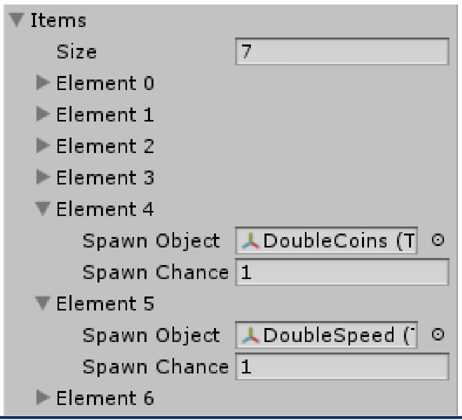
Let's see how the powerups are listed. The first power up doubles the score from collecting coins. It does so by sending a **SetScoreMultiplier** function command with a parameter of 2. Then the game counts 10 seconds before sending a **SetScoreMultiplier** function again with a parameter of 1, making the score multiplier return to normal. Finally we assigned an icon to represent the power up duration while it's activated. You can see what the icon looks like below.



This is the powerup icon, made of a fillamount circle shows the duration of the powerup and an icon.

Another powerup we have speeds up the player to twice its normal speed for 10 seconds. This is done by sending a **SetSpeedMultiplier** function command with a parameter of 2. To end the power up duration we wait 10 seconds and then call the same function but with a parameter of 1, making the player speed go back to normal.

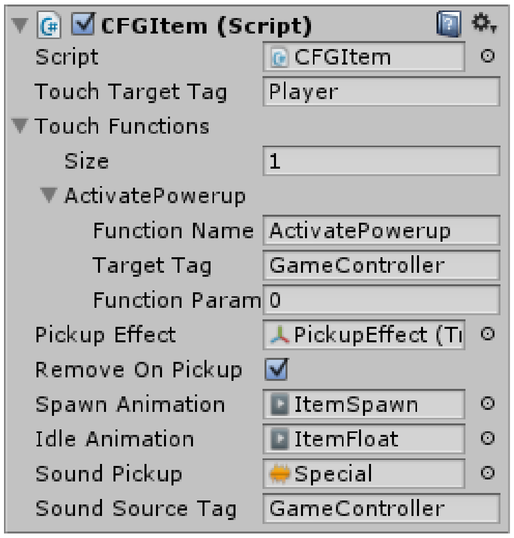
In order for the powerups to work we need to put items in the game which activate those powers when picked up. We also do this in the game controller.



```
▼ Items
    Size              7
  ▶ Element 0
  ▶ Element 1
  ▶ Element 2
  ▶ Element 3
  ▼ Element 4
      Spawn Object    DoubleCoins (T
      Spawn Chance    1
  ▼ Element 5
      Spawn Object    DoubleSpeed (
      Spawn Chance    1
  ▶ Element 6
```

The **Items** list holds all the possible items that can show up in a game. The item is chosen used a weighted randomization determined by the value of **SpawnChance**. The higher the value the more chance this item has to appear.

Let's take a look at the items we pick up in the game and how they activate a powerup. Go to the **Prefabs > Items** folder and drag **DoubleCoins** and **DoubleSpeed** items to the scene. The component in those prefabs is called **CFGItem**.



CFGItem (Script)

Script	CFGItem
Touch Target Tag	Player
Touch Functions	
Size	1
ActivatePowerup	
Function Name	ActivatePowerup
Target Tag	GameController
Function Param	0
Pickup Effect	PickupEffect (Ti
Remove On Pickup	✓
Spawn Animation	ItemSpawn
Idle Animation	ItemFloat
Sound Pickup	Special
Sound Source Tag	GameController

Click on the DoubleCoins object and you'll see the component, in which we called an **ActivatePowerup** function command from the gamecontroller, with a parameter of 0. This activates the ***first*** powerup in the powerups list in the gamecontroller, which is the **DoubleCoins** powerup. If you take a look at the **DoubleSpeed** object you'll see that the parameter we sent is 1, which activates the ***second**** powerup in the list.

UnityAds Integration (Unity 5.2 +)

Since Unity 5.2 UnityAds integration has been simplified, here’s how you can have full screen video ads in your game.

This video shows a quick process of integrating UnityAds into your project. In the example we used one of my templates, but it works on all my other templates too.

https://www.youtube.com/watch?v=EQNTgfV35DU

Here is what we did in the process:

1. Sign in to your Unity account in order to allow Unity Services such as UnityAds to be activated.
2. Open Build Settings and switch the platform to one of the supported ones (iOS, Android).
3. Download Puppeteer's UnityAds package from: puppeteerinteractive.com/freebies/PUPUnityAds.unitypackage
4. Drag the downloaded package into your Unity project, and import it. This UnityAds prefab can be used to display ads every several minutes.
5. Drag the prefab into any scene where you want ads to be shown. Make sure to save changes.
6. The time check is shared between all prefabs in all scenes, so you will never show too many ads.
7. The final step is to activate UnityAds services and get your unique project ID.
8. Open the services window and choose your organization, then click create.
9. Choose UnityAds from the list and turn it On.
10. Choose age group for your project (Will affect the nature of ads shown), and save changes.
11. While working on your project keep Test Mode activated. But when you are ready to release the final project, switch Test Mode off.
12. That's it! Now when you start the game, an ad will be shown after 3 minutes. The ad will never appear during gameplay or post-game

screen. Instead, it will wait until the next level load (restart, main menu, etc) and then show the ad.

Before releasing a game, make sure you uncheck **Enable Test Mode.**

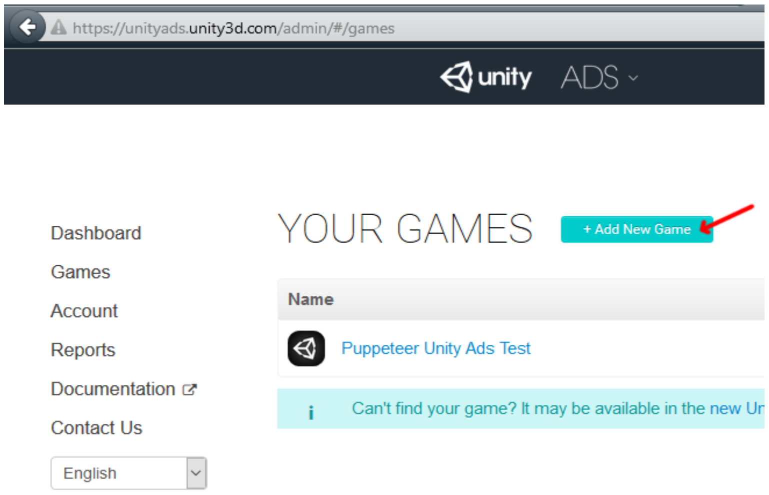
For more info about integrating UnityAds read this:

http://unityads.unity3d.com/help/monetization/integration-guide-unity

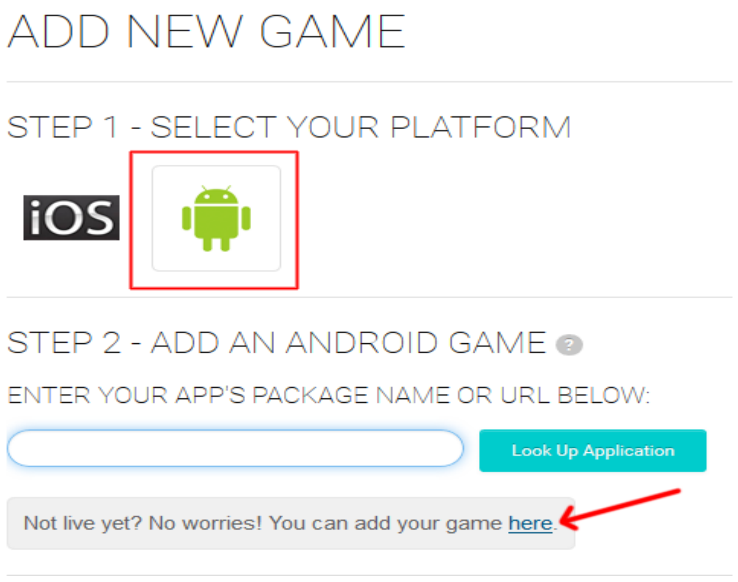
Integrating UnityAds into your project (Unity 4)

Adding support for UnityAds into your current project is simple and shouldn't take you more than 5 minutes. Let's start:

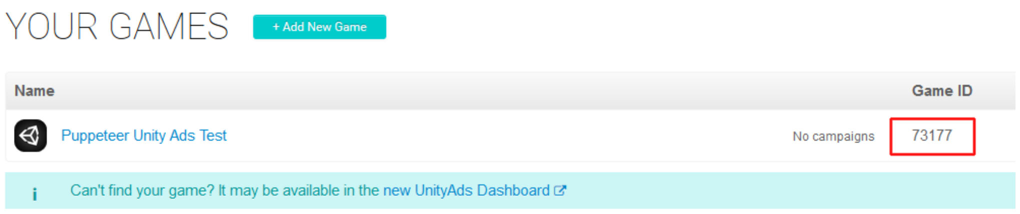
First we need to create our game entry on the UnityAds website. Go to https://unity3d.com/services/ads and create a new game. If you already have your app set and your GameID noted, just skip this part and go straight to importing the UnityAds package into the game.



Now we need to choose the platform. The process is similar for both iOS and Android but for the purpose of this tutorial we'll choose Android. If you have an app on Android, enter its name to find it. If you don't have an app, click below where the red arrow points in order to enter the name of the app that has not been added to the store yet. This way you can test the app before it goes live.


ADD NEW GAME
STEP 1 - SELECT YOUR PLATFORM
iOS
STEP 2 - ADD AN ANDROID GAME
ENTER YOUR APP'S PACKAGE NAME OR URL BELOW:
Look Up Application
Not live yet? No worries! You can add your game here.

After you created your app in the website, make note of the Game ID that appears. This will be used to link the ads to your app.



YOUR GAMES

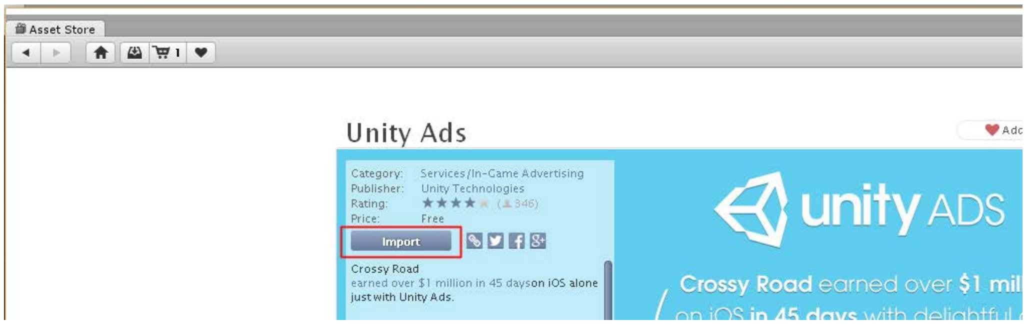
Name		Game ID
Puppeteer Unity Ads Test	No campaigns	73177

Can't find your game? It may be available in the new UnityAds Dashboard

In Unity Editor

Now we need to import the UnityAds package. Open the Unity Asset Store and download the UnityAds package. Import it into your project.

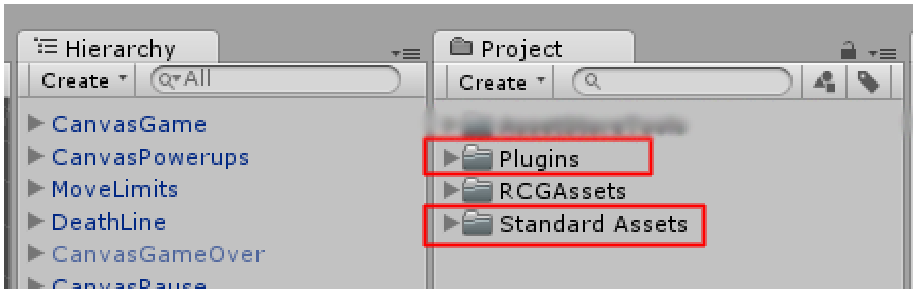
(https://www.assetstore.unity3d.com/en/#!/content/21027)


Asset Store
Unity Ads
Import
unity ADS
Crossy Road earned over $1 mil
on iOS in 45 days with delightful

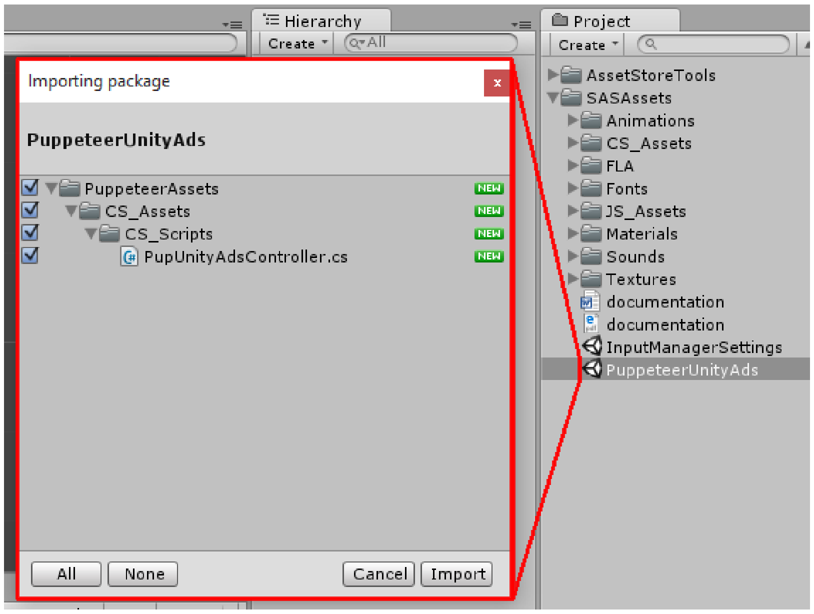
Category: Services/In-Game Advertising
Publisher: Unity Technologies
Rating: ★★★★ (346)
Price: Free

Crossy Road earned over $1 million in 45 dayson iOS alone just with Unity Ads.

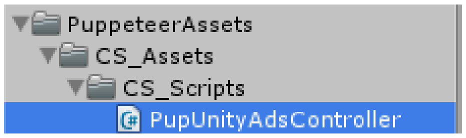
After import you should have two additional folders in your project.

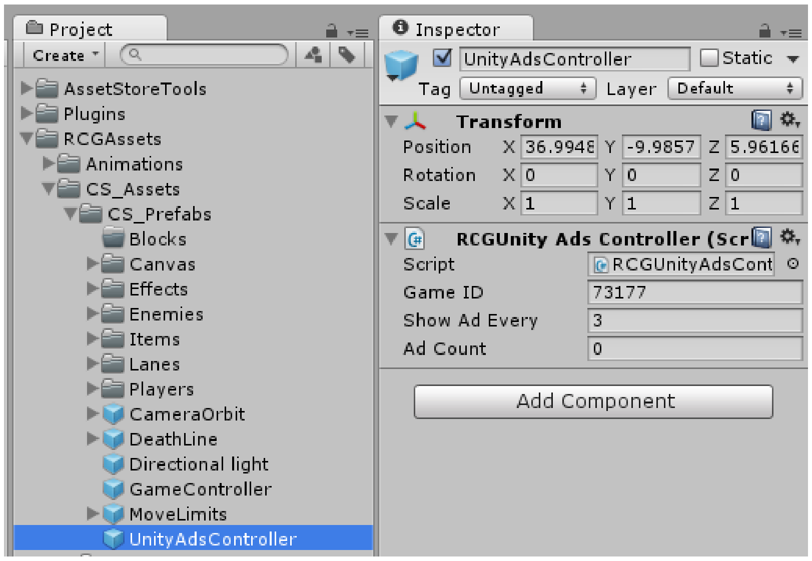


Now we need to bring in the code that integrates the ads into our game. Click on the **PuppeteerUnityAds** package in your project to import it into the game, or choose **Assets > Import Package > Custom Package...** from the top menu and navigate to the **PuppeteerUnityAds** package in your project to import it.



PupUnityAdsController.cs is the main script that links your app to the unityads system. Drag it into your game controller. Now when you look at it you see you can set the GameID of your app, and how often the ads appear. The ad is checked when the level is loaded. "**Show Ad Every**" decides how many times the level needs to be loaded before an ad appears.





UnityAdsController Static
Tag Untagged Layer Default

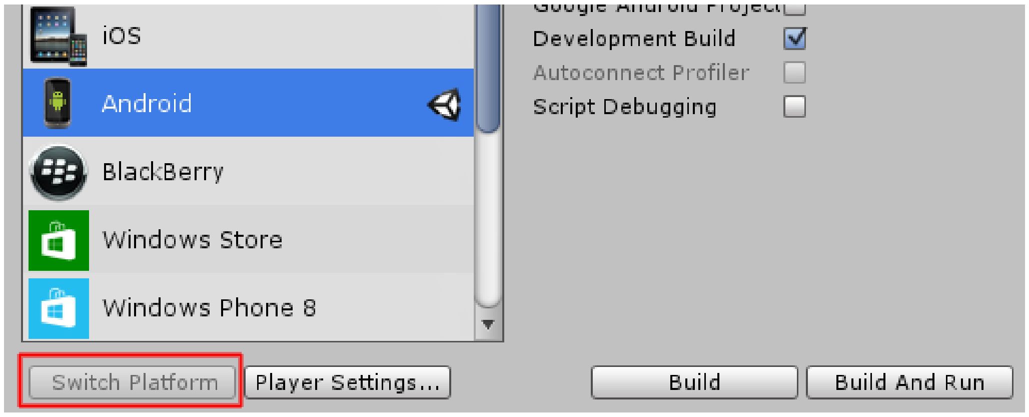
Transform

	X	Y	Z
Position	36.9948	-9.9857	5.96166
Rotation	0	0	0
Scale	1	1	1

RCGUnity Ads Controller (Scr

Script	RCGUnityAdsCont
Game ID	73177
Show Ad Every	3
Ad Count	0

In order to test the ads, we need to switch to the Android platform.



That's it! Now start a level and restart it 3 times, then you should see a blue screen showing the ad system has been activated correctly. If you build to Android you should see an actual video ad appear after 3 level loads.

Does this package work on mobile?

Yes, this package has been successfully tested on both Android and iOS devices. The scripts for each lock type include controls for mobile that are detected automatically based on the platform it's built on.

My sprites are not showing on iOS

Sprite-based textures made with the new Unity 4.3 can sometimes disappear when working on the iOS platform.

You can notice this by opening a scene playing it. When you switch from your current platform to the iOS platform the sprite textures become invisible.

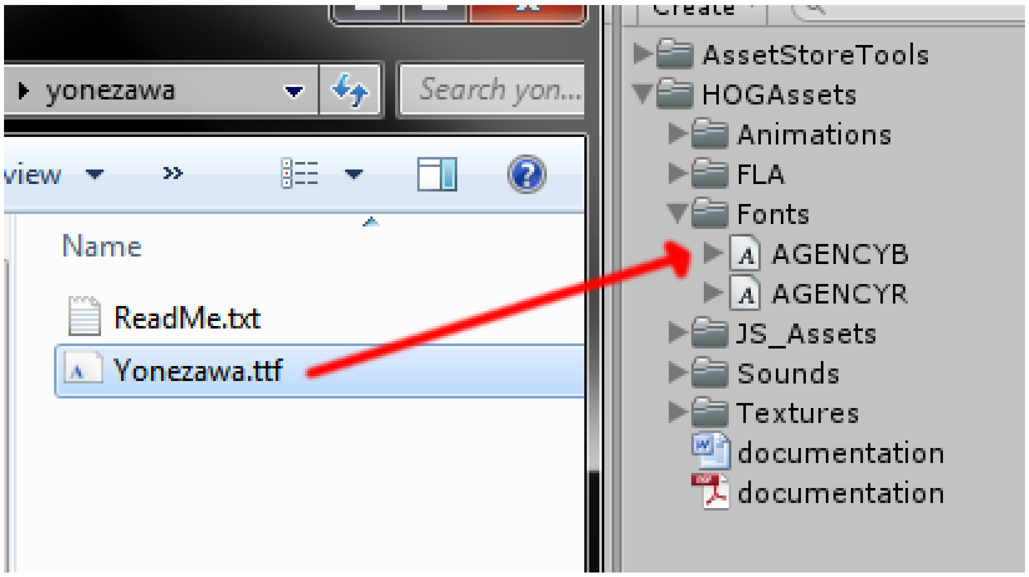
To solve this we must change the texture compression format for iOS. Follow these steps:

1. Click on a texture in the project view.
2. Click on the override for CFGone button on the right side.
3. Change the format to 16bit.
4. Click Apply.

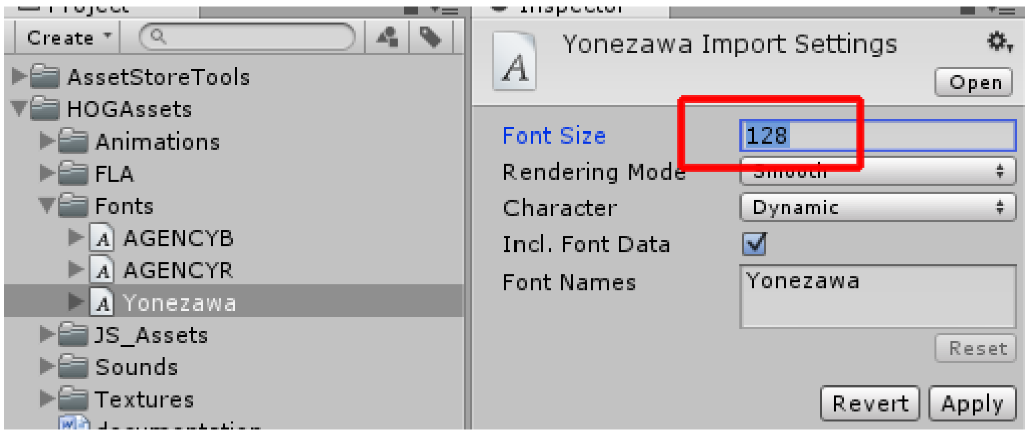
How to change font in the game?

To change a font in the game do the following:

Find a font you like and drag the .ttf file over to the Fonts folder in your game.



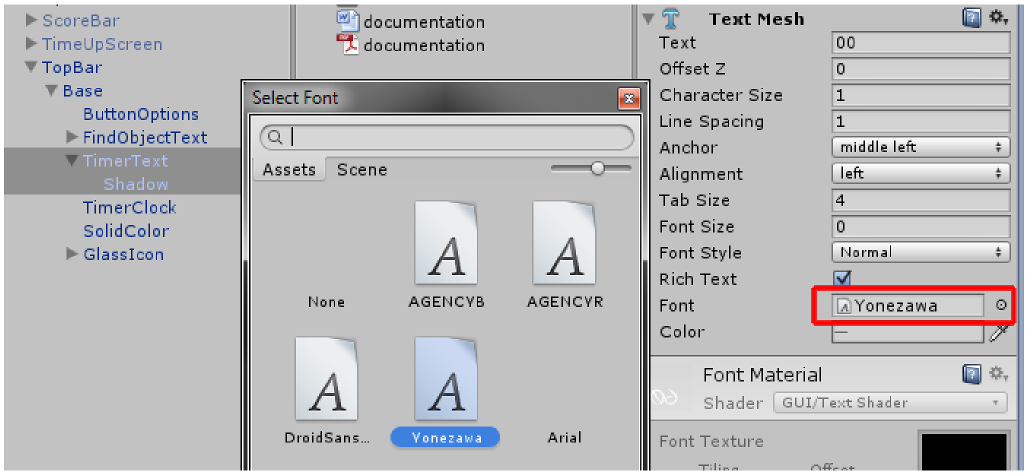
Click on the font you added and edit its attributes. I personally set all my fonts to a high number (and then scale the text object down) so that they look crisper in-game.



Yonezawa Import Settings

Font Size	128
Rendering Mode	Smooth
Character	Dynamic
Incl. Font Data	✓
Font Names	Yonezawa

Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and edit them together.



Text Mesh

Text	00
Offset Z	0
Character Size	1
Line Spacing	1
Anchor	middle left
Alignment	left
Tab Size	4
Font Size	0
Font Style	Normal
Rich Text	✓
Font	Yonezawa
Color	

Font Material
Shader GUI/Text Shader
Font Texture
Tiling Offset

More games by Puppeteer

It is highly advised, whether you are a designer or a developer to look further into the code and customize it to your pleasing. See what can be improved upon or changed to make this file work better and faster. Don't hesitate to send me suggestions and feedback to puppeteerint@gmail.com

Follow me on twitter for updates and freebies!

Good luck with your modifications!