Dart Wheel Game Template

Game documentation and HowTo guide.

This document contains:

Package Description and features

Dart Wheel Game is a full Unity template ready for release. It is compatible with mobile as well as standalone.

How to Play?

Throw darts at the targets. Hit closer to the center for more points, and don't hurt the poor fellow. Have Fun!

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.
- UnityAds support with integration guide.

Current version 1.10

Update history

1.10 (27.10.2018)
- Cleaned up project assets so they can be easily mixed into other projects.

1.08 (02.11.2017)
- Support for Unity 5.5, 5.6, and 2017

1.06 (13.06.2016)
- You can now set multiple hit areas, each with its own sound effect and facial expression.
- Support for versions of Unity higher than 5.3.
- Better UnityAds support for Unity 5.2 and above.

1.03 (19.01.2016)
- Support for UnityAds along with an integration guide.
- Uploaded packages for Unity 4.6.9, 5.1, 5.2, and 5.3
- Support for SceneManager.

1.0 (30.07.2015)
- Initial version

Credits

The font used is Big Bottom by Karen B. Jones

The sounds are courtesy of the free sound project.

Music is a clip from Barroom Ballet by Kevin Macleod (Free Public Domain)

Credits go to these authors for their great sound samples: **cubic-archon, memexikon, 110110010, fins, mattgarkusha, danhelbling**

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 DWGAssets folder using Unity3D 4.6.7 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:

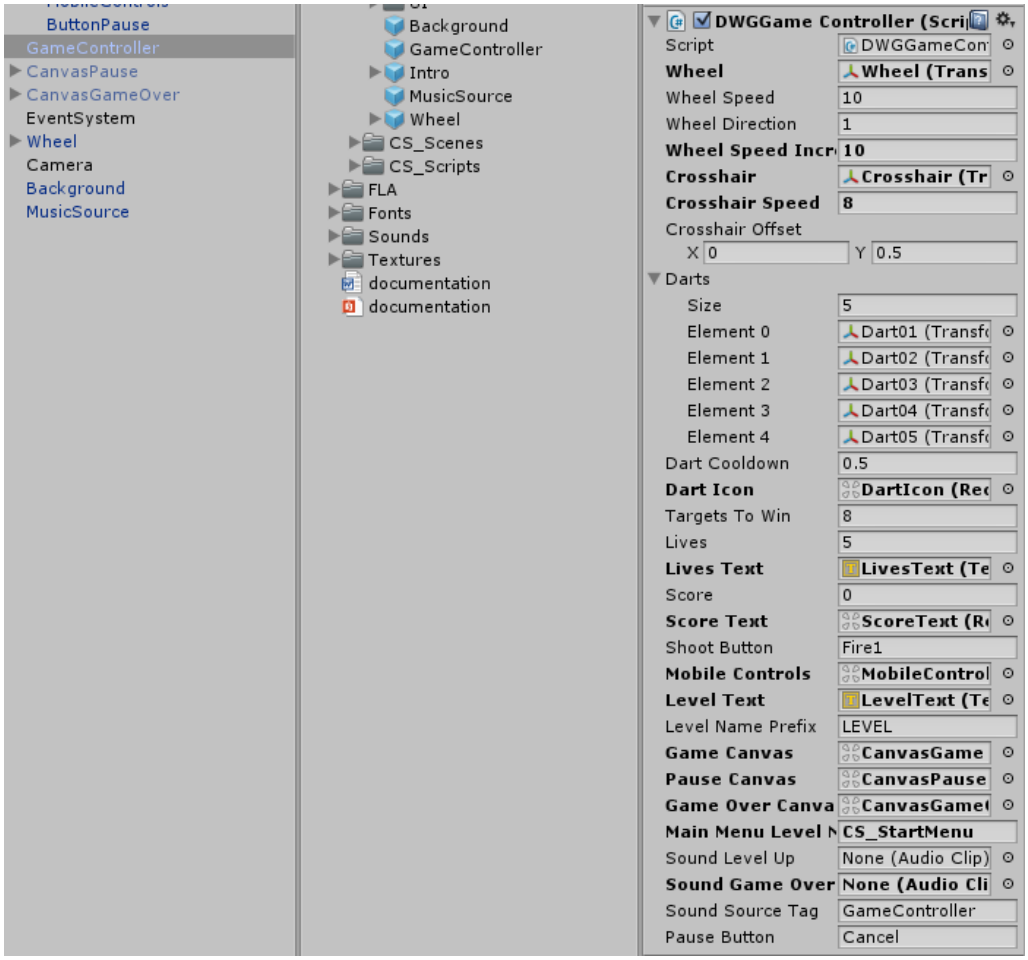
- **Animations:** Holds the animation clips made with Unity's built-in animation system.
- **FLA:** Holds the object graphics made with Flash CS3. These are vector graphics than can be easily scaled without loss of quality and then exported as PNG to be used in Unity.
- **Fonts:** Holds the font used 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

Infinite Platform Hopper (DWG) 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 DWG 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, counts target hits and score. The Game Controller holds all the other parts of the game together, allowing them to interact with each other during play. If any part of the game controller is not assigned correctly, it may cause the game to behave unexpectedly or not even run at all. The gamecontroller must be tagged as "GameController".



Wheel – The wheel object which spins during the game. You assign this from

the scene.

Wheel Speed – How fast the wheel spins.

Wheel Direction – 1 spins the wheel left, and -1 spins the wheel right.

Wheel Speed Increase – How much faster the wheel spins each level.

Crosshair – The crosshair object which is used when we play with the keyboard, gamepad, or when on mobile. You assign this from the scene.

Crosshair Speed – How fast the crosshair moves.

Crosshair Offset – How far from the center of the tap position the crosshair is placed. This is used when on mobile to allow the player to aim more easily.

Darts – a list of darts that can be thrown at the targets randomly.

Dart Cooldown – How many seconds before we can throw another dart.

Dart Icon – This icon shows the dart cooldown. Assigned from the scene.

Targets To Win – How many targets we have to hit in order to win the level.

Lives & Lives Text – The current lives of the player, and the text object that displays the lives. Assigned from the scene.

Score & Score Text – The current score of the player, and the text object that displays the score. Assigned from the scene.

Shoot Button – The button you press in order to throw a dart. This is only relevant to Mouse, Keyboard, and Gamepad controls. If you run the game on mobile, the mobile controls object will take over.

Mobile Controls – This is the mobile controls object which listens to tap on the screen to aim and throw the dart. This is assigned from the scene.

Level Text – The text object that displays the current level we are on. This is assigned from the scene.

Level Name Prefix – The text that is displayed before the number of the current level. Ex: “LEVEL 1“, “LEVEL 2”, etc

Canvases – These are canvas UI screens. **Game Canvas** appears during gameplay, **Pause Canvas** appears when the game is paused and at the start of the game, **Game Over Canvas** appears at the end of the game when the player dies.

Main Menu Level Name – The name of the level that will be loaded if we choose to quit after Game Over.

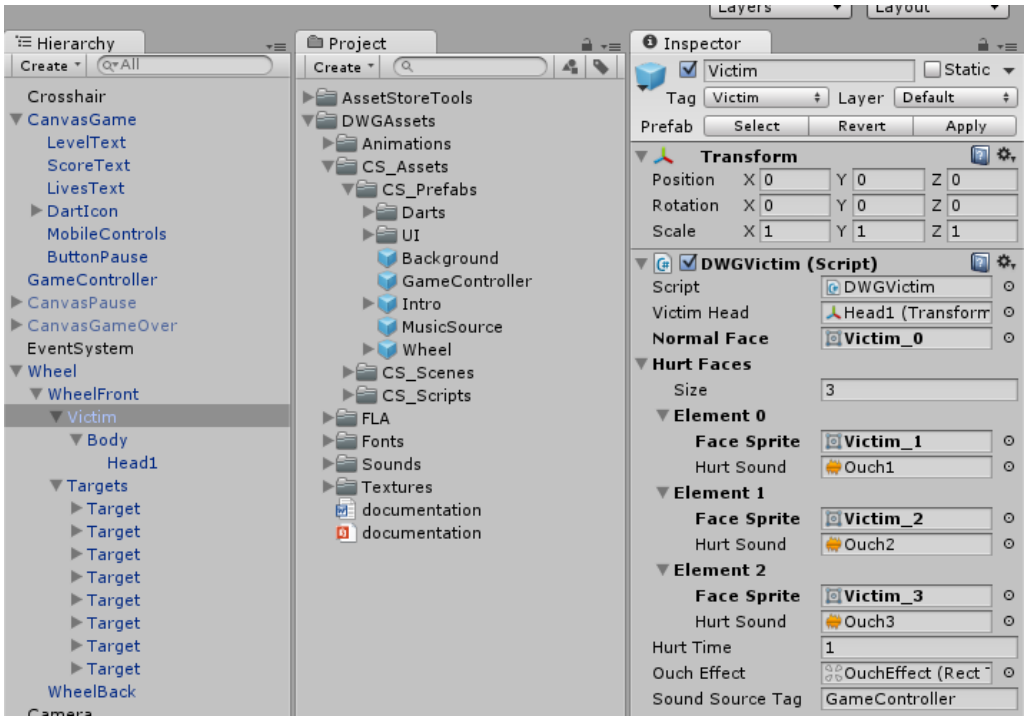
Sound Game Over – The sound that plays on Game Over.

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

Pause Button – This button pauses the game. It work on keyboard and gamepad.

The Victim

The Victim is the poor guy attached to the wheel, which you should try not to hit with a dart. If you hit him with a dart you lose a life. A victim can have several facial expressions along with sound that react to being hit. The victim must be tagged as "Victim".



Victim Head – The head object of the victim. This is used to display the different facial expressions when being hit. This is assigned from the scene.

Normal Face – The default facial expression the victim head has.

Hurt Faces – A list of facial expressions along with sound effects that are chosen randomly when the victim is hit.

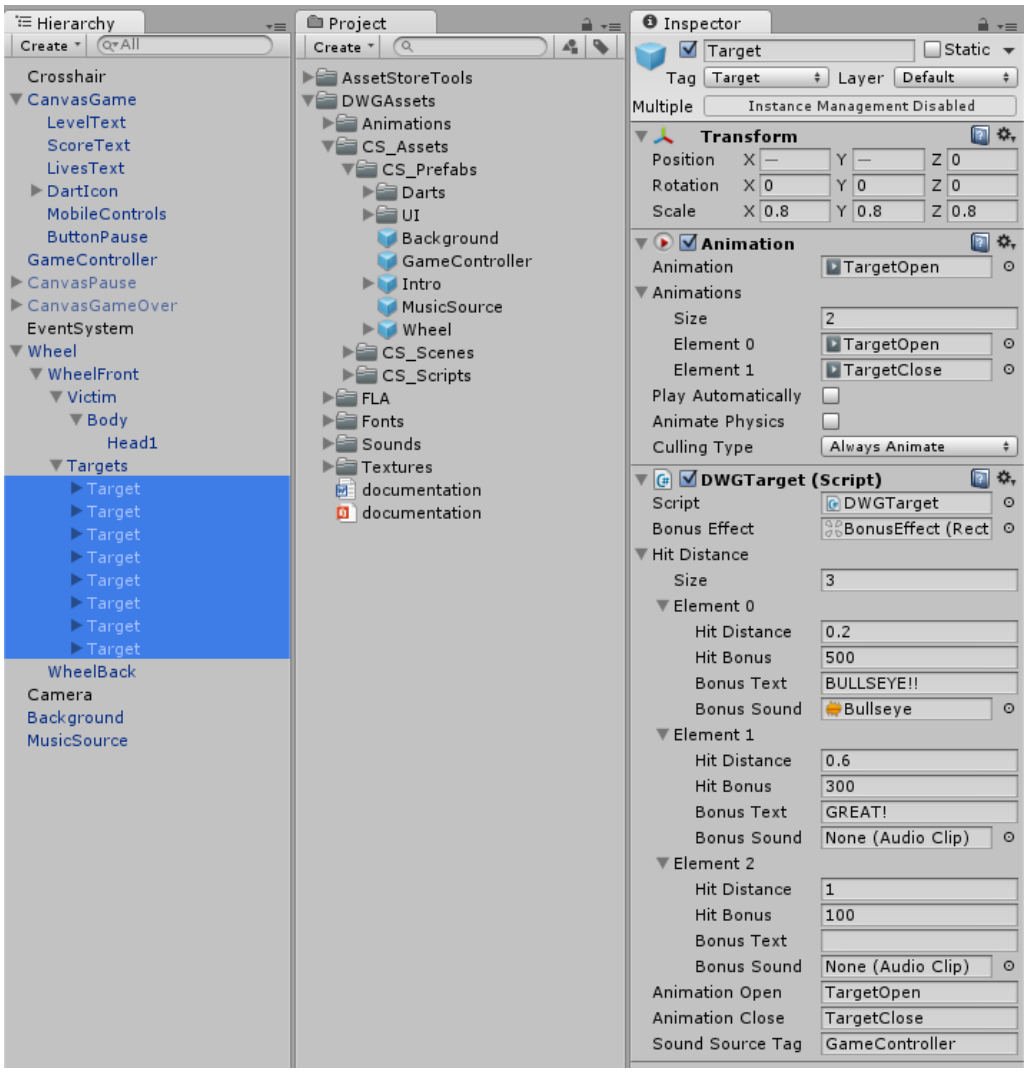
Hurt Time – How many seconds to wait before returning to the normal face.

Ouch Effect – The text effect that displays "OUCH!" when hurting the victim.

Sound Source Tag – The audio source tag from which sounds play.

The Targets

The Targets are the circles you need to hit in order to earn points and win the level. The closer to the center of a target you hit, the more points you get. Each target can be moved around and have its own hit distances and bonuses, so you can make many different target types and not just one size. A target must be tagged as “Target”.



Bonus Effect – The text effect that shows how many points you got for hitting this target.

Hit Distance – This is how far from the center you have to hit in order to get points. You can set the hit bonus and a text that appears alongside the bonus number. Make sure you set this list from closest to farthest.

Animation Open/Close – The animation that closes the target when you hit it, and opens it again at the start of the next level.

Sound Source Tag – The audio source tag from which sounds play.

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.**

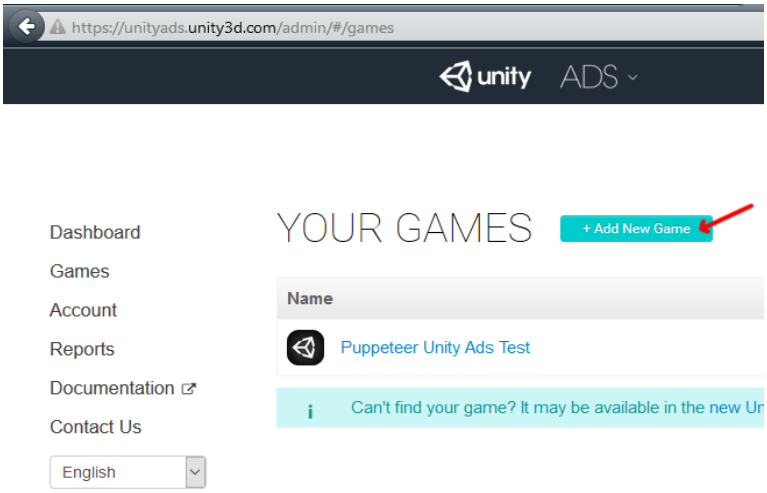
For more info about integrating UnityAds read this:

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

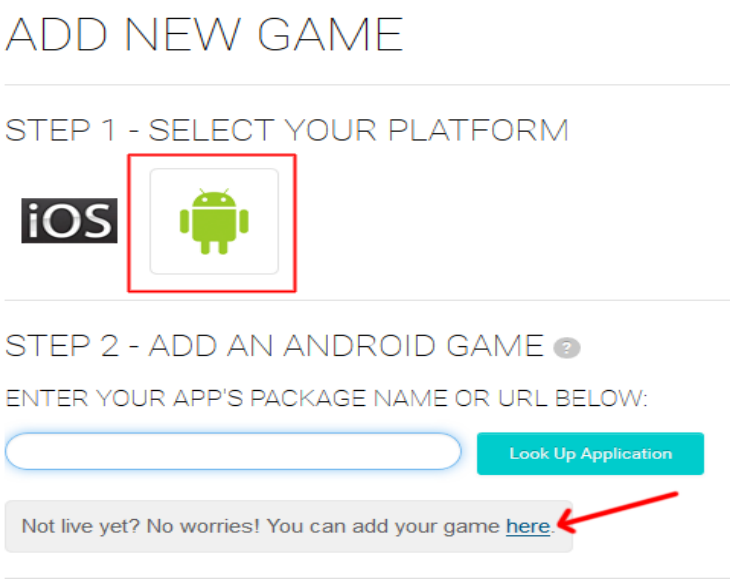
Integrating UnityAds into your project (Unity 4)

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

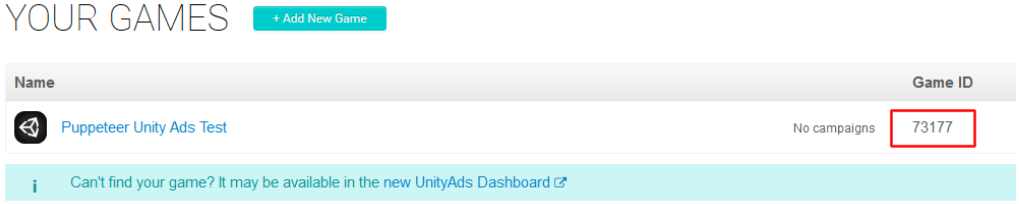
First we need to create our game entry on the UnityAds website. Go to https://unity3d.com/services/ads and create a new game. If you already have your app set and your GameID noted, just skip this part and go straight to importing the UnityAds package into the game.



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.



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.

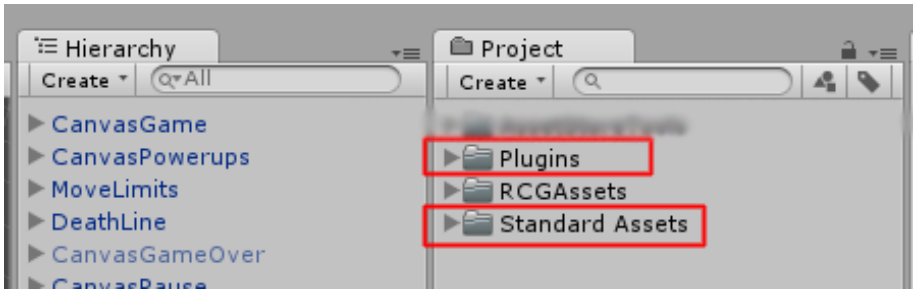


In Unity Editor

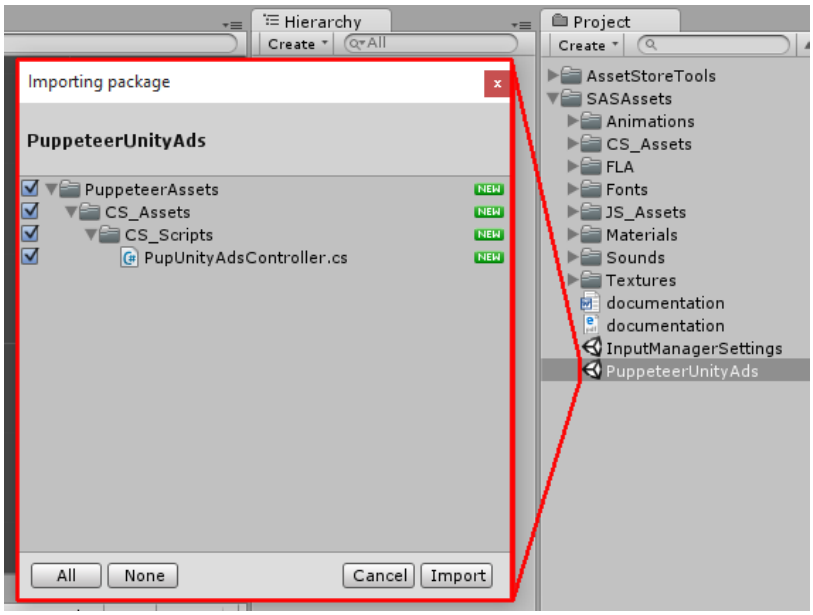
Now we need to import the UnityAds package. Open the Unity Asset Store and download the UnityAds package. Import it into your project.

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

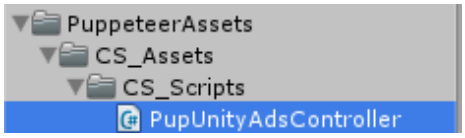
After import you should have two additional folders in your project.

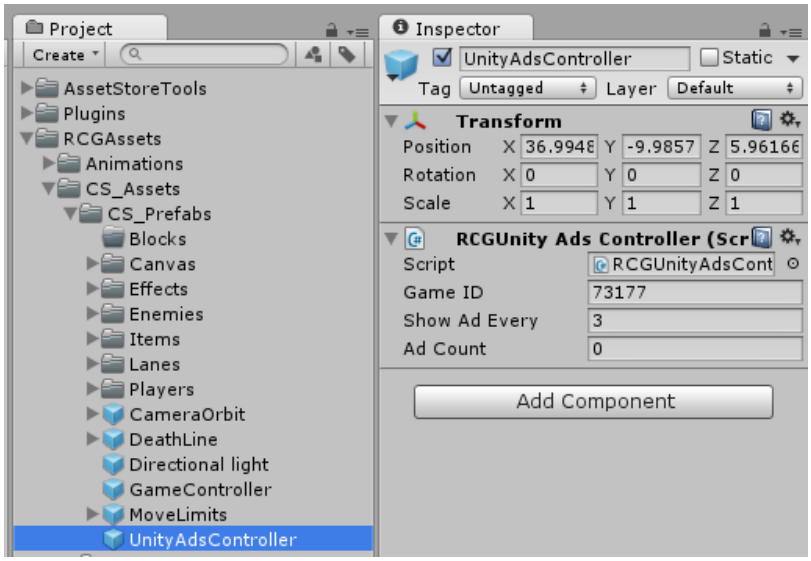


Now we need to bring in the code that integrates the ads into our game. Click on the **PuppeteerUnityAds** package in your project to import it into the game, or choose **Assets > Import Package > Custom Package…** from the top menu and navigate to the **PuppeteerUnityAds** package in your project to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

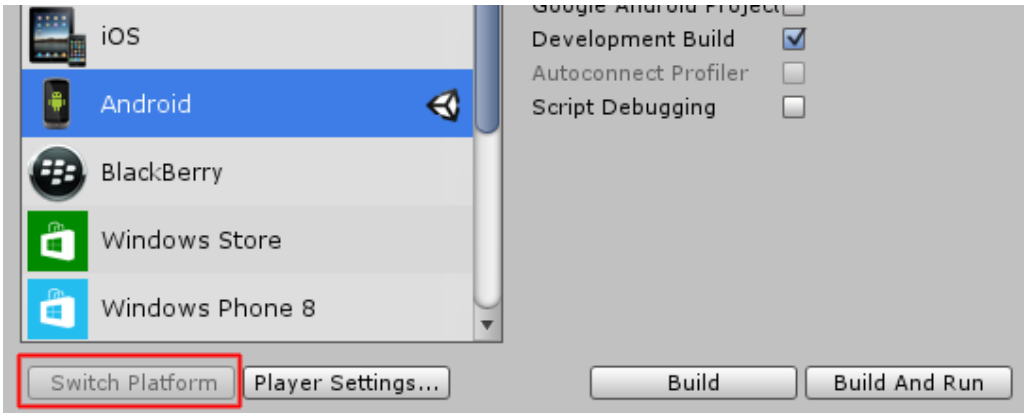


PupUnityAdsController.cs is the main script that links your app to the unityads system. Drag it into your game controller. Now when you look at it you see you can set the GameID of your app, and how often the ads appear. The ad is checked when the level is loaded. "**Show Ad Every**" decides how many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





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.

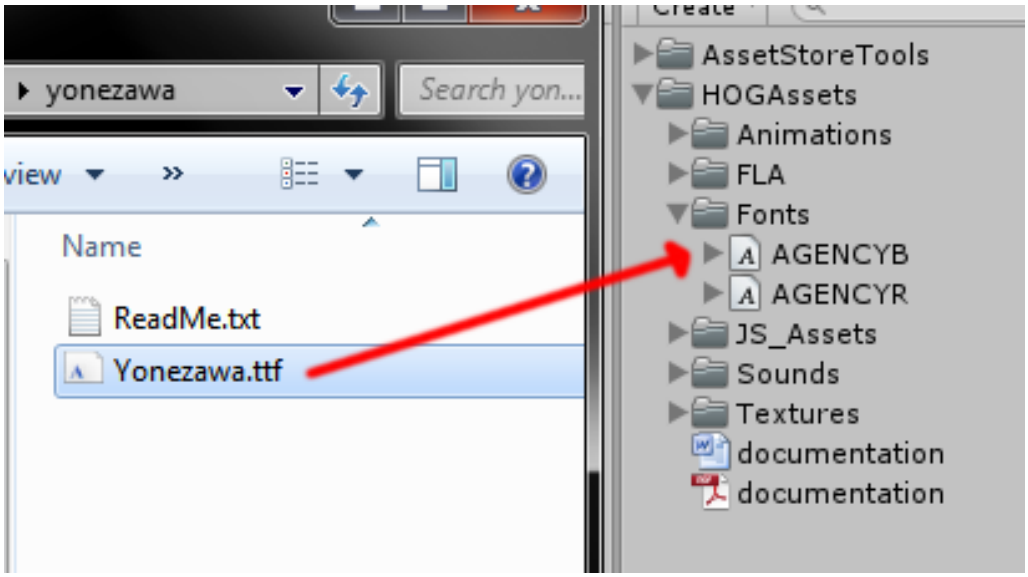
To solve this we must change the texture compression format for iOS. Follow these steps:

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

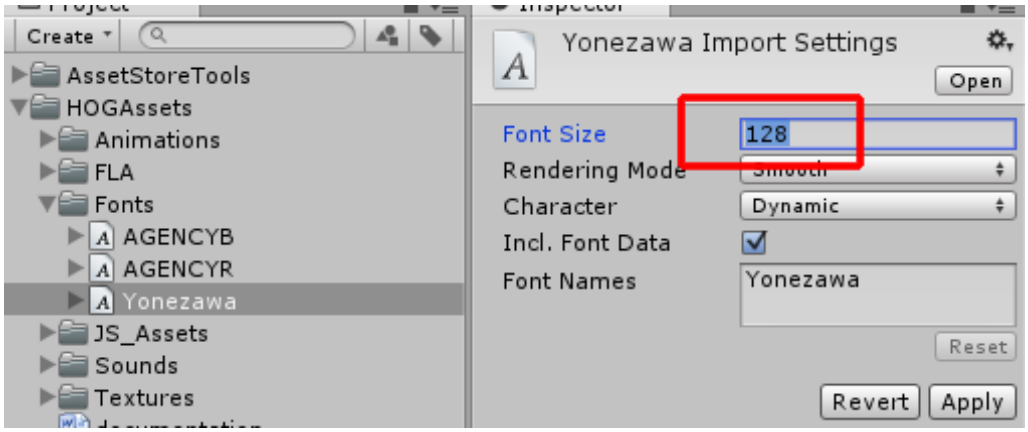
How to change font in the game?

To change a font in the game do the following:

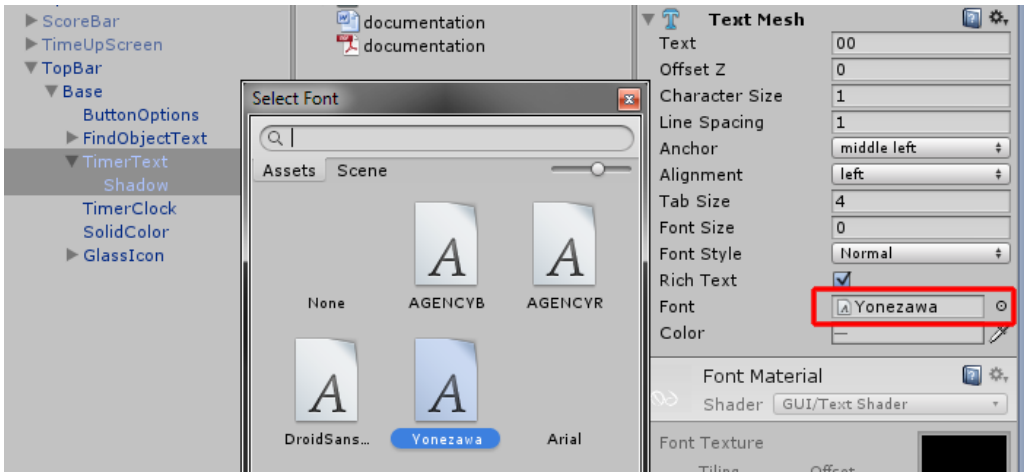
Find a font you like and drag the .ttf file over to the Fonts folder in your game.



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.

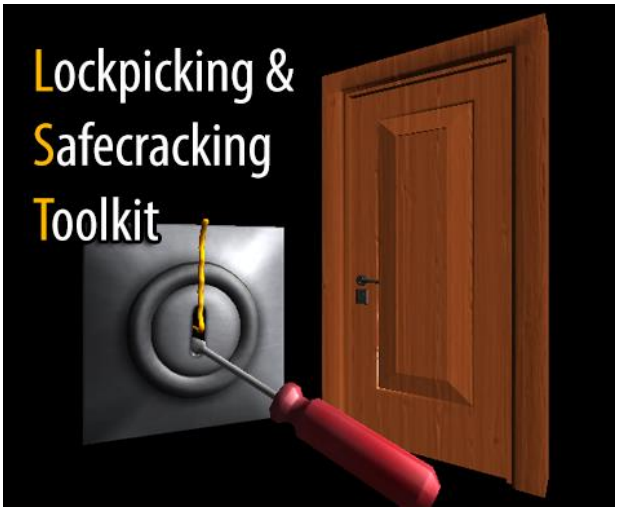


Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and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



More games by Puppeteer

Click here to see the full catalogue of Asset Store files!



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!