Shooting Gallery Template

Game documentation and HowTo guide.

This document contains:

Package Description and features

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

How to Play?

Shoot the moving targets when they appear. Hit them one after the other for a streak bonus. Have Fun!

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.

1.07 (19.07.2016)
- Support for Unity 5.3 and higher versions.
- Better support for UnityAds 5.2 and above.

1.05 (06.01.2016)
- Added versions for Unity 5.1, Unity 5.2, and Unity 5.3.
- Added support for Unity 5.3 SceneManager.

1.04 (04.10.2015)
- Added support for UnityAds along with a quick integration guide.
- Added a final level victory condition (endless mode still available), and a victory screen/sound.

1.0 (27.08.2015)
- Initial version

Credits

Graphics are part of the Kenney.nl assets pack

The font used is Fava Black by Themnific

The sounds are courtesy of the free sound project.

Music from demo (not included in package):

"Circus Waltz FX" Kevin MacLeod (incompetech.com)
Licensed under Creative Commons: By Attribution 3.0
http://creativecommons.org/licenses/by/3.0/

Credits go to these authors for their great sound samples: **harris85, lmr9, xyzr-kx, lemudcrab, mickleness**

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 SGTAssets 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.

Customization Guide

Getting started

Shooting Gallery Game Template (SGT) 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 SGT 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 and throws targets at the player. The Game Controller is also used to calculate the bonus the player gets when hitting a target.

Start Delay – How many seconds to wait before starting the game.

Ready Go Effect – The effect that appears at the start of the game.

Time Left – The timer of the game. If time runs out, it's game over.

Time Text – The text showing the timer.

Moving Targets – A list of targets that will move from one side of the screen to the other.

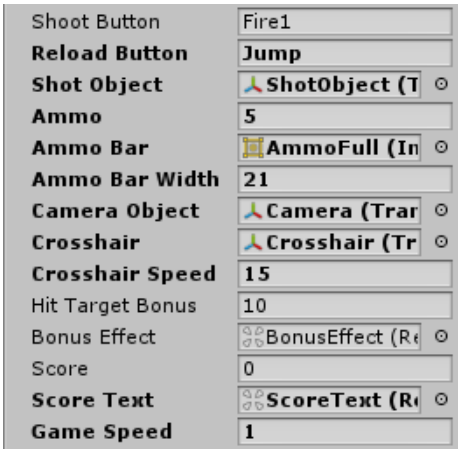
Moving Speed – The movement speed of the moving targets.

Maximum Targets – The maximum number of targets that appear at the same time.

Target Show Area – The horizontal area in which targets can appear. No targets will ever appear outside of this area.

Show/Hide Delay – How long to wait before showing/hiding targets.

Left Edge/ Eight Edge – These collider walls bounce the targets back.



Shoot Button – The keyboard/gamepad button that shoots.
Jump Button – The keyboard/gamepad button that jumps.
Shot Object – The bullet object that appears when shooting. This object collides with targets and hits them.

Ammo – The number of bullets we have.

Ammo Bar – The UI graphics that show how many bullets we have left.

Ammo Bar Width – The pixel width of a single bullet in the Ammo Bar.

Crosshair – The crosshair object you can aim with.

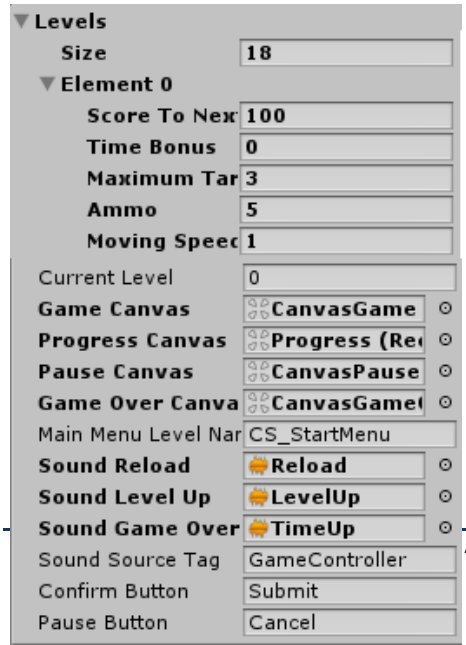
Crosshair Speed – The speed of the crosshair movement when using a keyboard or a gamepad.

Hit Target Bonus – The points we get when shooting a target. Consecutive shots increase the bonus until the targets hide again.

Bonus Effect – The effect that appear where we hit a target.

Score/ScoreText – The current score of the player, and the text object that displays the score.

Game Speed – The current speed of the game. This affects Time.timeScale.



Levels – A list of all the levels in the game. Each level has **Score To Next Level** (How many points we need to win this level), **Time Bonus** (How many seconds are added to the timer when we win a level), **Maximum Targets** (The maximum number of targets that can be on-screen at the same time). **Ammo –** The maximum number of bullets we have in this level. **Moving Speed –** The movement speed of the targets.

Current Level – The current level we are on.

Canvases – These are canvas UI screens. **Game Canvas** appears during gameplay, **Progress Canvas –** Shows the current level and the progress to the next level. **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.

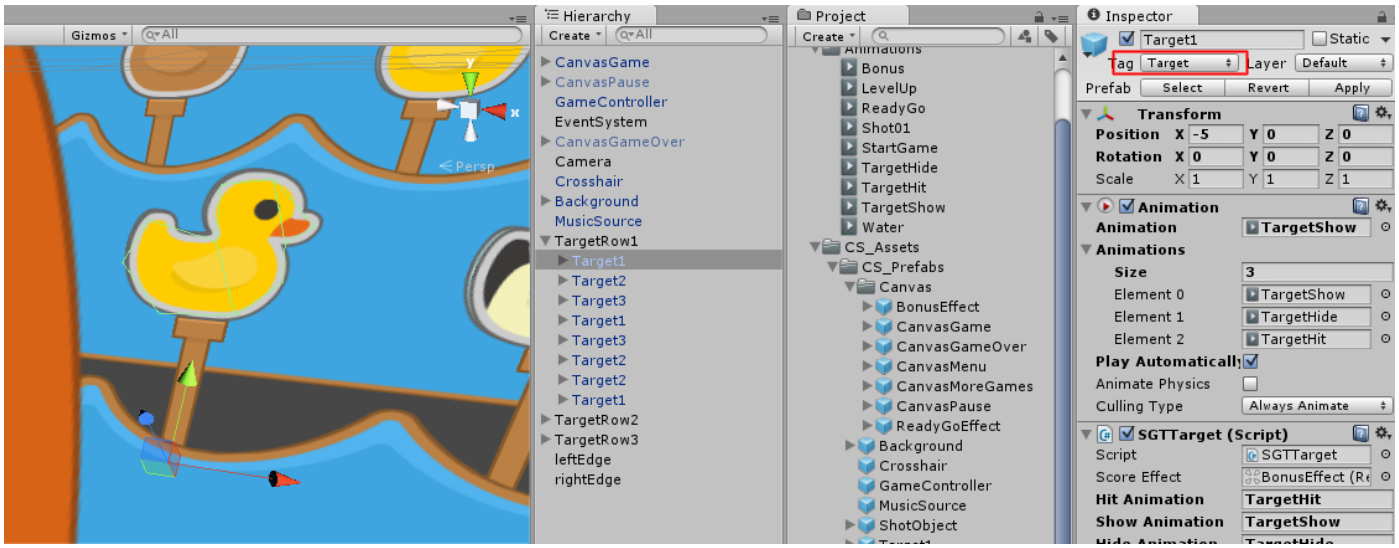
Sounds – Various sounds for leveling up, losing a life, and game over.

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

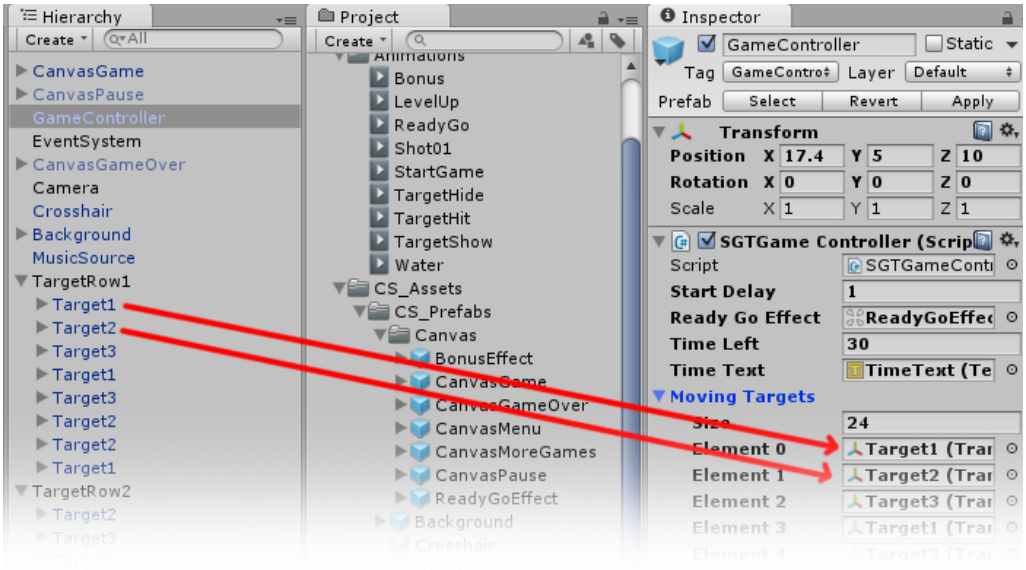
Confirm & Pause Button – These are the Keyboard/Gamepad equivalents to the regular UI buttons. If you press Confirm on Game Over you restart, and if you press Pause you quit the level.

Editing Targets

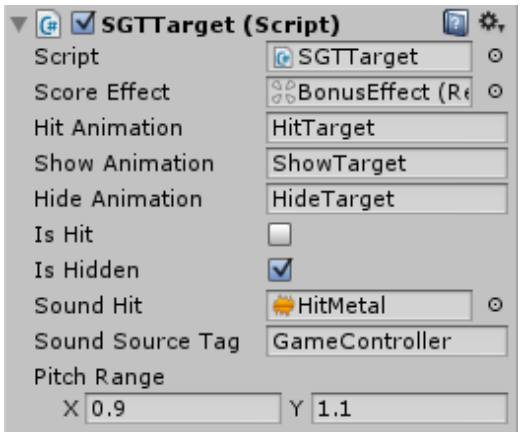
A target is the object that can be hit by a shot. Each target must be tagged with the "Target" tag. Targets start moving in the direction they are looking (based on scale.x value), and when they reach the left or right edges of the game, they reset to the other side.



Targets are assigned to the gamecontroller, which in turn controls when they appear and hide, and how fast they move. You don't need to put them in an organized way, just drag and drop the whole bunch.



The script attached to a target controls what happens when it appears, hides, and gets hits, and the effects when that happens.



Score Effect – The effect that appears when hitting a target.
Hit/Show/Hide Animation – The animations that play when the target is hit, shown, or hidden.

Is Hit – Checks if the target has been hit. A hit target will hide.

Is Hidden – Checks if the target is hidden. Only hidden targets can appear again.

Sound Hit – The sound that play when this target is destroyed.

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

Pitch Range – The pitch range is used to make the sound more varied.

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.**

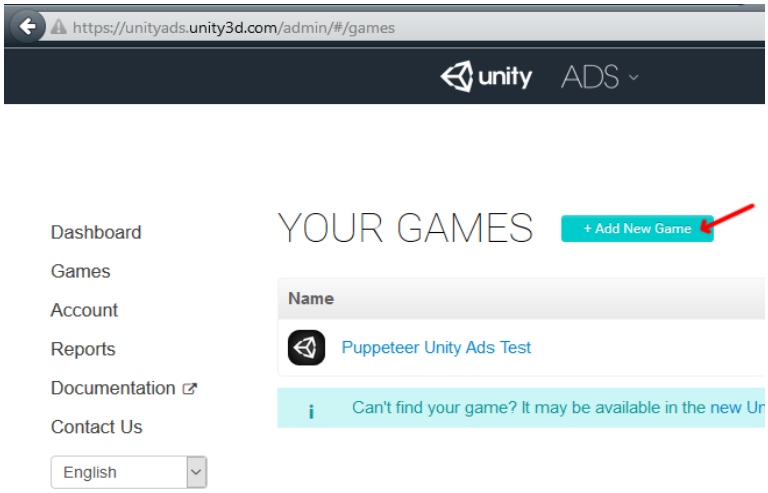
For more info about integrating UnityAds read this:

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

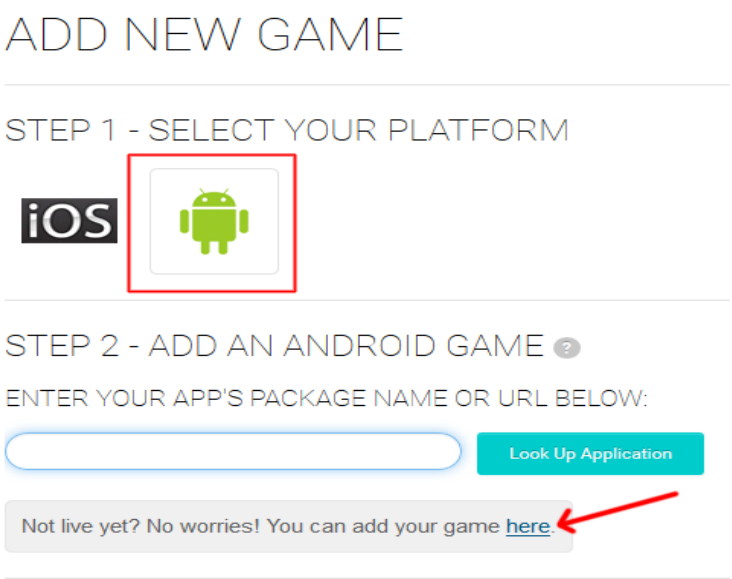
Integrating UnityAds into your project

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

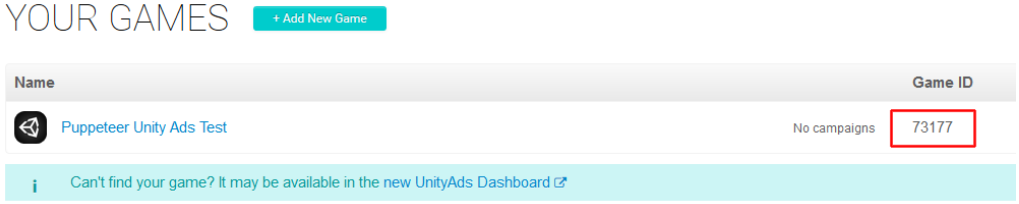
First we need to create our game entry on the UnityAds website. Go to https://unity3d.com/services/ads and create a new game. If you already have your app set and your GameID noted, just skip this part and go straight to importing the UnityAds package into the game.



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.



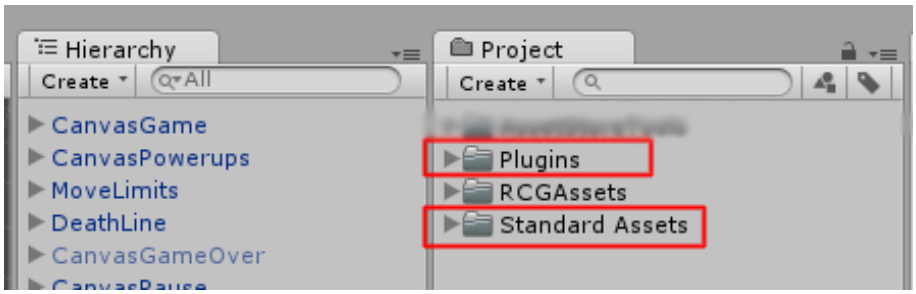
In Unity Editor

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

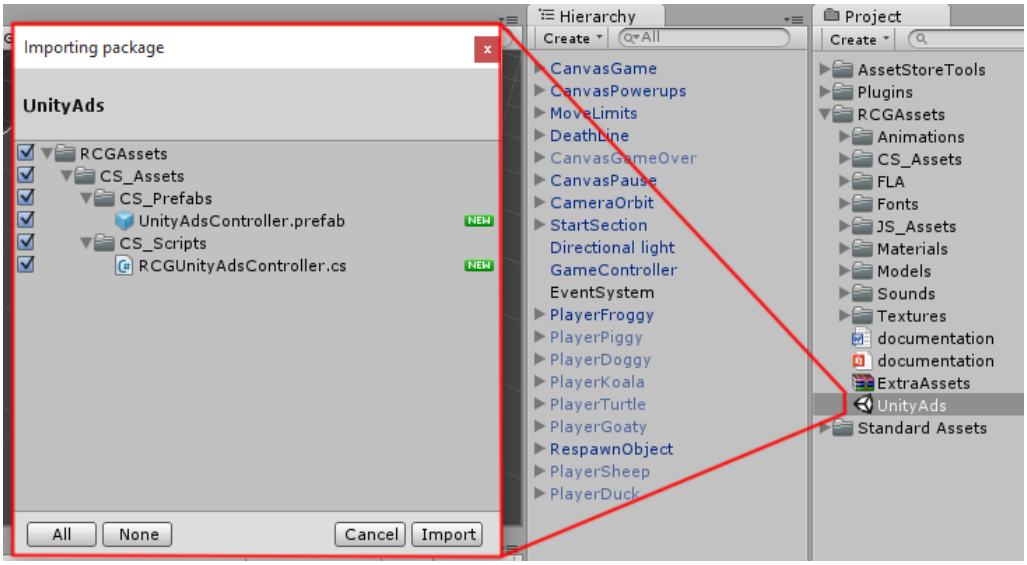
(Notice that this tutorial refers to assets from Road Crossing Game, but the principle is that same for my other packages too)

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

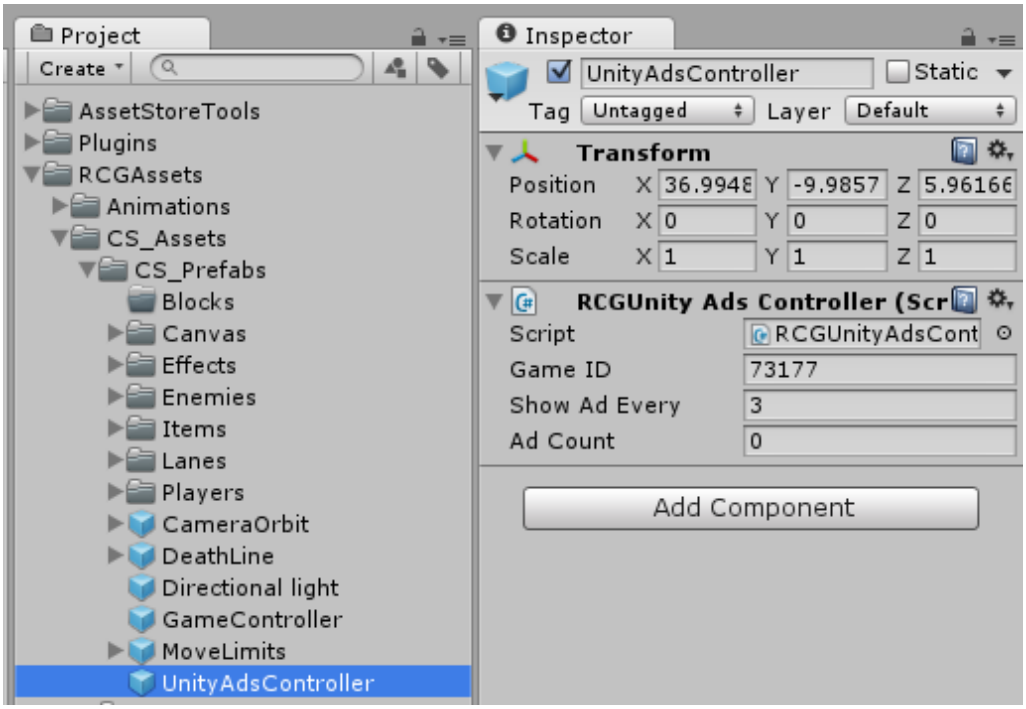
After import you should have two additional folders in your project.



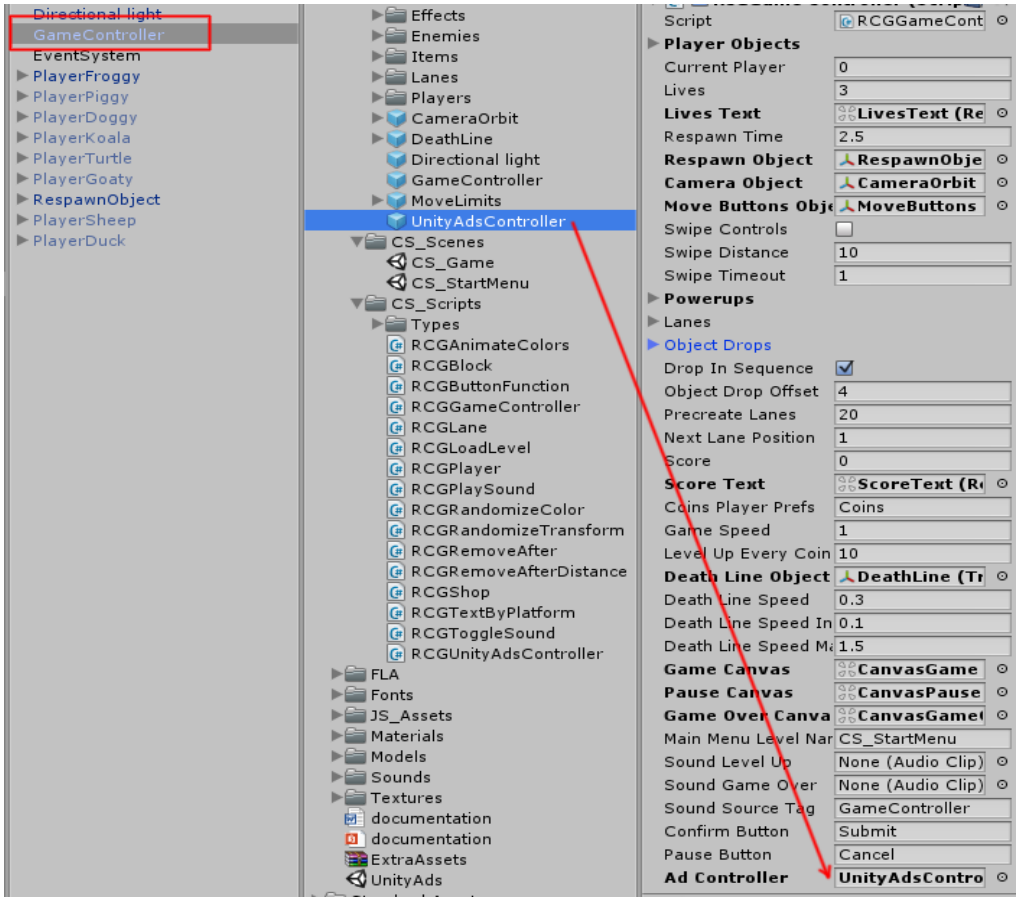
Now we need to bring in the code that integrates the ads into our game. Click on the UnityAds package in RCGAssets to import it into our game. (You can also go through the top menu: Assets > Import Package > Custom Package)



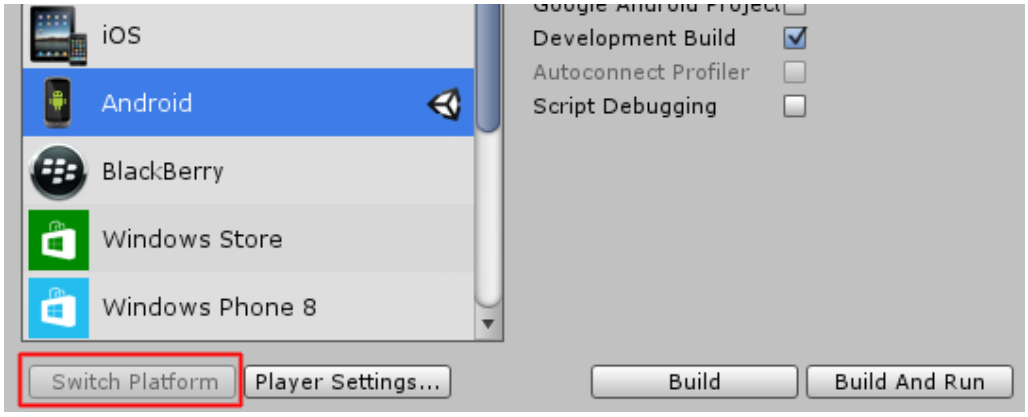
UnityAdsController is the main prefab that links your app to the unityads system. Here you can set the GameID of your app, and how often the ads appear. In the GameController, the ad is activated on Game Over. “Show Ad Every” decides how many times Game Over appears before an ad appears.



In your game scene select the GameController object, then assign the UnityAdsController prefab to it by dragging and dropping in the relevant slot.



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



That's it! Now run the game and reach the Game Over state more than 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 GameOvers.

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.

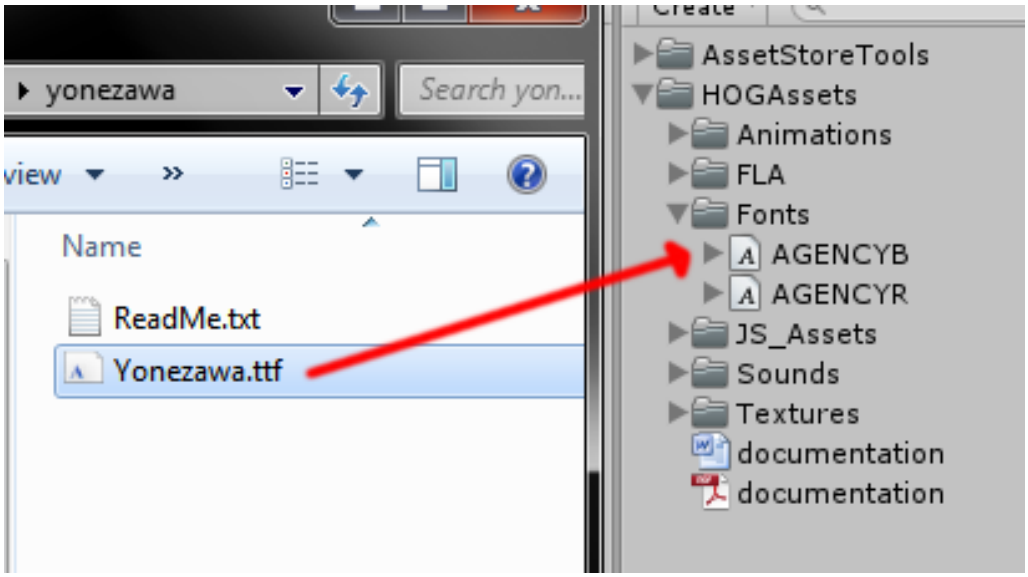
To solve this we must change the texture compression format for iOS. Follow these steps:

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

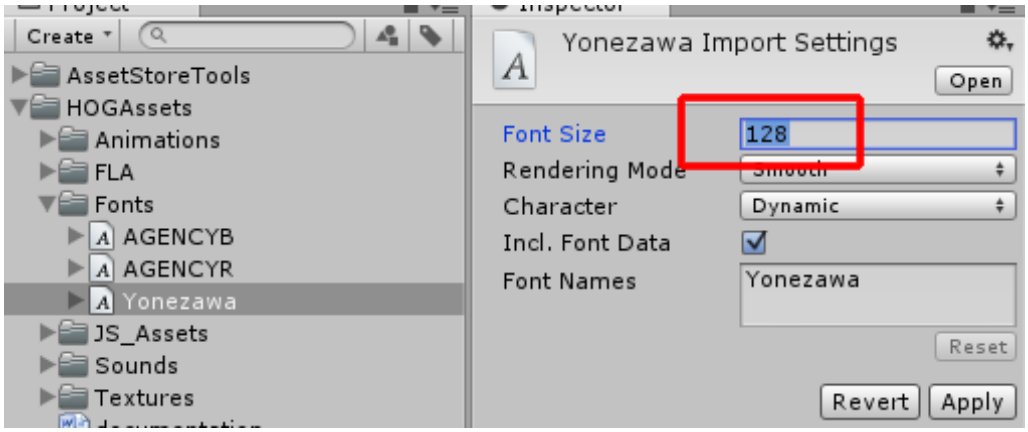
How to change font in the game?

To change a font in the game do the following:

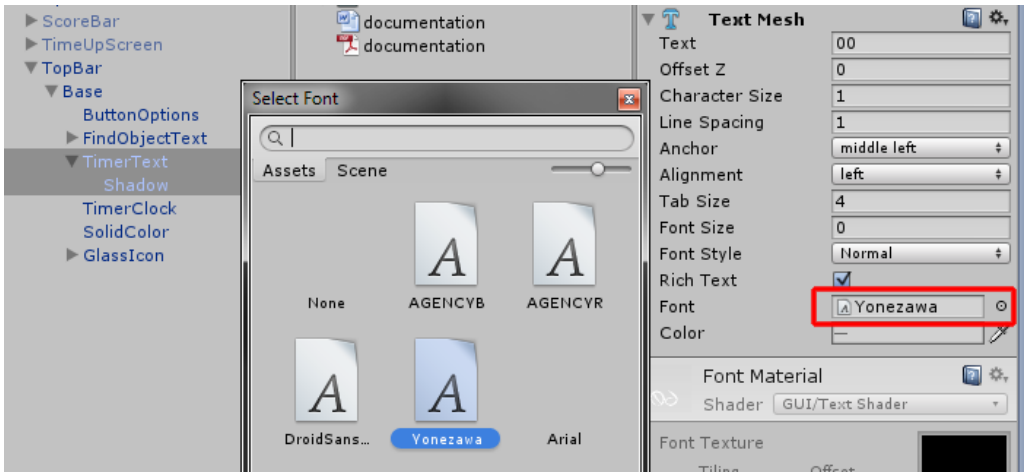
Find a font you like and drag the .ttf file over to the Fonts folder in your game.



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.

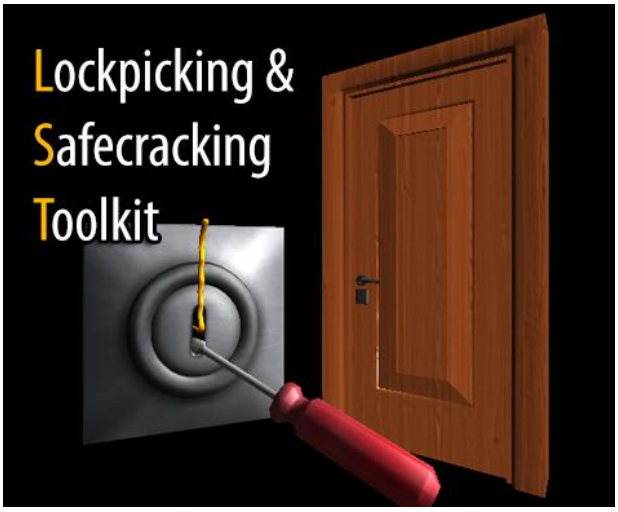


Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and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