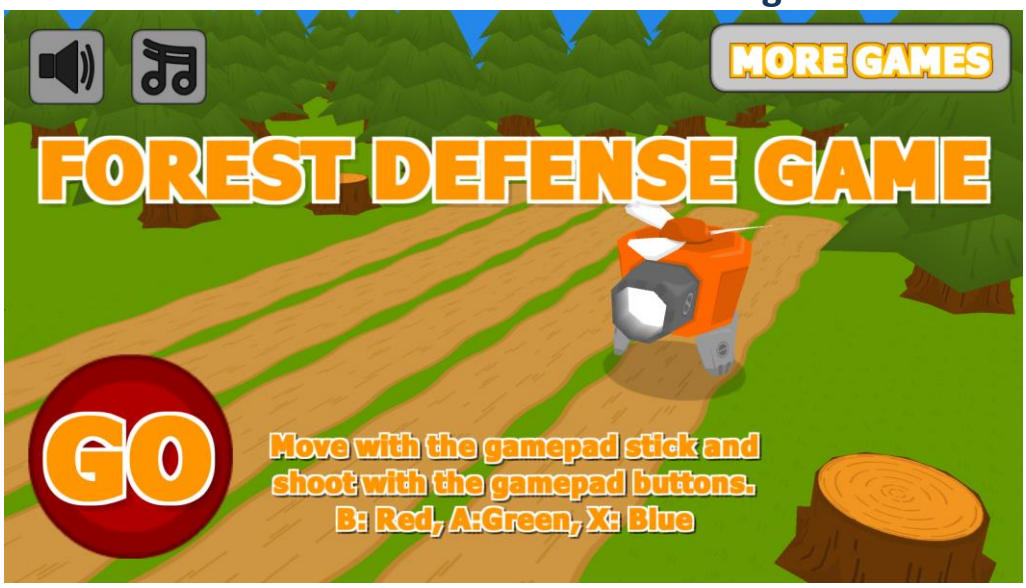
Forest Defense Game

Game documentation and HowTo guide.



This document contains:

Package Description and features

Forest Defense 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 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 A/D) across the lanes and shoot (Mouse buttons) the correct color at the enemy to kill it. Shoot without missing to get a streak bonus.

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.

Current version 1.20

Update history

1.20 (06.12.2018)

- Added option to use Animator component for player, enemies, and items. Kept boss enemies with old Animation system as an example.
- Added item drop art and animation.
- Fixed mobile controls and game over score text.

1.14 (30.06.2018)

- Cleaned up project assets so they can be easily mixed into other projects.

1.13 (04.11.2017)

- Support for Unity 5.5, 5.6, and 2017.

1.12 (10.07.2016)

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

1.1 (25.12.2015)

- Added support for UnityAds along with an integration guide.
- Added support for Unity 5.3 SceneManager.

- Uploaded versions for Unity 4.6.9, 5.1, 5.2, and 5.3

1.07 (11.11.2015)
- Added end of level bosses. These guys are bigger, have more health, and have a tiny cute crown!
- Added spawn sound effects, low health effect.
- You can now only hit enemies once they land on the lane.
- Minor changes and fixes.

1.03 (03.11.2015)
- Added in-game tips which appear once you do various actions (start game, shoot enemy, miss enemy, lose the game). You can also toggle tips on/off.
- Added an option to assign scene transitions when loading scenes. These connect between levels with an animated effect.
- Minor changes.

1.0 (15.10.2015)
- Initial version

Credits

The font used is Fava Black by Themnific

The sounds are courtesy of the free sound project.

Music is **Your Sparking Rubber Soul** from the album **Fractal Planetoid 2.0** (
Public Domain)

Credits go to these authors for their great sound samples: **adcbicycle, oddworld, ggctuk, lmr9, nickmorris, peridactyloptrix, cydon**

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 FDGAssets 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:

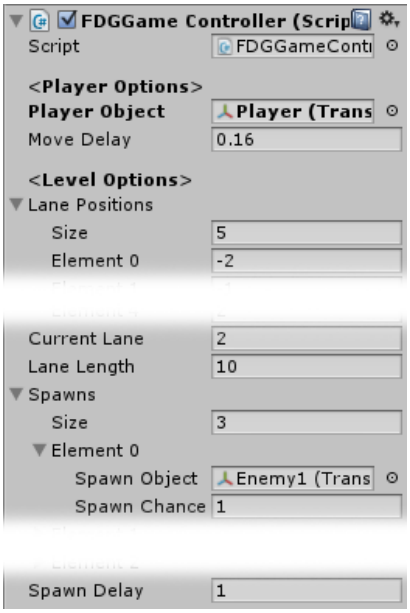
- **Animations:** Holds the animation clips made with Unity's built-in animation system.
- **FLA:** Holds the object graphics made with Flash CS3. These are vector graphics than can be easily scaled without loss of quality and then exported as PNG to be used in Unity.
- **Fonts:** Holds the font used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

Forest Defense Game Template (FDG) 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 FDG 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.



Player Object – The player object, assigned from the scene

Move Delay - How long to wait before the player can move again. Don't set this to 0, or you'll change lanes instantly resulting in movement from the first to the last lane, and back.

Lane Positions – The X positions of the lanes in the game. All lanes are placed at (X,0,0).

Current Lane - The index number of the current lane the player is in. At the start of the game the player is placed in this lane from the list of lane positions above.

Lane Length – The length of a lane. Enemies are spawned at the end of a lane, opposite the player.

Spawns – A list of all the enemies/objects spawned in the game.

- **Spawn Object –** The object to be spawned.
- **Spawn Chance –** How often the object appears. Minimum value is 1.
- **Spawn Delay –** How may seconds to wait between enemy spawns.

Levels – A list of levels in the game, including the number of kills needed to win the game, the speed of enemies, and the spawn rate of enemies.

- **Level Name –** The name of the current level.
- **Enemy Count –** The number of kills needed to win this level.
- **Enemy Speed –** The speed of the enemies in this level.
- **Spawn Delay –** How quickly enemies are spawned in this level.

Current Level – The index number of the current level we are on.
Hit Target Bonus – How many points we get when we hit an enemy. This bonus is multiplied by our kill streak.
Bonus Effect – The bonus effect that shows how much bonus we got when we hit a target.
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).

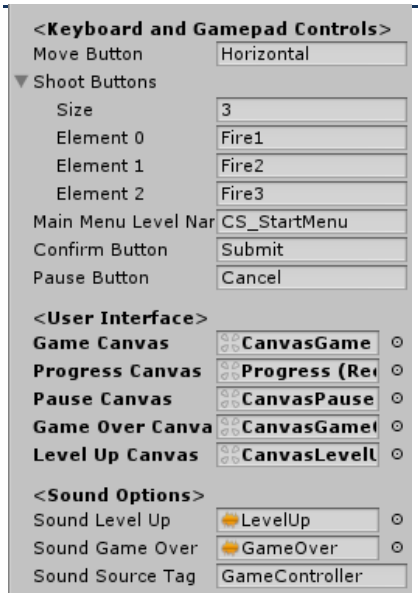
Mobile Shoot Buttons – This canvas assigned from the scene has the shooting buttons in it for mobile devices.

Mobile Move Buttons – This canvas assigned from the scene has the move buttons in it for mobile devices.

Use Swipe Controls – Use swipe controls instead of the move buttons on screen.

Swipe Distance – The swipe distance; How far we need to swipe before detecting movement.

Swipe Timeout – How long to wait before the swipe command is cancelled.



Move Button – The button that moves the player left/right. This is defined in the Input Manager.

ShootButtons – A list of buttons that shoot. Each button index number corresponds to the shot type in the player script.

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.

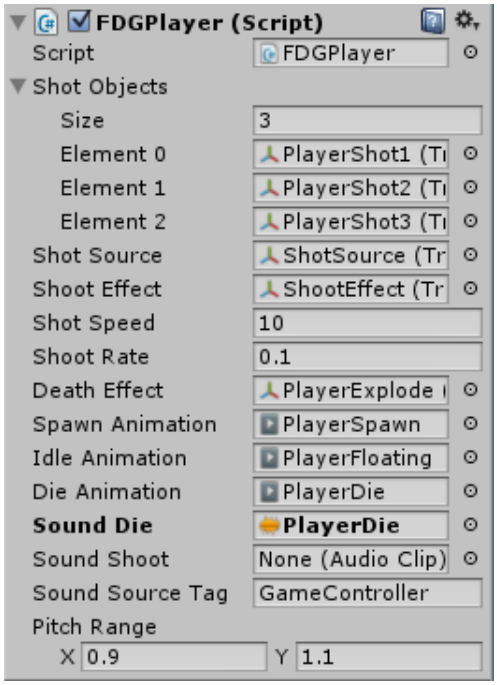
User Interface – Various canvases for the UI, assign them from the scene.

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 across the lanes and shoot at enemies. It can have several shot types which correspond to the shot buttons assigned in the gamecontroller. Here is what you can change in the player component:



Shot Objects - The shot types that this player can shoot. These correspond to the shoot buttons defined in the game controller and in the UI buttons.

Shot Source - The source from which shots are fired. The muzzle of a gun.

Shoot Effect - The muzzle effect when shooting.

Shot Speed - The speed of the shot.

Shoot Rate - Your fire rate, or how often you can shoot.

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

Animations – These are various animation clips that play during the game.

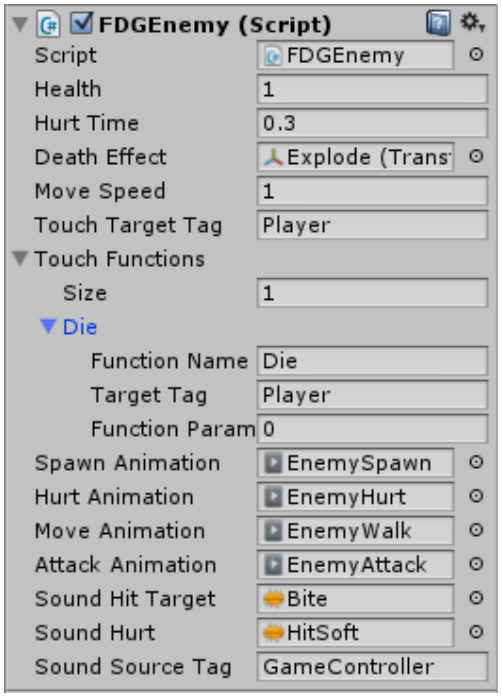
Sounds – These are various sound effects that play during the game.

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

Pitch Range - A random range for the pitch of the audio source, to make the sound more varied.

Editing the Enemy

The enemy is spawned at the end of a lane randomly. It has a health value and will kill the player if it passes the lane start or touches the player. Here is what you can change in the enemy component:



Health – The health value of the enemy. If the enemy is shot with the correct type of shot it will be damaged. If health reaches 0 the enemy dies.

Hurt Time - How long the hurt effect is active when the enemy gets hurt.

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

Move Speed - The movement speed of the enemy. This is controlled through the Levels in the Game Controller.

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.

Animations – These are various animation clips that play during the game.

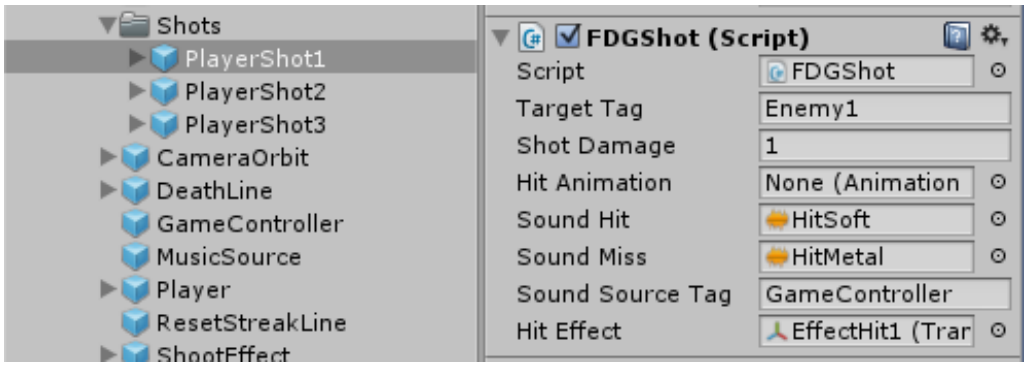
Sounds – These are various sound effects that play during the game.

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

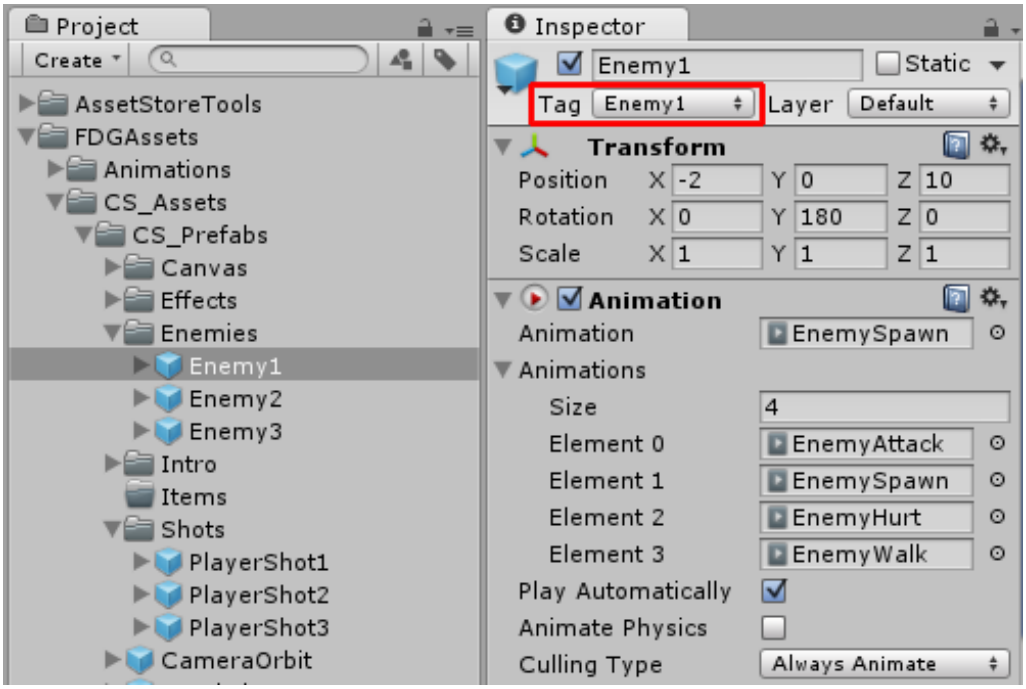
Making Shots Kill Enemies

The way the shooting system works in this game is that it gives each shot a target tag which it can hit, and it gives each enemy a tag name that can be hit by the shot.

For example if we look at **PlayerShot1** we can see that it has a **Target Tag** called **Enemy1**.



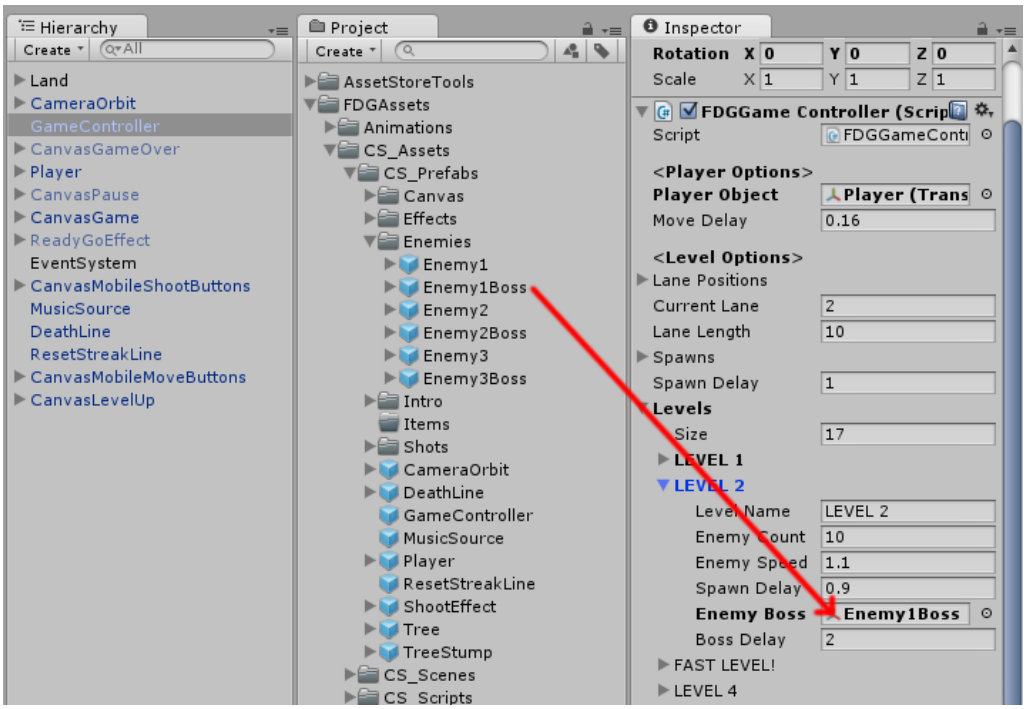
And then when we look at the **Enemy1** prefab we can see that it has a tag of **Enemy1.**



This means that whenever **PlayerShot1** touches **Enemy1** it hurts it.

What about bosses?

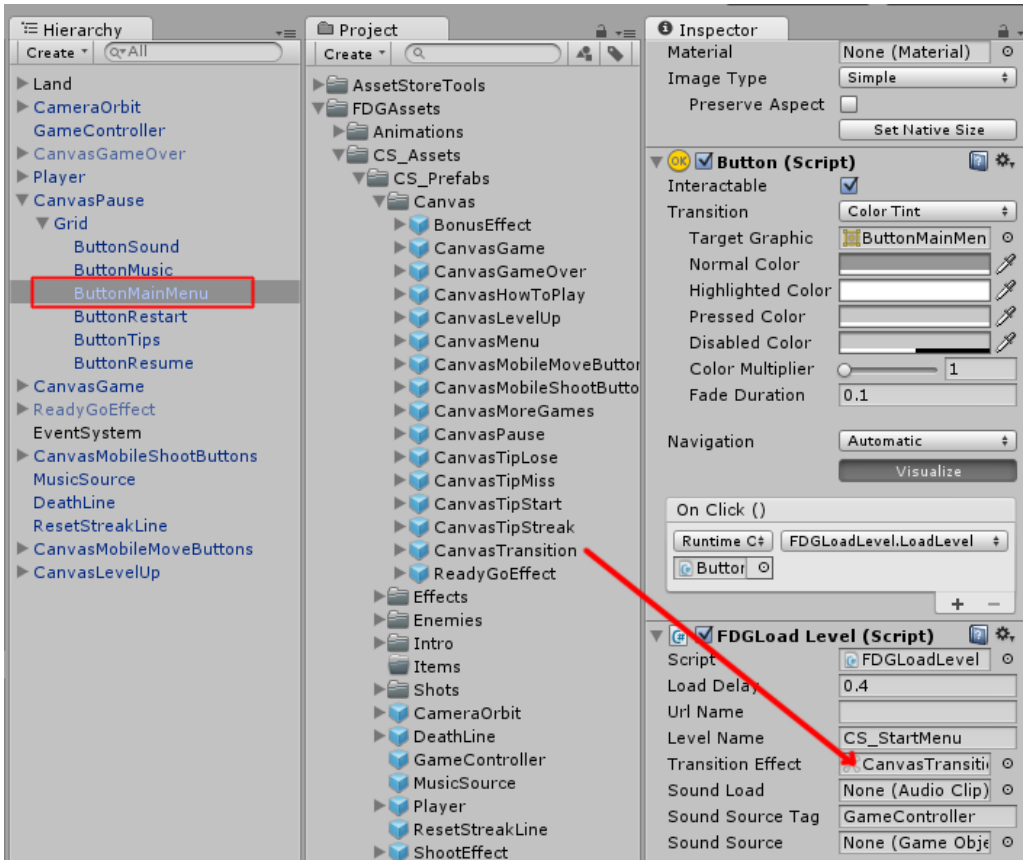
Bosses are just like normal enemies, except they have higher health and they appear at the end of a level. In each level you can choose to assign a boss or not.



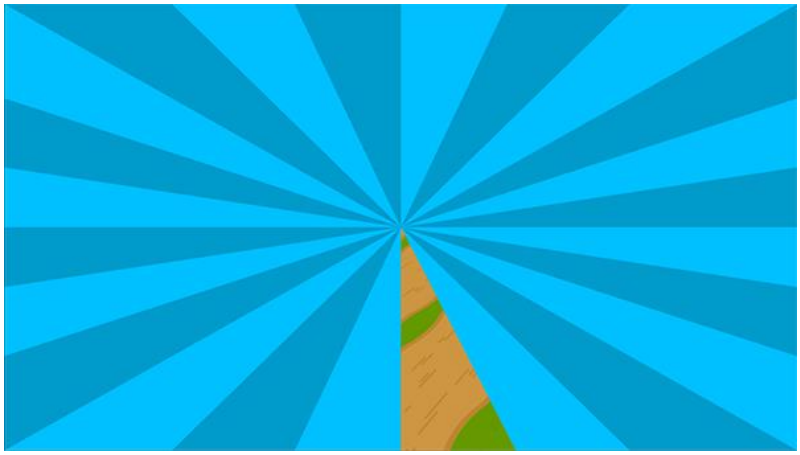
Simply drag and drop the boss enemy in the Enemy Boss slot. You can also set how many seconds to wait before spawning the boss after all other enemies have been spawned.

Adding Scene Transitions

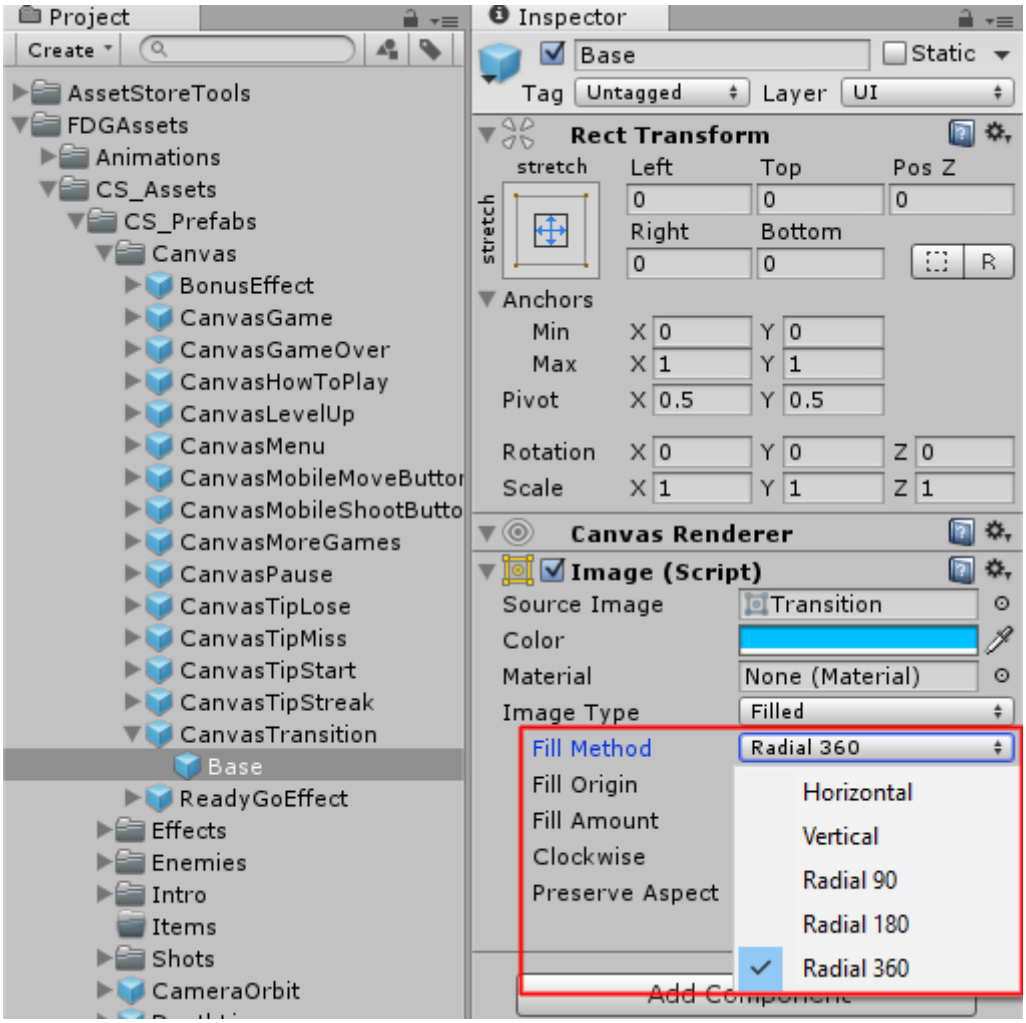
You can add animated transitions between scenes. To do this, choose any of the buttons that load levels (such as the StartGame button in the menu, or the MainMenu button in the game level), and then assign a transition effect to the component:



The default transition effect is a circular swipe effect that covers the screen and then reveal the screen on the loaded level.



You can also customize it to swipe differently by editing the Image component:



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.**

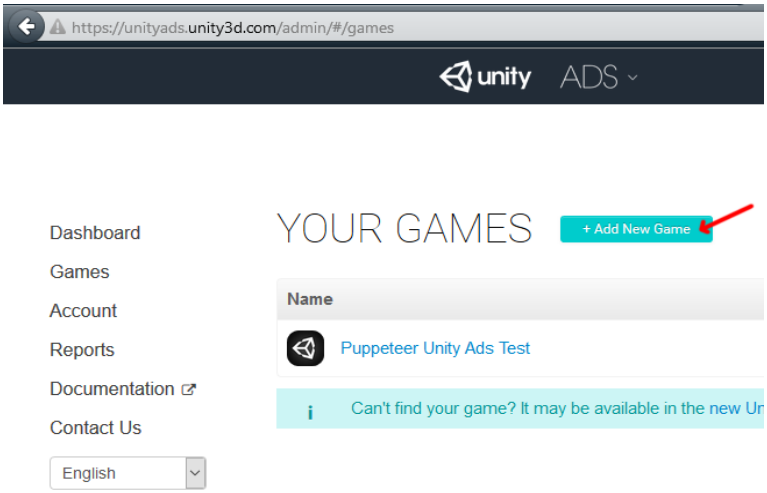
For more info about integrating UnityAds read this:

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

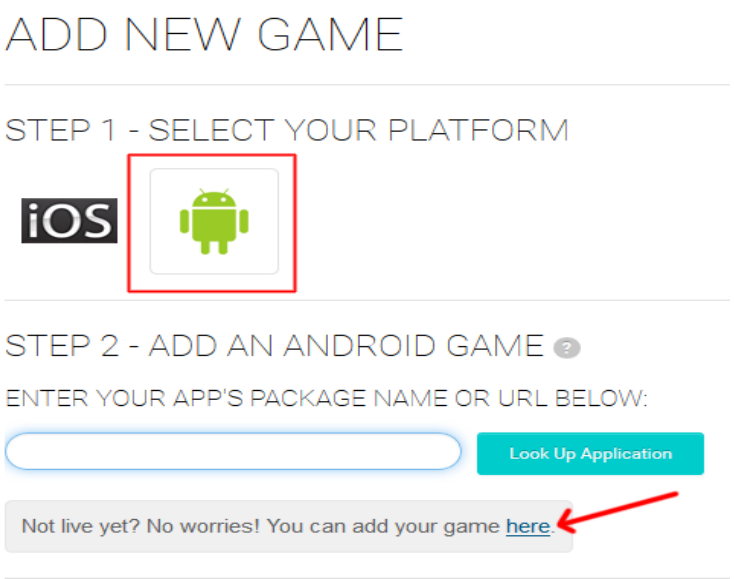
Integrating UnityAds into your project (Unity 4)

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

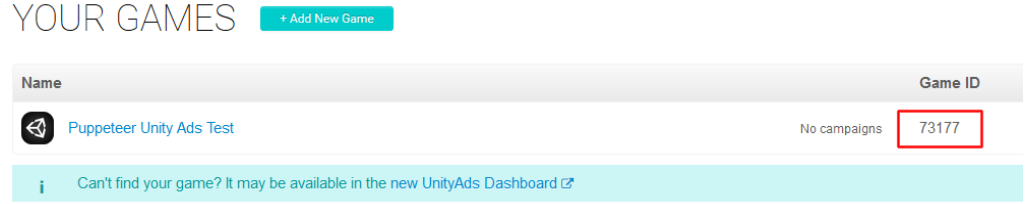
First we need to create our game entry on the UnityAds website. Go to https://unity3d.com/services/ads and create a new game. If you already have your app set and your GameID noted, just skip this part and go straight to importing the UnityAds package into the game.



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.

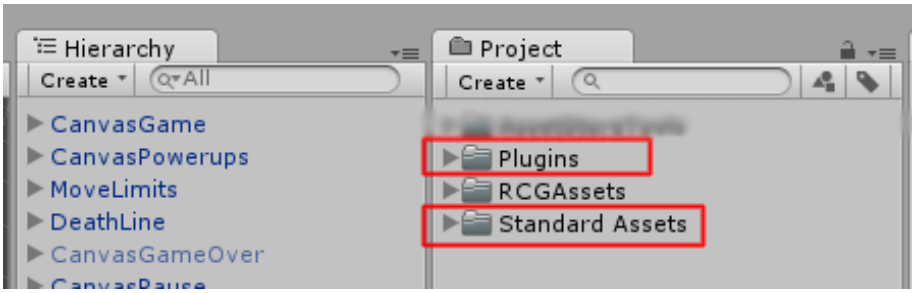


In Unity Editor

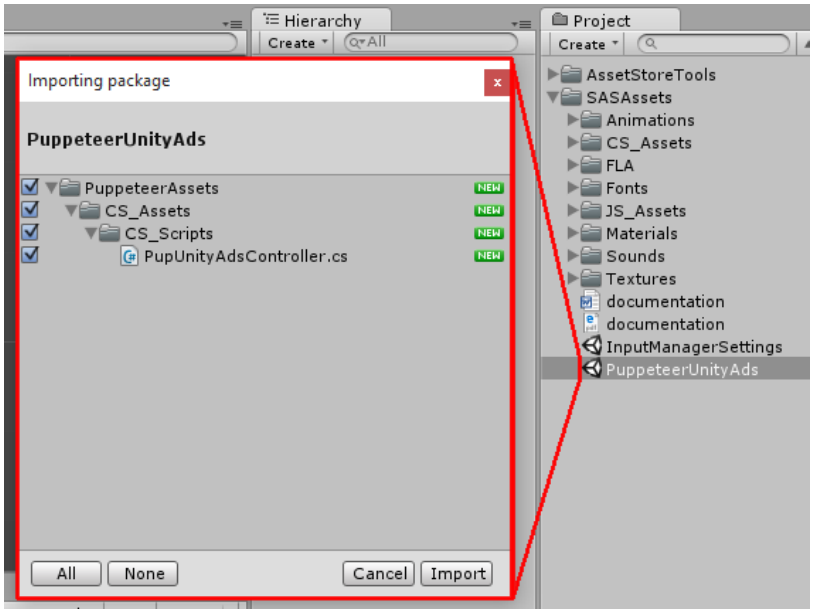
Now we need to import the UnityAds package. Open the Unity Asset Store and download the UnityAds package. Import it into your project.

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

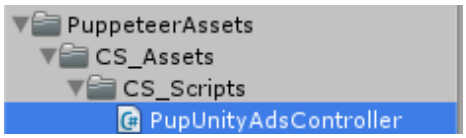
After import you should have two additional folders in your project.

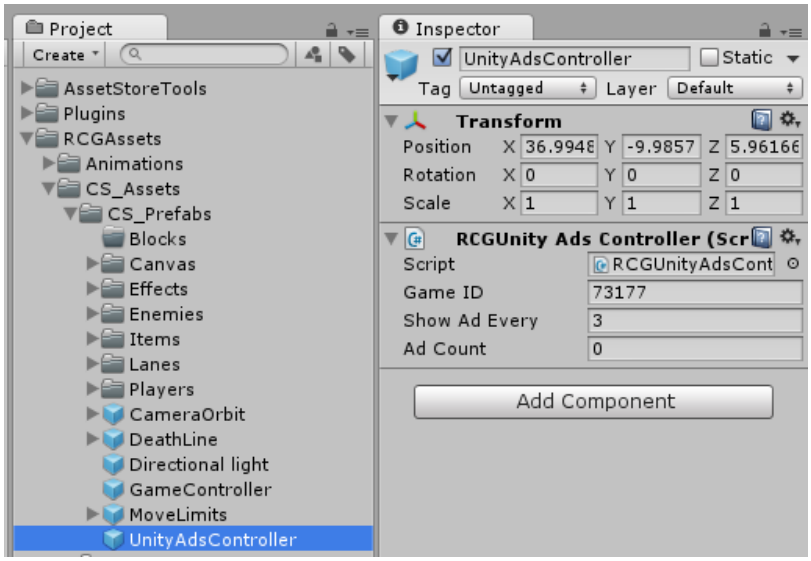


Now we need to bring in the code that integrates the ads into our game. Click on the **PuppeteerUnityAds** package in your project to import it into the game, or choose **Assets > Import Package > Custom Package…** from the top menu and navigate to the **PuppeteerUnityAds** package in your project to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

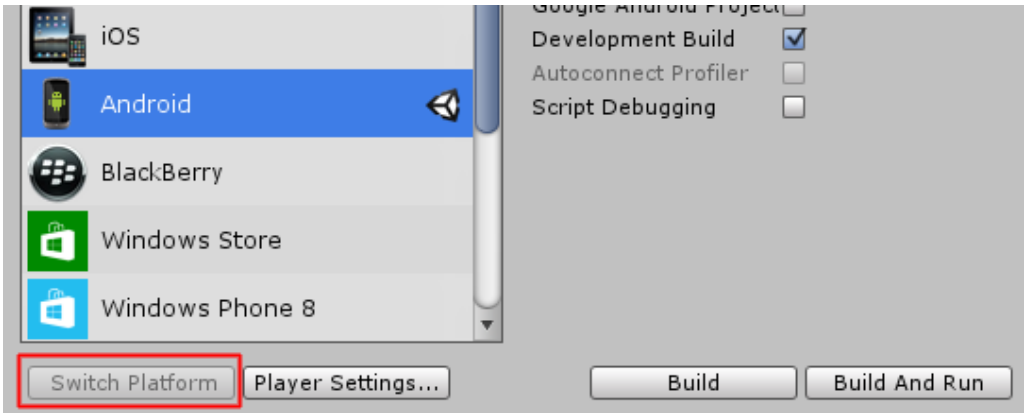


PupUnityAdsController.cs is the main script that links your app to the unityads system. Drag it into your game controller. Now when you look at it you see you can set the GameID of your app, and how often the ads appear. The ad is checked when the level is loaded. "**Show Ad Every**" decides how many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

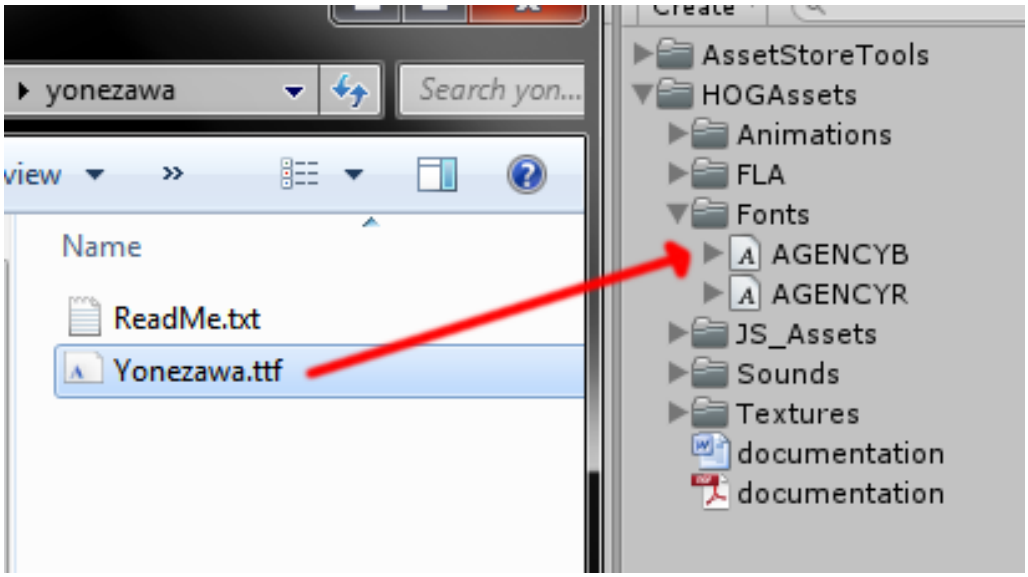
To solve this we must change the texture compression format for iOS. Follow these steps:

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

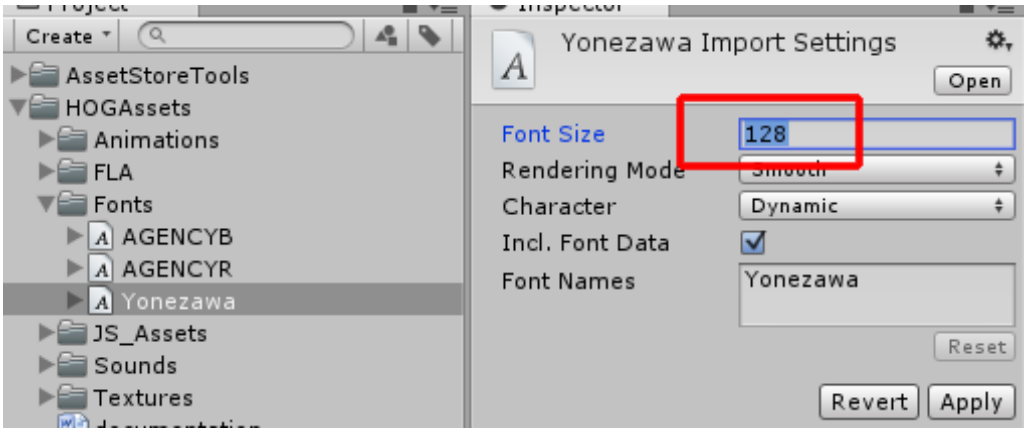
How to change font in the game?

To change a font in the game do the following:

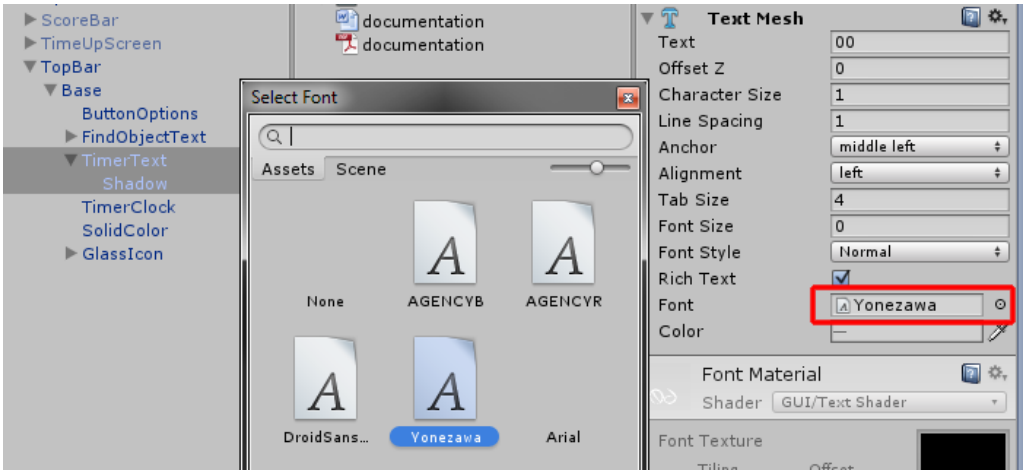
Find a font you like and drag the .ttf file over to the Fonts folder in your game.



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.

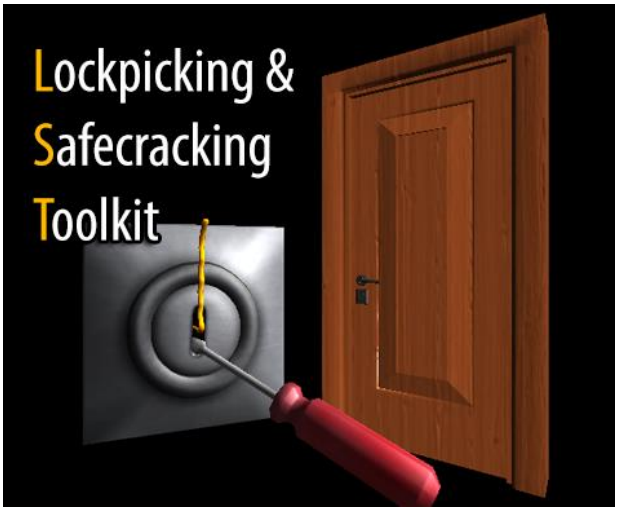


Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and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