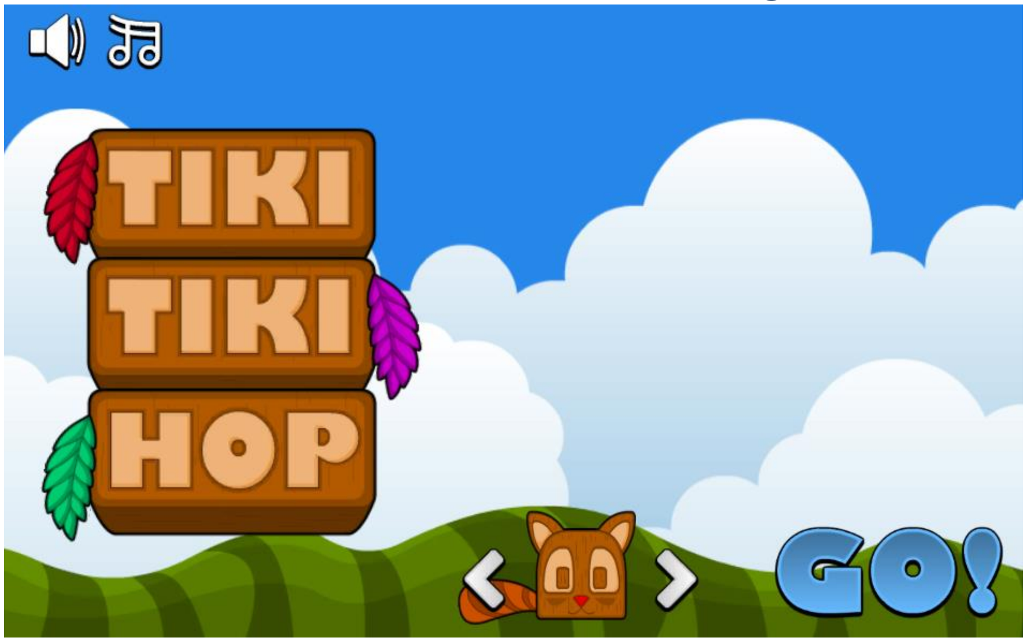
Infinite Platform Hopper Template

Game documentation and HowTo guide.



This document contains:

Package Description and features

Infinite Platform Hopper Game is a full Unity template ready for release. It is compatible with mobile as well as standalone and webplayer. It is similar in genre to classic arcade games, as well as modern infinite runners.

How to Play?

Use the keyboard arrows or the mouse to move around, collect coins and avoid being killed by the many deadly obstacles. The game controls also fit mobile and console without need for further coding.

Try the webplayer

Current version 1.28

Update history

1.28 (25.12.2015)
- 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.
- Minor fixes.

(1.25) 17.06.2015
- New leveling up system increases difficulty as you progress: Leveling up increases the height gap of columns and increases the chance of moving columns to appear.

(1.22) 17.06.2015
- Added powerups to the game that show with a circular timer when you pick them up. Two example powerups included: "2X" gives you twice as many points for 10 seconds, and "Giant" makes you much larger for 10 seconds so you don't fall easily.
- Fixed the double landing bug. You can now land on the same platform more than once.

(1.19) 19.05.2015
- Added an option to disable auto jumping when reaching the maximum jump power.
- Added a power bar that fills up as your jump power increases.

(1.17) 08.05.2015
- Entire game ported to C#. Project now contains both JS and C# in the same package.

(1.07) 19.04.2015

- Added unlockable characters. Collect tokens throughout the level to unlock new characters in the player selection screen.
- Added perfect streaks. Try to land closest to the center of a column to start a streak. Your bonus is multiplied for each consecutive perfect landing.
- Added a video tutorial on how to create new characters using Flash CS3, and setting them up in Unity.
- Added 2 new characters to the game, Doggy and Piggy.
- Optimized sound levels.

(1.0) 28.03.2015
- Initial version

Credits

The sounds are courtesy of the free sound project.

Music is a clip from Waterford by Kevin MacLeod

Credits go to these authors for their great sound samples: **tristan, issalcake, adam-n, fins, freefire66, blukotek**

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 IPHAssets folder using Unity3D 4.6.3 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:

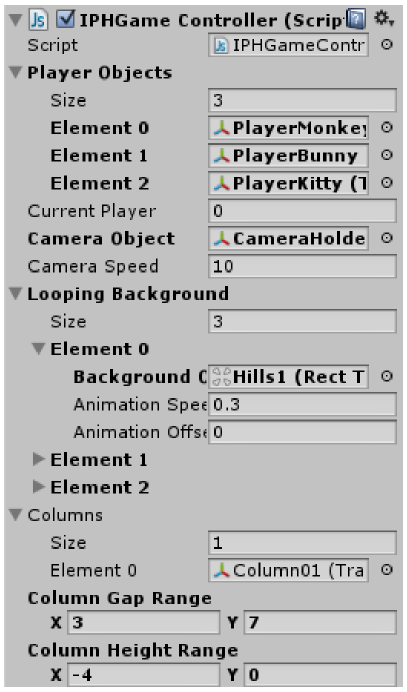
- **Animations:** Holds the animation clips made with Unity's built-in animation system.
- **FLA:** Holds the object graphics made with Flash CS3. These are vector graphics than can be easily scaled without loss of quality and then exported as PNG to be used in Unity.
- **Fonts:** Holds the font used in the game, AGENCYB.
- **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 (IPH) 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 IPH 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 columns, and also makes the camera follow the player. The Game Controller is also used to calculate the bonus the player gets when landing on a column.



```
IPHGame Controller (Scrip
Script                IPHGameContr
Player Objects
  Size                3
  Element 0           PlayerMonke
  Element 1           PlayerBunny
  Element 2           PlayerKitty (1
Current Player        0
Camera Object         CameraHolde
Camera Speed          10
Looping Background
  Size                3
  Element 0
    Background C      Hills1 (Rect T
    Animation Spee    0.3
    Animation Offs    0
  Element 1
  Element 2
Columns
  Size                1
  Element 0           Column01 (Tra
Column Gap Range
  X 3                 Y 7
Column Height Range
  X -4                Y 0
```

Player Objects – A list of player objects. At the start of the game, the current player is kept active while all the others are deactivated. The current player is defined in the **Player Selection component** and registered in Player Prefs.

Camera Object – The camera that chases the player. If no camera is assigned to this field, the main camera in the scene is assigned automatically.

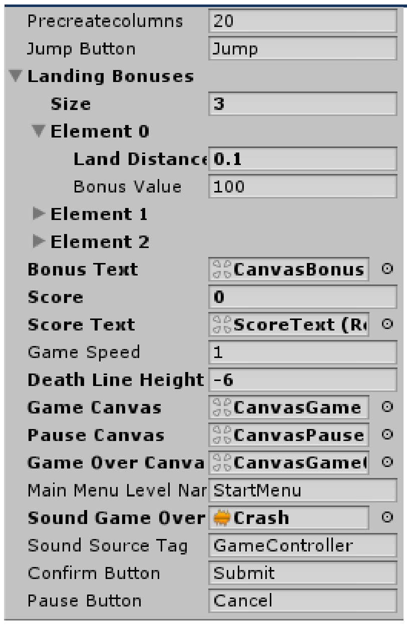
Camera Speed – How fast the camera chases the player. The camera only moves horizontally.

Looping Background – This class defines background elements that are animated to loop. You can set the **Speed** and **Offset** of the animation to create a parallax effect of passing background objects. Note that the background elements are Images in a canvas and are stretched to fill the screen area.

Columns – a list of objects that are created as the player moves forward. These are the columns that the player needs to land on.

Column Gap Range – The horizontal distance between each two columns. This number is random between X and Y.

Column Height Range – The vertical distance between each two columns. This number is random between X and Y.



Precreatecolumns	20
Jump Button	Jump
▼ Landing Bonuses	
Size	3
▼ Element 0	
Land Distanc	0.1
Bonus Value	100
▶ Element 1	
▶ Element 2	
Bonus Text	CanvasBonus
Score	0
Score Text	ScoreText (R
Game Speed	1
Death Line Height	-6
Game Canvas	CanvasGame
Pause Canvas	CanvasPause
Game Over Canva	CanvasGame(
Main Menu Level Nar	StartMenu
Sound Game Over	Crash
Sound Source Tag	GameController
Confirm Button	Submit
Pause Button	Cancel

Precreate Columns – How many columns to create at the start of the game.

Jump Button – The button you need to press to charge the jump power. Releasing this button will launch the player.

Landing Bonuses – This class defines a list of bonuses the player gets based on how far from the center of a column he lands. Each element defines a **landing distance** from the center of a column, and the **bonus value** it adds to the score. If the player lands closer to the center of the column than a certain defined distance, he gets the corresponding bonus.

Bonus Text – The text object that displays the bonus the player gets when landing on a column. This is a Text within a Canvas.

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

Game Speed – The overall speed of the game.

Death Line Height – The Y position of the death line. If the player moves below this point, he dies.

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.

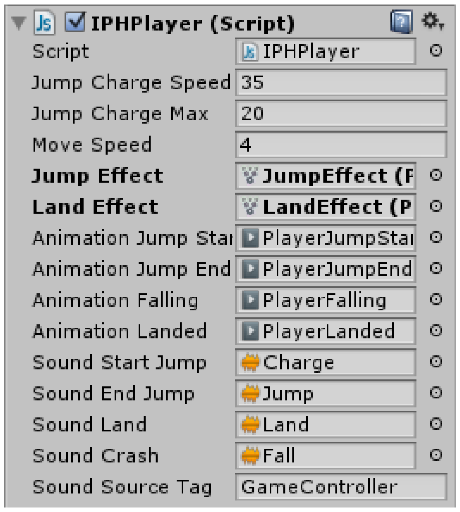
Sound Game Over – The sound that plays on Game Over.

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

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 the Player

The player object has several variables that can change how it behaves, mainly the move speed and jump power.



IPHPlayer (Script)

Script	IPHPlayer
Jump Charge Speed	35
Jump Charge Max	20
Move Speed	4
Jump Effect	**JumpEffect (F**
Land Effect	**LandEffect (P**
Animation Jump Stai	PlayerJumpStai
Animation Jump End	PlayerJumpEnd
Animation Falling	PlayerFalling
Animation Landed	PlayerLanded
Sound Start Jump	Charge
Sound End Jump	Jump
Sound Land	Land
Sound Crash	Fall
Sound Source Tag	GameController

Jump Charge Speed – How fast the jump power of the player increases. The jump power defines the vertical velocity of the player.

Jump Charge Max – The maximum jump power you can achieve by charging. If you get to the maximum jump power you automatically jump.

Move Speed – The horizontal movement speed of the player. This is not affected by jump power.

Jump Effect & Land Effect – These particle effects play when you launch from a column and land on it, respectively.

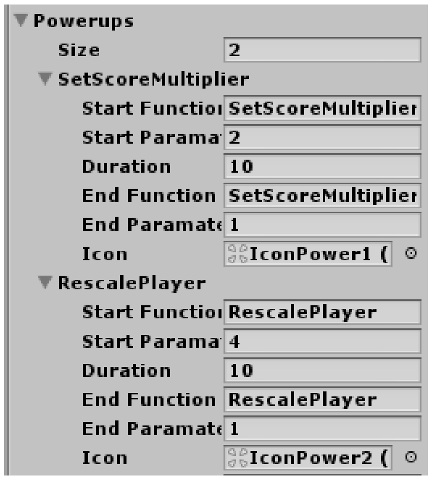
Animations – Various animations for charging up and launching, as well as falling and landing.

Sounds – Various sounds for charging up and launching, as well as falling and landing.

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

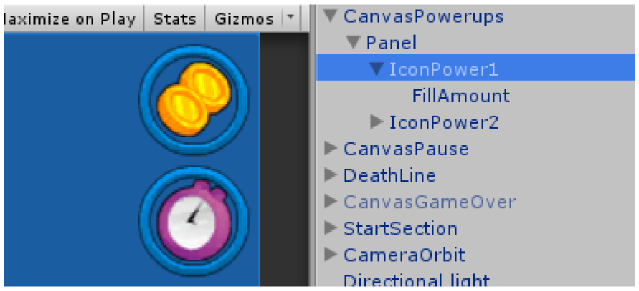
Editing Powerups

A new addition to IPH 1.22 is the ability to add powerups you can pick up in the game. The list of powerups you have is defined in the game controller. When a powerup is activated it runs a function in the gamecontroller, using SendMessag.



▼ Powerups	
Size	2
▼ SetScoreMultiplier	
Start Functio	SetScoreMultiplier
Start Parama	2
Duration	10
End Function	SetScoreMultiplier
End Paramate	1
Icon	IconPower1 (
▼ RescalePlayer	
Start Functio	RescalePlayer
Start Parama	4
Duration	10
End Function	RescalePlayer
End Paramate	1
Icon	IconPower2 (

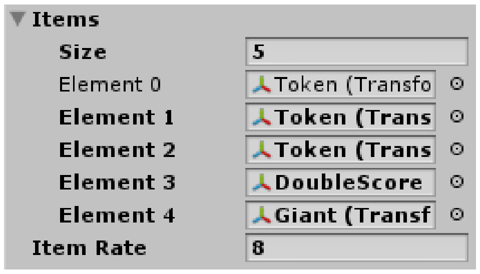
Let's see how the powerups are listed. The first power up doubles the score from collecting coins. It does so by sending a **SetScoreMultiplier** function command with a parameter of 2. Then the game counts 10 seconds before sending a **SetScoreMultiplier** function again with a parameter of 1, making 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 an icon and a fillamount circle which shows the duration of the powerup.

Another powerup we have rescales the player to 4 times its normal size for 10 seconds. This is done by sending a **RescalePlayer** function command with a parameter of 4. 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 scale go back to 100%.

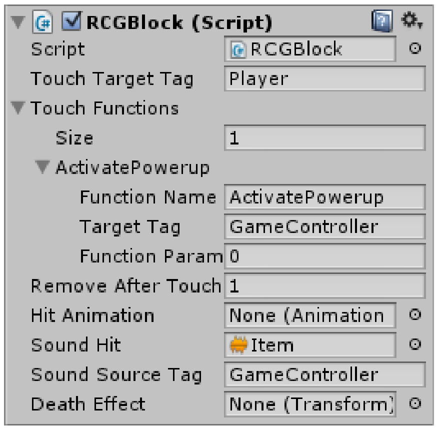
Now that we listed the possible powerups, we need to put items in the game which activate those powers when picked up. We also do this in the game controller.



▼ Items	
Size	5
Element 0	Token (Transfo
Element 1	Token (Trans
Element 2	Token (Trans
Element 3	DoubleScore
Element 4	Giant (Transf
Item Rate	8

The **Items** list holds all the possible items that can show up on a column. The item is chosen randomly from this list so if you want a certain item to appear more often than others, just add it several times to the list. In our example we have the **Toekn** item in the list 3 times which means it's 3 times more likely to appear on a column. Notice we also have the **DoubleScore** and **Giant** items in the list. **Item Rate** is how often any item is created on a column, so **8** mean an item appers every 8th column.

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 **Giant** items to the scene. The component in those prefabs is called **IPHItem**.



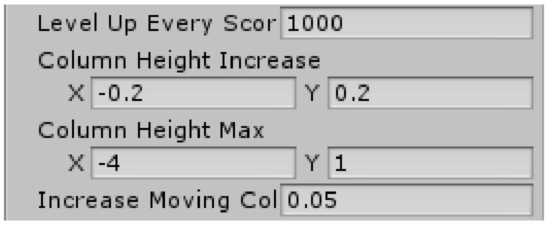
RCGBlock (Script)

Script	RCGBlock
Touch Target Tag	Player
Touch Functions	
Size	1
ActivatePowerup	
Function Name	ActivatePowerup
Target Tag	GameController
Function Param	0
Remove After Touch	1
Hit Animation	None (Animation
Sound Hit	Item
Sound Source Tag	GameController
Death Effect	None (Transform)

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 **Giant** object you'll see that the parameter we sent is 1, which activates the *second* powerup in the list.

Leveling up

A new addition to IPH 1.25 is the leveling up system which increases the column height and the chance of moving columns as you progress. You can change the leveling up properties in the **GameController**.



Level Up Every Scor	1000
Column Height Increase	X -0.2 Y 0.2
Column Height Max	X -4 Y 1
Increase Moving Col	0.05

First you can set the number of points you need to get in order to advance to the next level.

You can also set the increase in minimum (X) and maximum (Y) column height.

Finally you can set the increase in the chance of a moving column to appear. The chance is set between 0 (no chance for moving platform) and 1 (always create a moving platform).

Creating a new character

The following video will show you the process of creating a new character using Flash CS3, and then importing it and setting it up in Unity.

You can use any other graphic software of course. Just open spritesheet1024x1024.png in your preferred software and edit the contents. I prefer Flash because the image quality is independent of resolution (vector graphics), and you can just copy one of the other characters and make your own out of them.

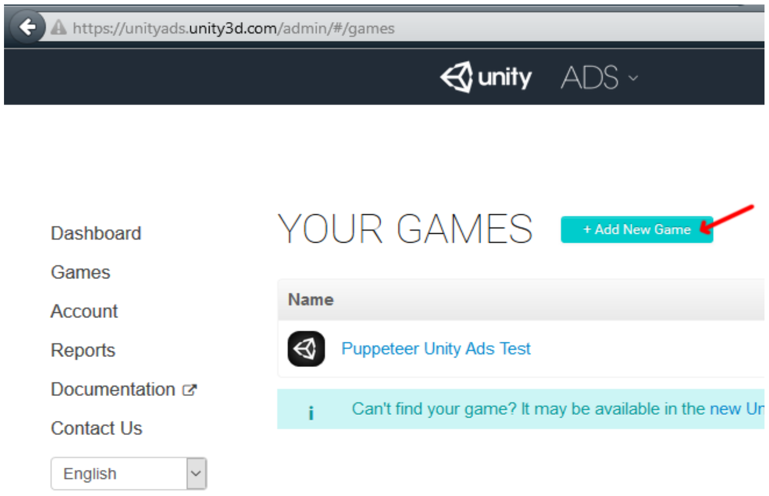
Here is the full video:

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

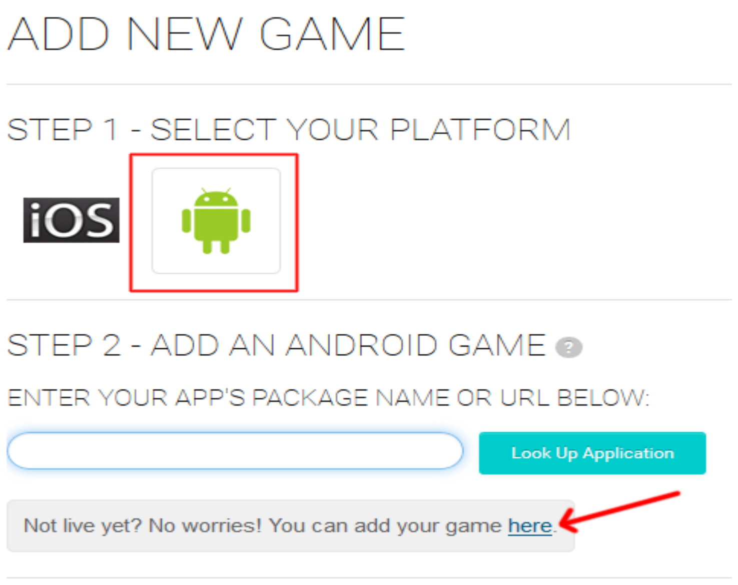
Integrating UnityAds into your project

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

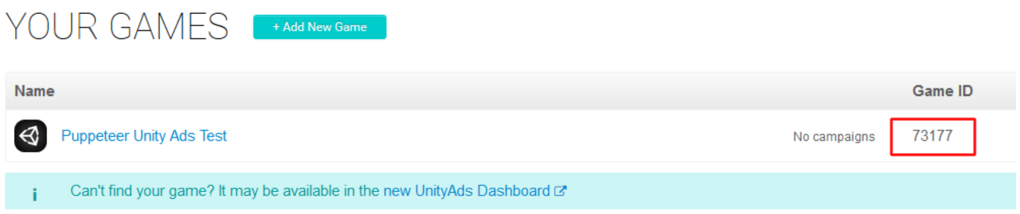
First we need to create our game entry on the UnityAds website. Go to https://unity3d.com/services/ads and create a new game. If you already have your app set and your GameID noted, just skip this part and go straight to importing the UnityAds package into the game.



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.


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.



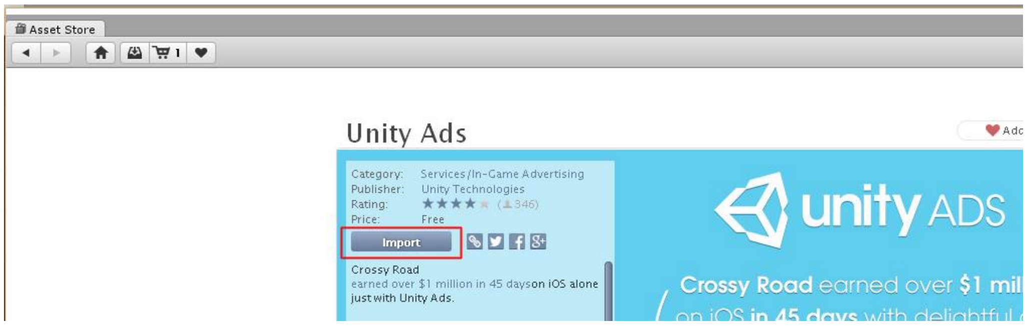
Name		Game ID
Puppeteer Unity Ads Test	No campaigns	73177

i 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.

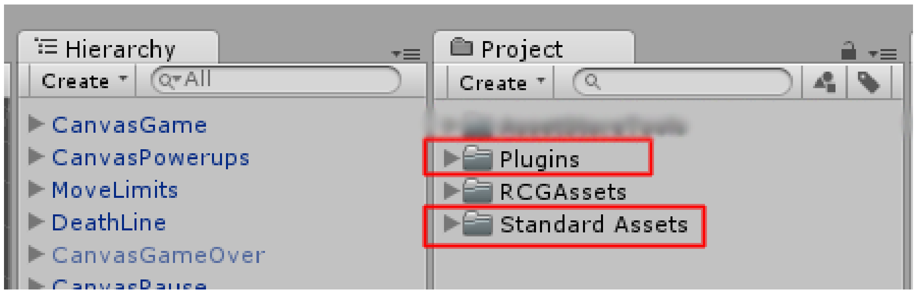
(https://www.assetstore.unity3d.com/en/#!/content/21027)


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

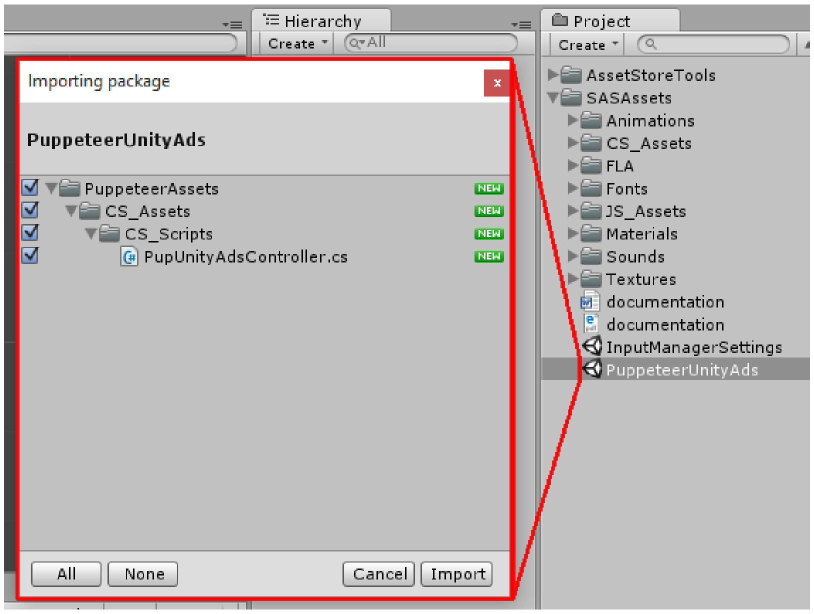
Category: Services/In-Game Advertising
Publisher: Unity Technologies
Rating: ★★★★☆ (346)
Price: Free

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

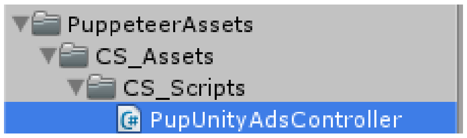
After import you should have two additional folders in your project.

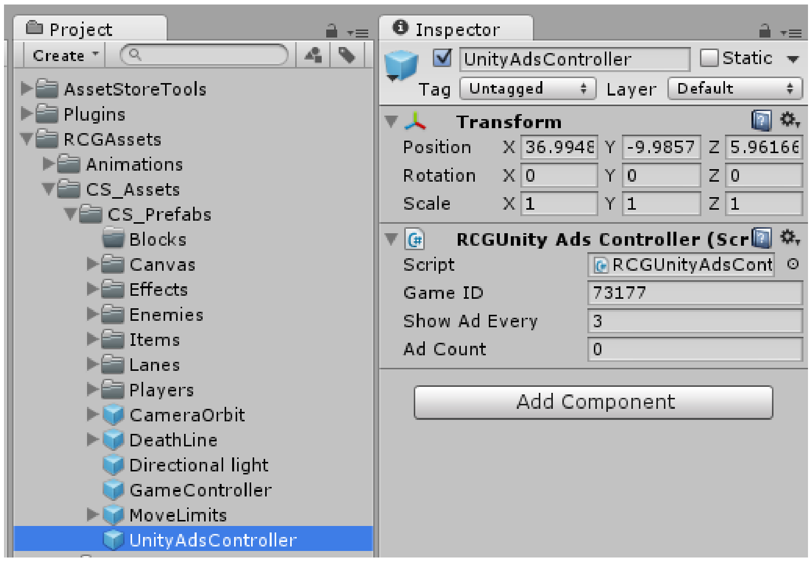


Now we need to bring in the code that integrates the ads into our game. Click on the **PuppeteerUnityAds** package in your project to import it into the game, or choose **Assets > Import Package > Custom Package…** from the top menu and navigate to the **PuppeteerUnityAds** package in your project to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

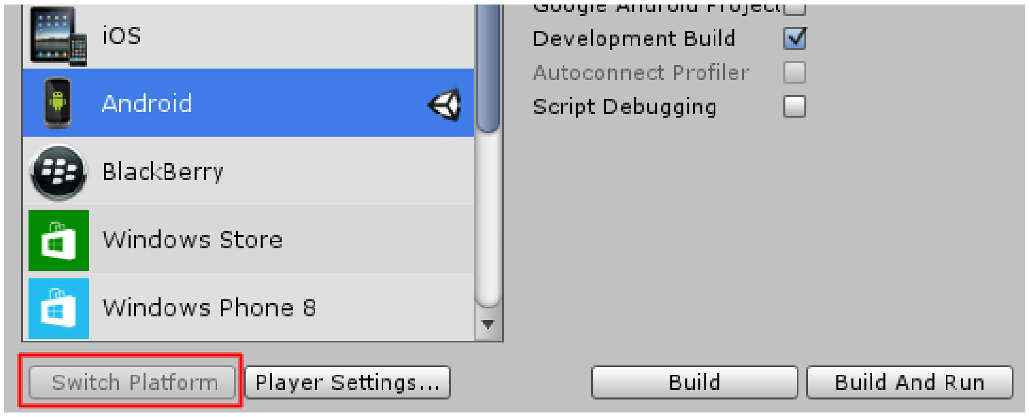




Position X 36.9948 Y -9.9857 Z 5.96166
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

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.

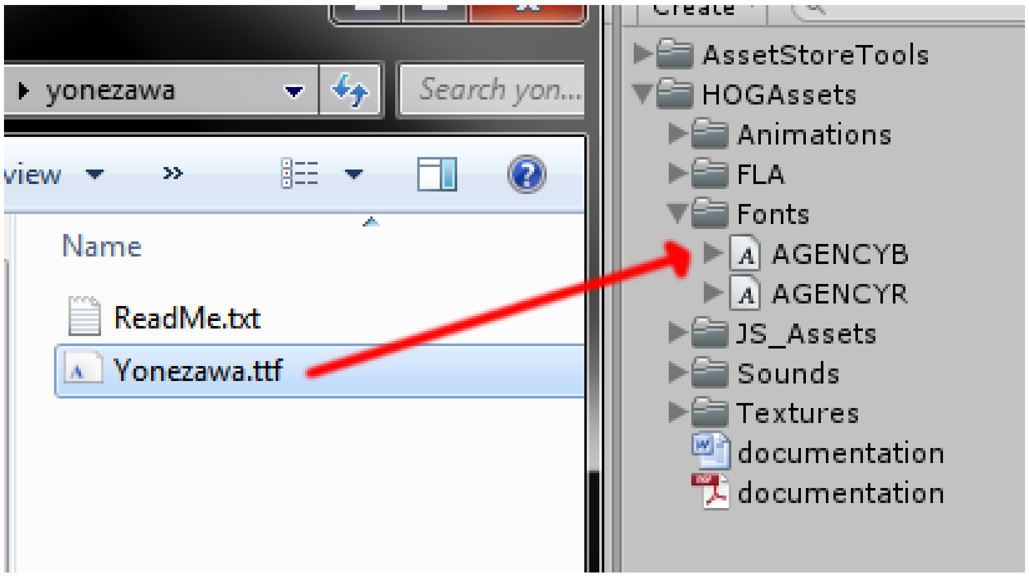
To solve this we must change the texture compression format for iOS. Follow these steps:

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

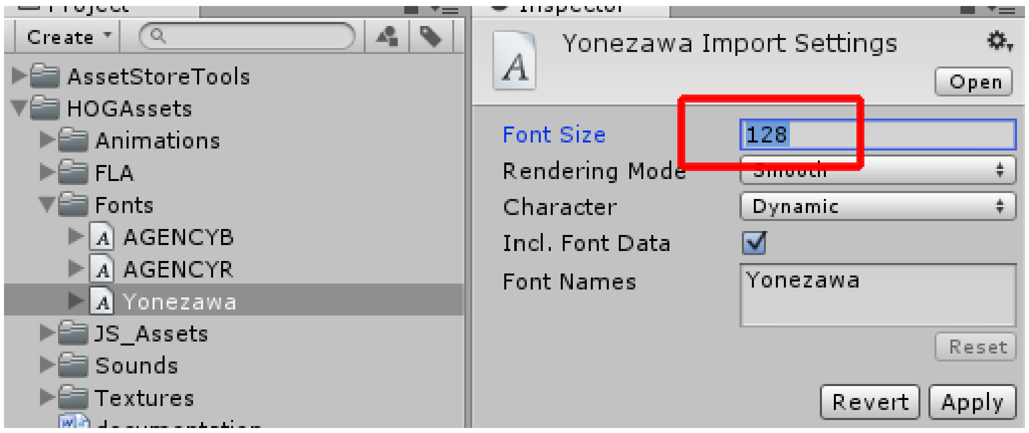
How to change font in the game?

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.



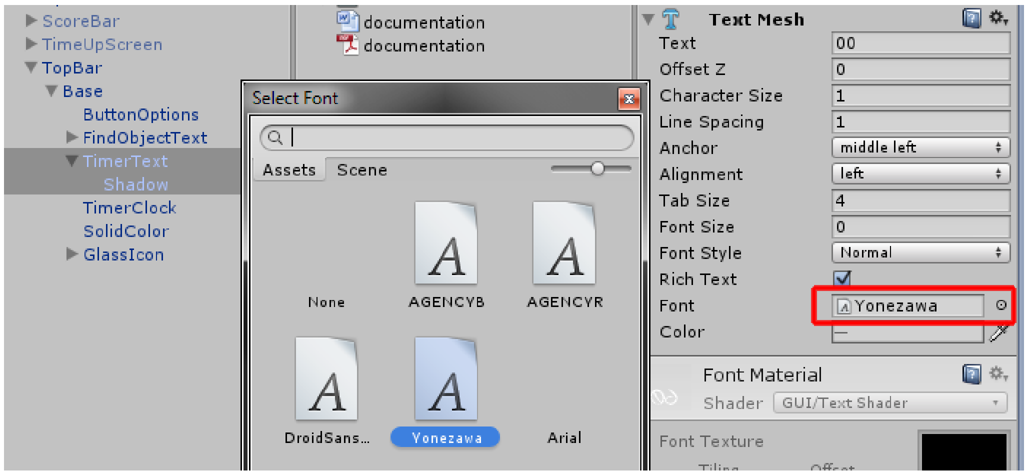
Click on the font you added and edit its attributes. I personally set all my fonts to a high number (and then scale the text object down) so that they look crisper in-game.



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

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!