

Infinite Platform Hopper Template

Game documentation and HowTo guide.



This document contains:

Package Description and features	2
Try the webplayer.....	2
Update history	2
Credits.....	3
Overview of the game's library contents	4
Customization Guide.....	5
Getting started.....	5
The Game Controller	5
Editing the Player.....	7
Editing Powerups.....	8
Leveling up.....	9
Creating a new character.....	10
UnityAds Integration	11
Frequently Asked Questions	13
Does this package work on mobile?	13
My sprites are not showing on iOS	13
How to change font in the game?	13
More games by Puppeteer	15

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?

Hold the jump button down to charge up your power, then release to jump! Try to land as close as possible to the center of the column to get top bonus!

Try the webplayer

Current version 1.3

Update history

1.3 (03.08.2016)

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

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.

Customization Guide

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.



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 Distance	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 Canvas	CanvasGameOver
Main Menu Level Name	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.

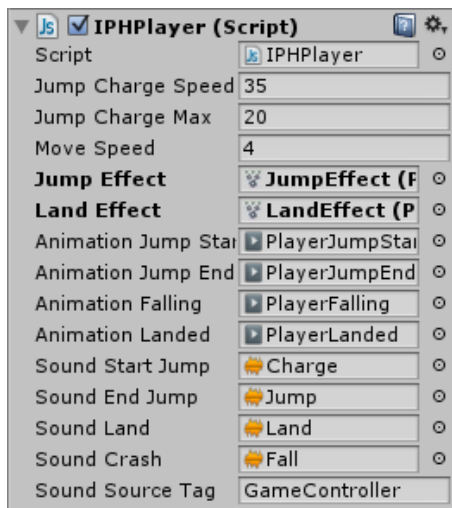
Sound Game Over – The sound that plays on Game Over.

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.



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.

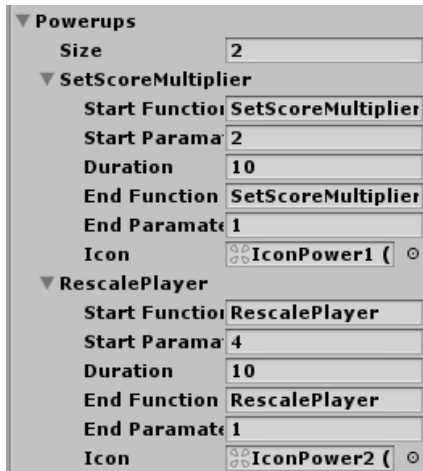
Animations – Various animations for charging up and launching, as well as falling and landing.

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

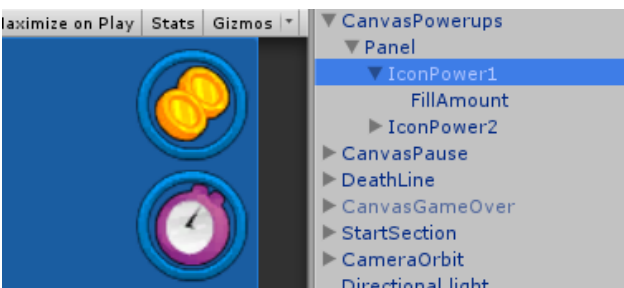
Sound Source Tag – The audio source from which sounds play.

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 `SendMessage`.



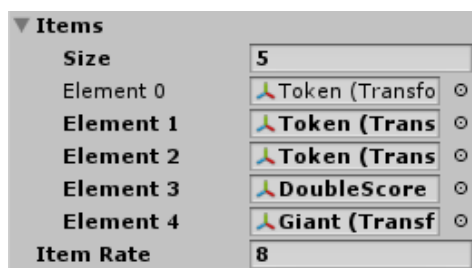
Let's see how the powerups are listed. The first power up doubles the score from collecting coins. It does so by sending a **SetScoreMultiplier** function command with a parameter of 2. Then the game counts 10 seconds before sending a **SetScoreMultiplier** function again with a parameter of 1, making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

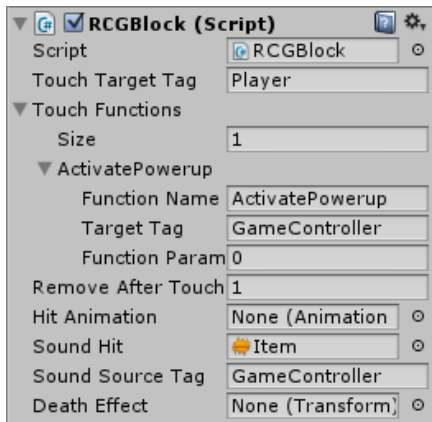
Now that we listed the possible powerups, we need to put items in the game which activate those powers when picked up. We also do this in the game controller.



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**.



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**.



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>

UnityAds Integration

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:
<http://www.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>

Frequently Asked Questions

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.

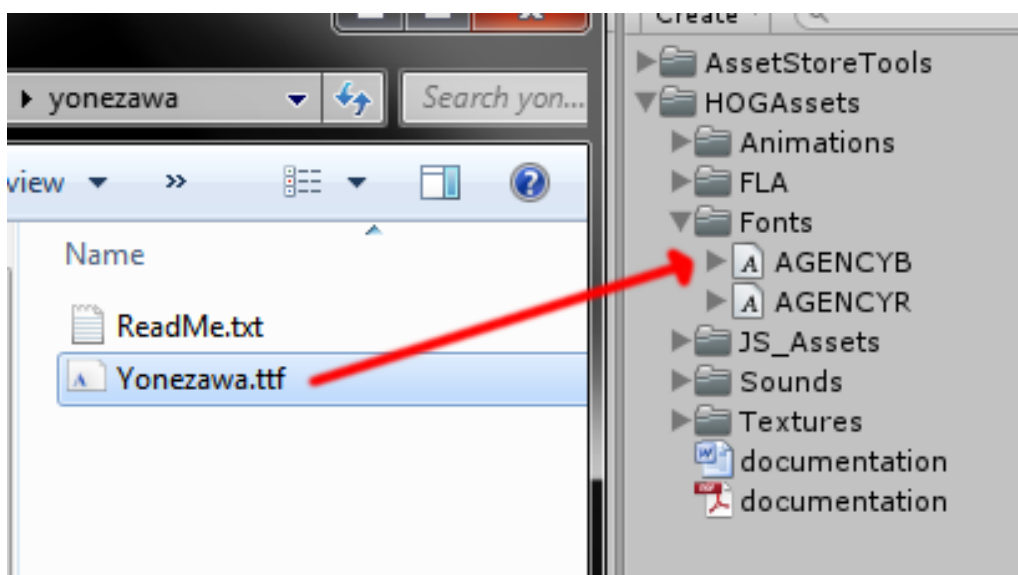
To solve this we must change the texture compression format for iOS. Follow these steps:

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.

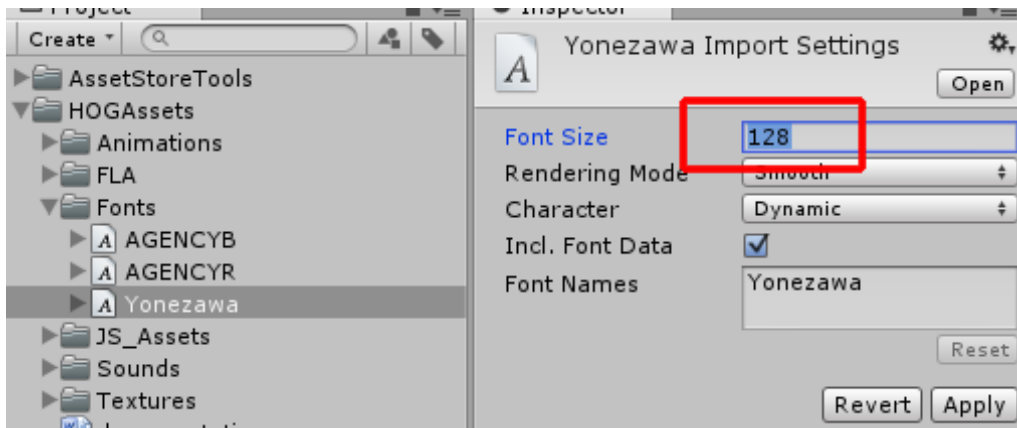
How to change font in the game?

To change a font in the game do the following:

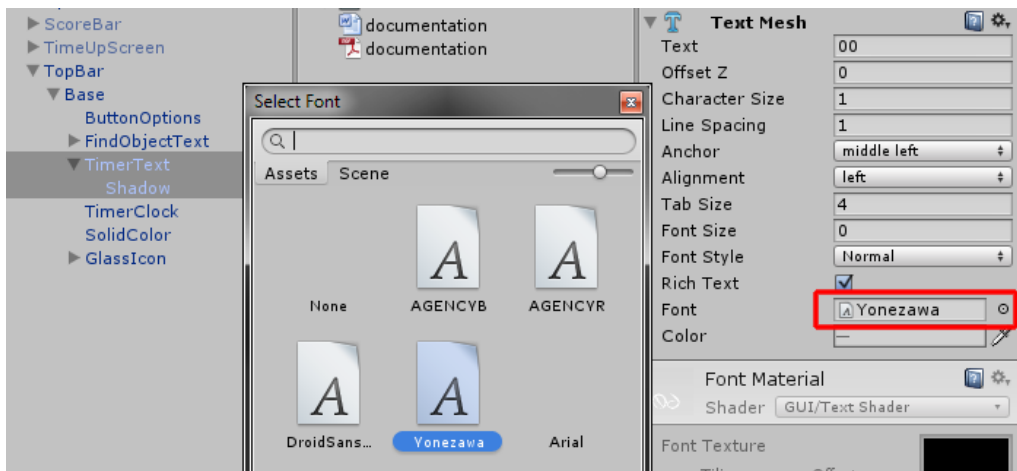
Find a font you like and drag the .ttf file over to the Fonts folder in your game.



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.

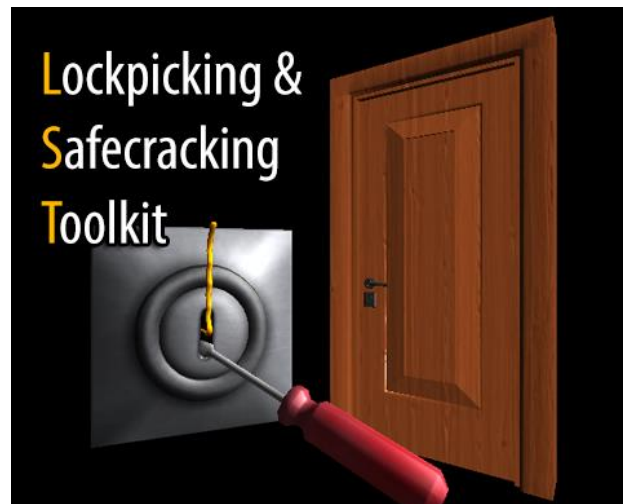


Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and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