Zigzag Infinite Runner Game

Game documentation and HowTo guide.

This document contains:

Package Description and features

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

How to Play?

Press SPACE or CLICK to turn the car. Collect cash and don't crash! The game controls also fit mobile and console without need for further coding.

Current version 2.0

Update history

2.0 (10.09.2017)
- New upgraded version modernized for the latest versions of Unity 5.5, 5.6, and 2017.
- Updated visuals using Unity's demo assets for the background, camera effects and updated lighting.
- Updated UI and animated menus and in-game messages, and shop menu.
- Day/Night cycle for lighting across multiple play-throughs.

1.24 (28.06.2016)
- Support for Unity 5.3 and higher versions.
- Better support for UnityAds 5.2 and above.

1.22 (04.10.2015)
- Added support for UnityAds along with a quick integration guide.
- Added a victory condition, victory screen, player victory animation, and sound.

1.18 (12.08.2015)
- Improved performance for both C# and JS.
- Minor changes.

1.17 (01.08.2015)
- Added 5 new vans: Mystery, Orange, Hotdog, Flamer, and Turtle.
- Money Text suffix ($ sign) now appears correctly in the shop.
- Updated packages to 4.6.7 and 5.1.2

1.11 (05.05.2015)
- Reorganized C# and JS assets in their respective folders
- Video tutorial: Adding a new car to the game

1.1 (02.05.2015)
- Project converted to C#. You now have both JS and C# in the same project.
- Added script to remove objects after a distance.
- Minor graphical change

1.0 (23.04.2015)
- Initial version

Credits

The sounds are courtesy of the free sound project.

Some demo models are replicated from the Unity official Tanks demo

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 ZIRAssets folder using Unity3D 5.5.0f3 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
- **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

Zigzag Infinite Runner Game (ZIR) 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 ZIR 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 roads, animates the background, and also makes the camera chase the player. The Game Controller is also used to increase the difficulty of the game after collecting a set amount of cash.

Camera Object – The camera follows the player at all times.

Player Objects – This is a list of all the player objects in the game. **Current Player** is the index number of the playable character.

Turn Button – This is the button that turns the player when using a keyboard or gamepad.

Ground Object – This is the background object behind the road. This object is always attached to the player while its texture is animated to give an illusion of motion. Don't change its size.

Ground Details – This is a list of the objects that are created around the road to fill the ground with details (cactus, rock, etc.) **Details Offset** is the random distance from the edge of the road where these objects are created. 0 means they are created right at the edge of the road, and the larger the number the farther they are created from the road edge.

Next Road Section – This is the next road object in the scene. Each road that is created continues from the spot of the next road section.

Default Road Size – This is the default size (in Unity3D units) of the road objects. All road objects must have the same default size.

Road Straight – This is the basic building block for a straight road. Any length of road is made up of several straight road objects.

Road Turn – This is the object that is created on a road bend.

Road Section Length – This is the length range of a straight road section.

Precreate Roads – How many road sections to create at the start of the game. A single straight road section may be as long as you set it in the **Road Section Length** value.

Item Drops – This is a list of the objects that will be created along the width of the road such as cash. **Drop Offset** is the distance range along the width of a road section that these objects will be created at. 0 creates the item at the center of the road, while a higher number moves the item towards the edge.

Score/Text – This is your score in a single match, and the text object that displays it. The **Suffix** is the text that is displayed after the score value.

Money Player Prefs – This record keeps track of all the scores you ever collected in all matches, and uses them as currency to buy things in the shop.

Game Speed – The overall speed of the game (Time.timeScale)

Level Up Every Score – How much score do we need to reach before leveling up. Leveling up increases the **Game Speed** by the amount of **Level Up Speed Increase.**

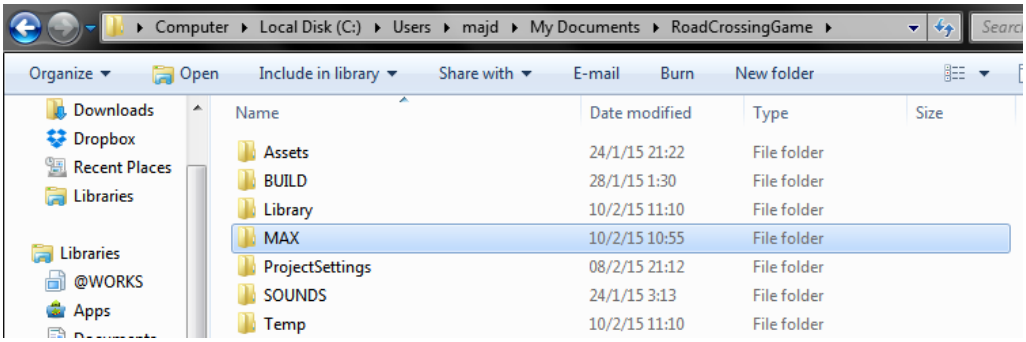
Level Up Messages – These are the texts bits that are displayed in the text bubble near the player when it levels up.

Main Menu Level Name – This is the name of the level that is loaded when we go back to the menu.

Confirm/Pause Buttons – These buttons come into action when the game ends. Pressing the confirm button (default: SPACE) will restart the level, and pressing the pause button (default: Esc) will return to the main menu.

Changing Graphics

Graphics in Zigzag Infinite Runner Game are 3D. They are modelled in 3ds Max and then exported to FBX to be used in Unity. I included the MAX files in the package; you can find it in ExtraAssets.rar. The reason for having it inside an archive and not in a simple folder in the project is because unity has trouble dealing directly with MAX files, so when you have any MAX models in unity the software will hang up for a while in order to accommodate the new objects. So my recommendation is to keep these files just outside of your project folder, like this:



The textures are made using Flash CS3, and then exported to PNG. I do this because Flash allows me to control the quality of the textures easily since it is vector based graphics.

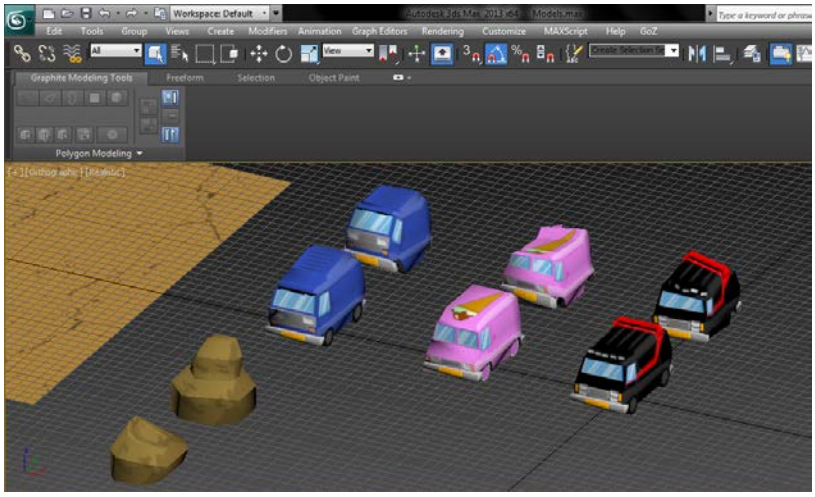
The various animations are made in unity itself and are assigned to generically named parts of the model, so that you can easily switch models while maintaining the animation on the object.

Reskinning 3D Models

Here's how you can put your 3D models instead of the original ones, using the "reskin" method. Using this method you can simply "reskin" your model with another model without having to worry about creating new prefabs and assigning meshes.

- The package includes a MAX folder that contains all the models in the game. You can open this in 3DS Max 2014 or above. Open the file Models.max. This contains all the models in the game.

- Now you can edit each model individually, and check how it scales relative to other objects. Once you are done, you can export the whole file as FBX to be used in unity.



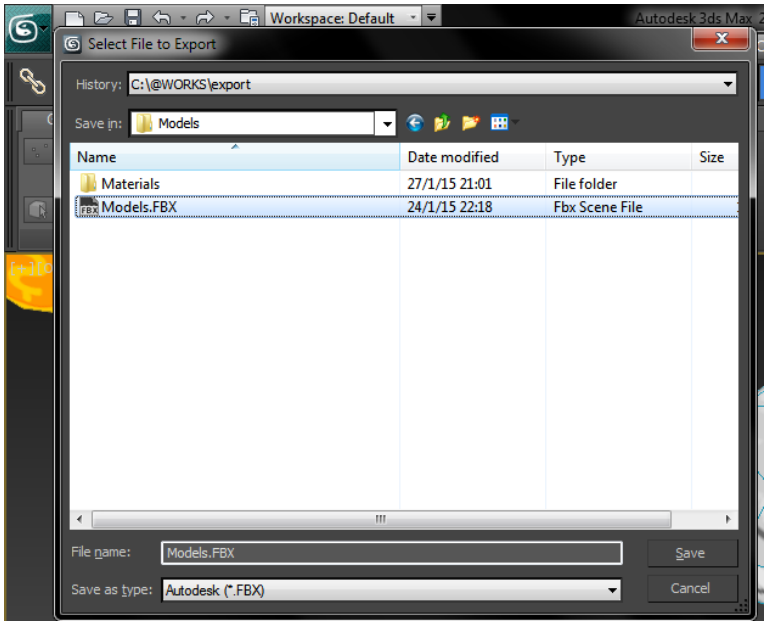
- For our specific example I will replace the blue van model with a black van model. First I shift+drag the black van object to create another copy of it.

- Then I select the black van model and **Attach** to it the blue van model. Click the blue van, click attach, and then click the black van. Now the black van is part of the blue van model.

- Now switch to element mode, select and move the black van to stand right over the position of the blue van.

- Next we select the blue van element and delete it. Now we are left only with the black van.

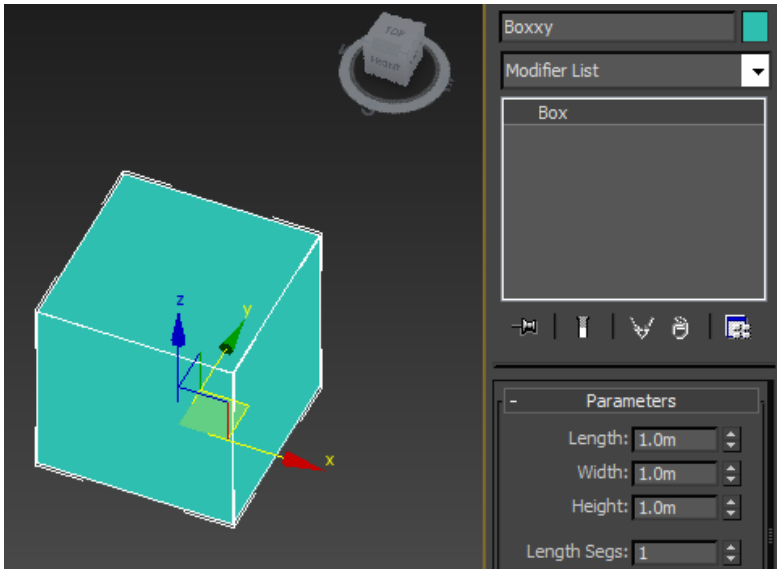
- To complete our work we will export our models into FBX for use in Unity.
- Click **Export** from the menu, go to the folder **Models** inside your game project and overwrite the file **Models.FBX**



You'll notice the "black van" player has the same attributes of the "blue van" as before because we only replace the mesh and nothing else. This is the most straightforward method for reskins, but you can also of course create new models and use them in the game without having to replace a current model.

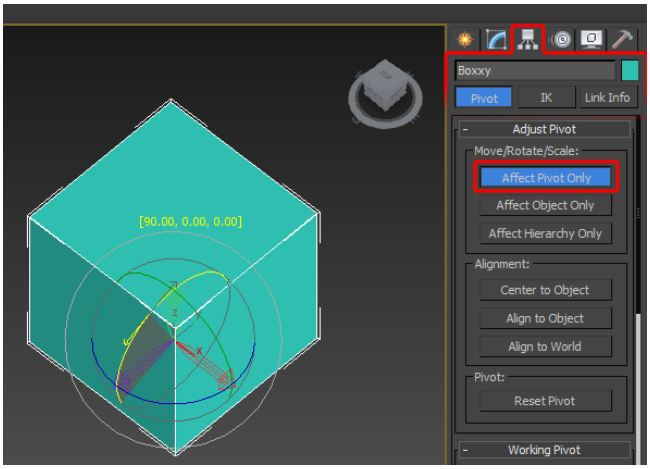
Creating a new model

To create an entirely new model, just model it in 3ds max inside **Models.MAX**, give it a unique name and then export to **Models.FBX** again.

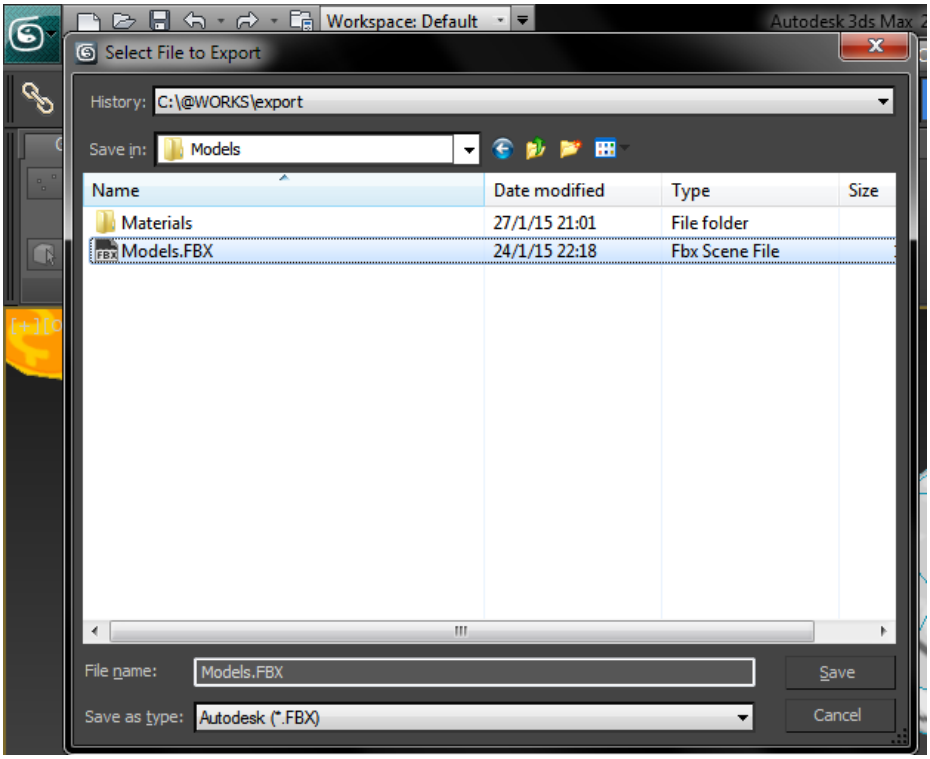


- Note: When creating a model in 3ds max for unity, make sure the Z (blue) axis is looking at the front and the Y (green) axis is looking up. This will align the model properly in unity.

- To do this, select your model and go to the Hierarchy pane, then choose Affect Pivot Only, and rotate the axis 90 degress so it faces the proper direction.



- Click **Export** from the menu, go to the folder **Models** inside your game project and overwrite the file **Models.FBX**



- Now when you go to Unity and check the file **Models.FBX** inside **Models** folder, you'll find a new model named **Boxxy** which you can use in your game!

Video tutorial: Adding a new car

This 10 minute video shows the process of adding a new car to the game, making it playable, and then adding it to the shop. For the purpose of the tutorial we used a car model from the package Cartoon Race Cars 12 but you can use any FBX/OBJ car model you like. We also used Flash CS3 to edit the icon of the car in the shop but you can also use any graphic editing software you like.

Watch the video on youtube:

https://www.youtube.com/watch?v=X2S-_0yop40

What we did:

1. Import the CRC12 package into your Zigzag project, and open the game scene.

2. Duplicate one of the vans in the scene, and then drag one of the car prefabs from the CRC folder to the van hierarchy.

3. Align position and rotation of the car to the van.

4. Drag each part of the car model to the corresponding part in the van prefab (tireBackL to WheelBL, carBody to part1, etc).

5. Align the new wheels to their parents, and move the parents to fit well within the new car body.

6. Remove the Mesh Filter and Mesh Renderer from the old model parts.

7. Duplicate one of the crash effects, and assign the car parts similar to how you did in steps 5 & 6.

8. Drag the new car and the new crash effect to the Project folders to create prefabs which can be used later.

9. Assign the new crash effect to the new car in the component.

10. Increase the number of players in the Game Controller, and assign the new car as the 4th player.

11. In the Start Menu scene, open the shop object and duplicate one of the buy buttons.

12. Increase the number of items in the Shop, and assign the new button to it, set the cost of the car, and set the PlayerPrefs name of hte car (this is the record that keeps track if a car was purchased or not).

13. Now we create an icon for the shop by taking a screenshot of the car in the game scene, cropping it, and then placing it in a nice icon frame in Flash.

14. After exporting the PNG to unity, and slicing it in the Sprite Editor, we can add it to the shop icon.

That's it!

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

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.

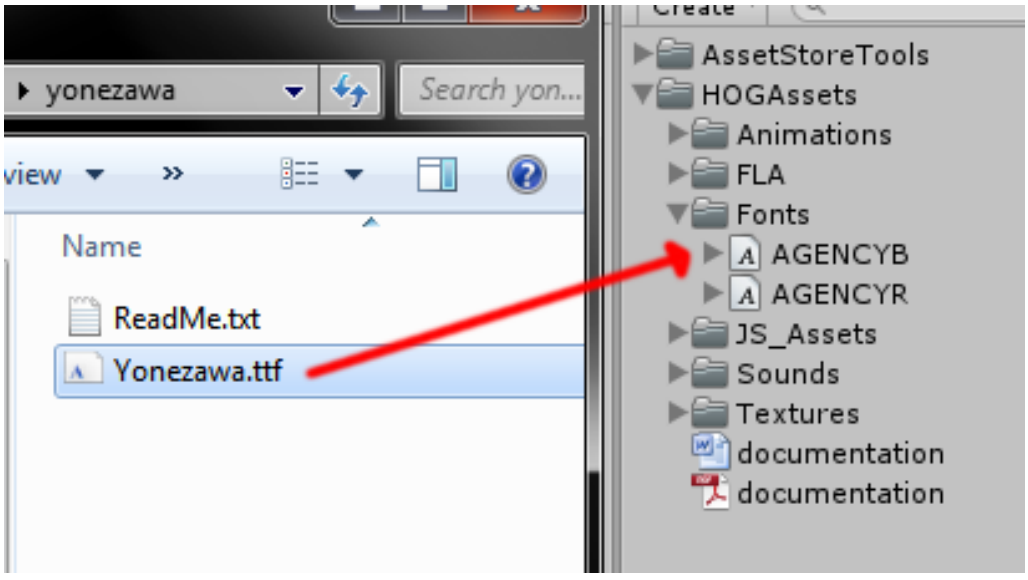
To solve this we must change the texture compression format for iOS. Follow these steps:

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.

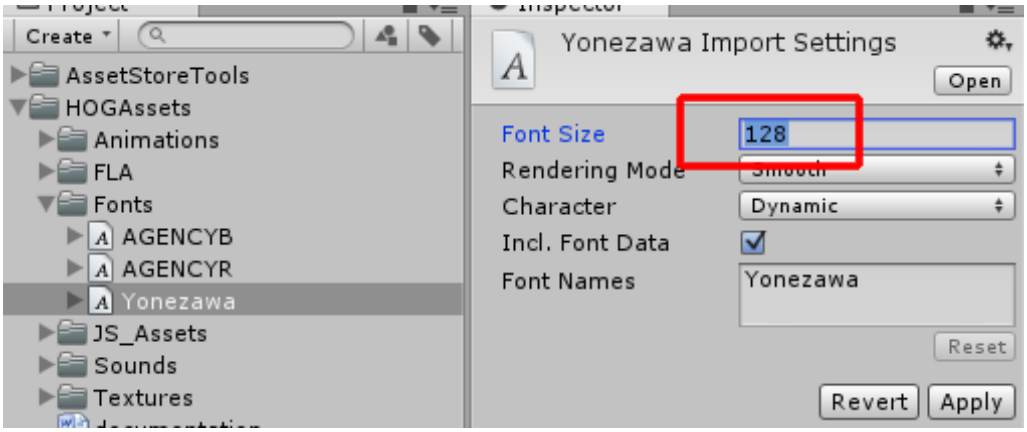
How to change font in the game?

To change a font in the game do the following:

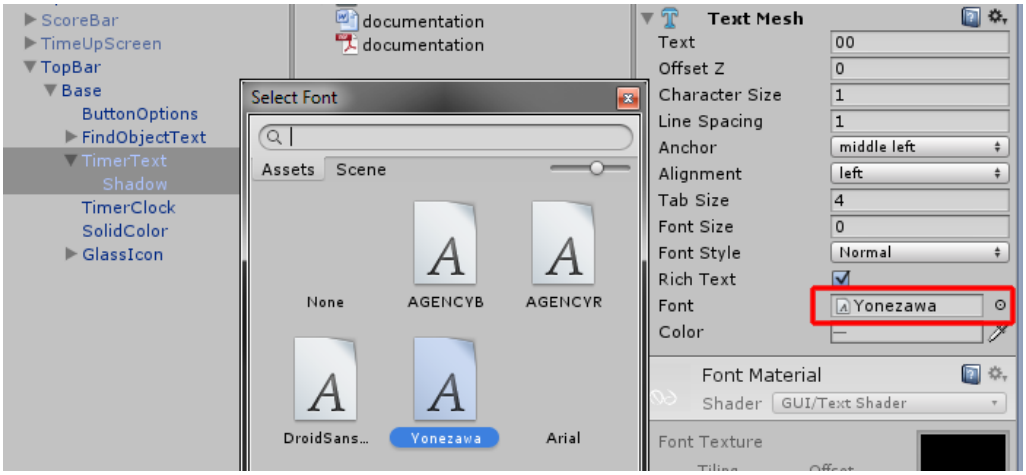
Find a font you like and drag the .ttf file over to the Fonts folder in your game.



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.

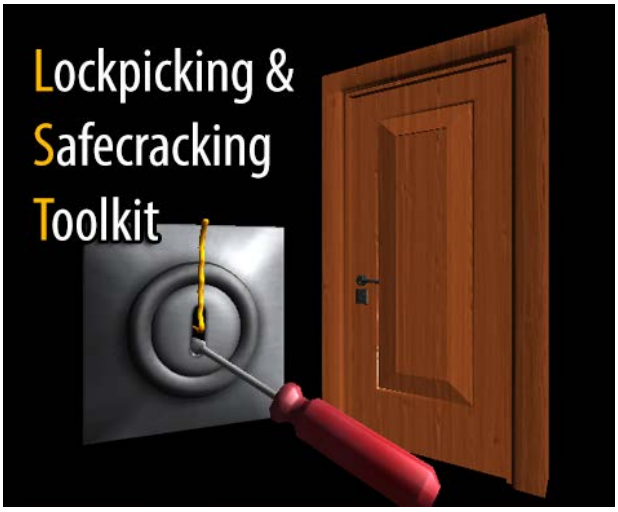


Select any text object in the game and change its font to the new font you have. Sometimes the text might disappear, but it's normal. Just write something in the text box above and it will refresh. Also, make sure you change the text for the shadow; you can select both the main text and its shadow and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