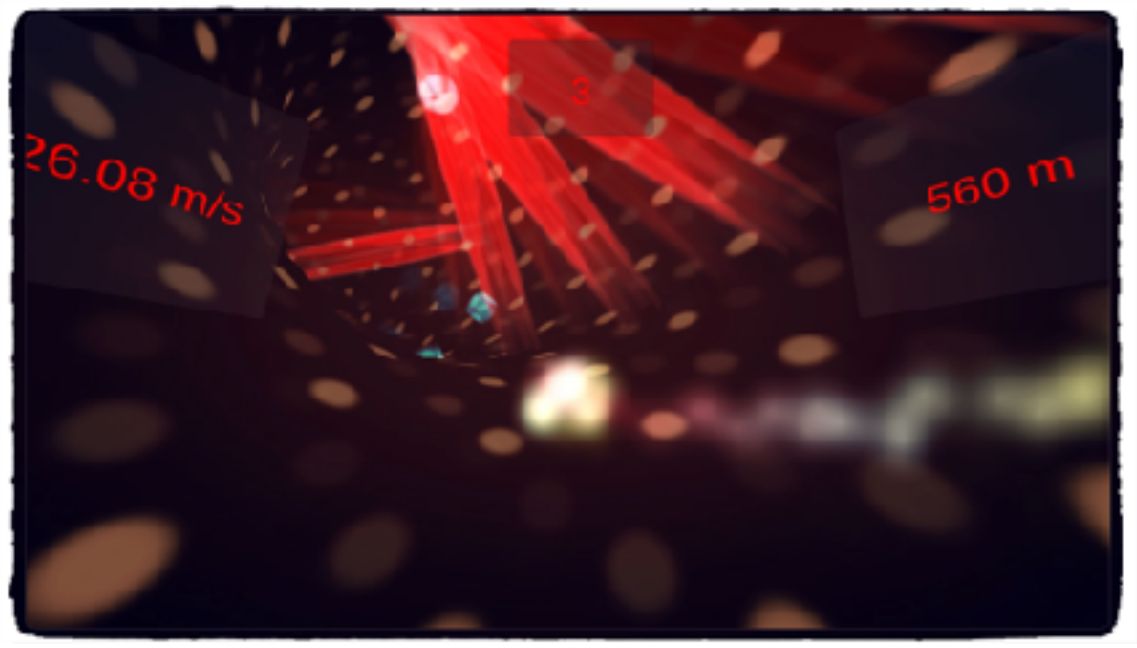


Into The Hole Documentation

Mamadou Cisse

Thank you for your purchase!

Table Of Contents

IMPORTANT

This package works with the plugin Infinity Engine written with C# 7 syntax, this means that you must active this option in Unity Editor. If you don't do this, Unity will not compile your project. Go to this link to enable C# 7 syntax.

How to use Infinity Engine?

Infinity Engine is a set of plugins that extends Unity. Each plugin is located at :

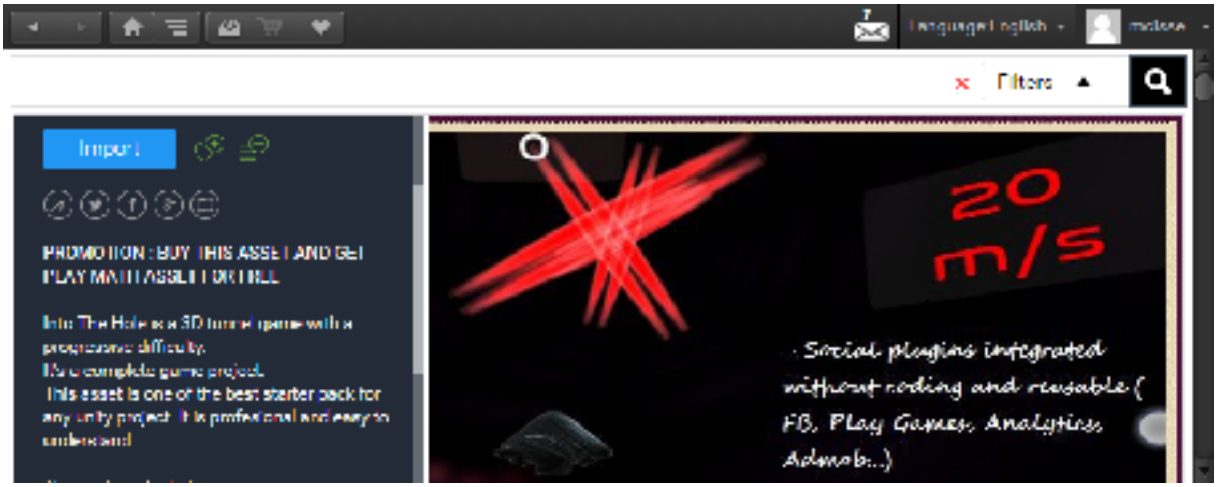
"Assets/InfinityEngine/Frameworks/Name of the feature"

You can find a documentation for each of these plugins here

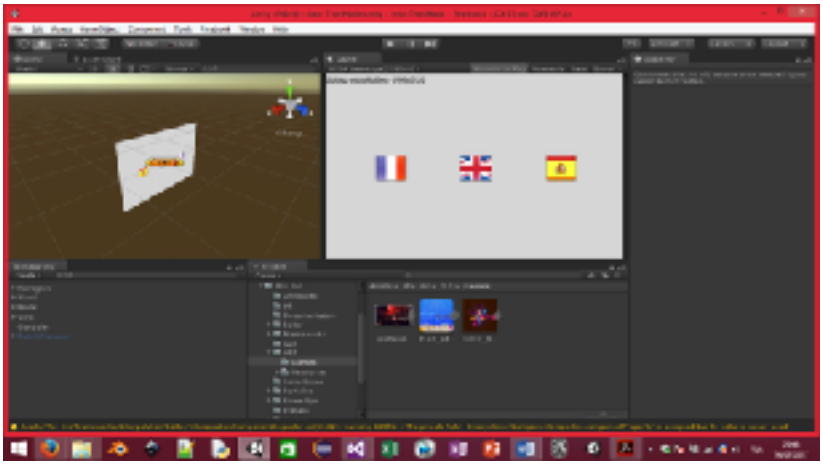
How to test this package?

- **On Desktop :**

Open Unity editor and import the package that you downloaded on Unity Asset Store.



Open the scene “Into The Hole” placed at “Assets/Into The Hole/Scenes/”.

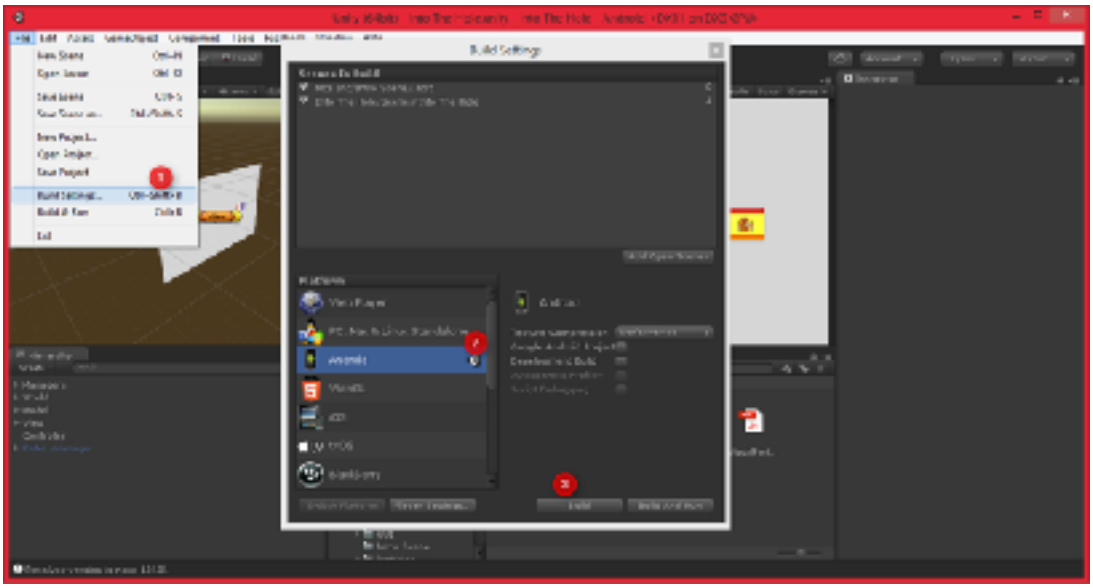


Click on Play button to start the game. Click on left arrow to move to the left and the right arrow to move to the right.

- **<u>On Android Device :</u>**

Go to the tab “File/Build Settings…” and switch the current build platform to Android.

Click on the button build to generate the apk file of the game.



What's in the Package?

InfinityEngine.dll: The assembly that contains the main code of Infinity Engine (You can get the code of the assembly by sending an email at mciissee@gmail.com).

InfinityEngine Folder: Placed at the root of the project, this folder contains all plugins of **Infinity Engine**.

Into The Hole Folder: Placed at the root of the project, this folder contains main code of Into The Hole game.

Pipe.cs: Script which allows to generate a procedural pipe and obstacles.

PipeManager.cs: Script which manages pipe instances.

PipeItemGenerator.cs: Base class of all scripts which allows to place obstacles into the pipes (SpiralPlacer.cs, CirclePlacer.cs, RandomPlacer.cs).

BaseItem.cs: Base class of all Obstacles.

BaseAvatar.cs: Base class of all player avatars.

Player.cs: "Model" component of MVC design pattern, this script contains the data of the game like the score, the distance covered...

View.cs: "View" component of MVC design pattern, this script contains a reference to UI objects in the scene.

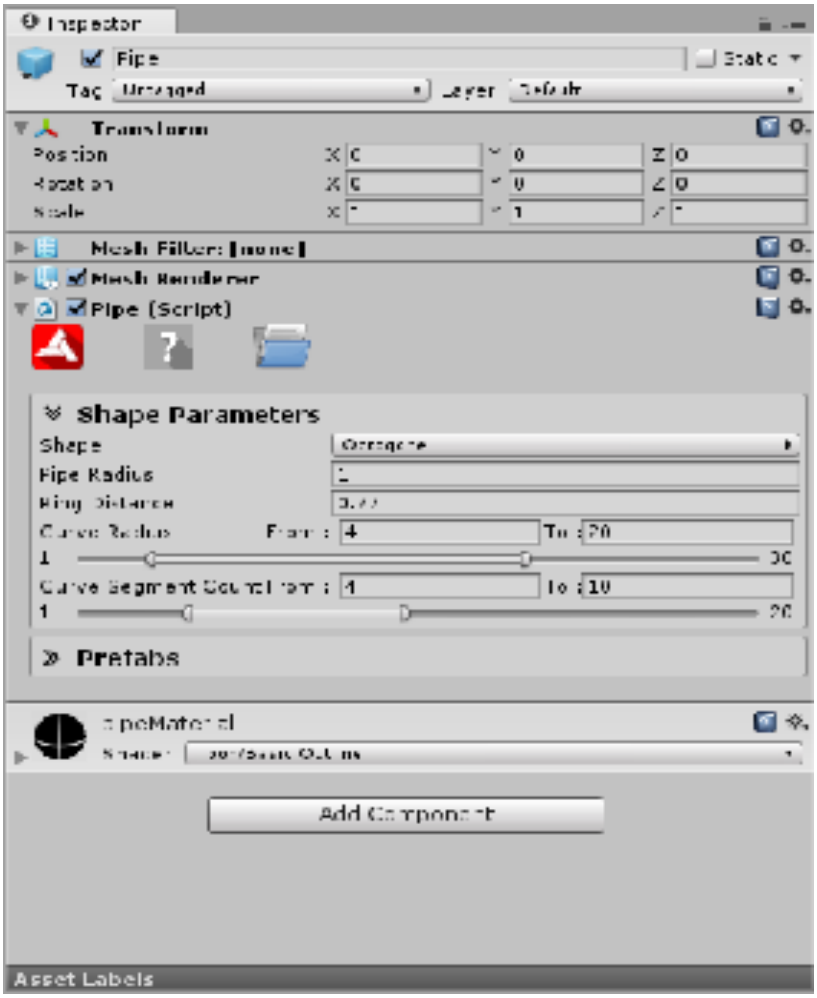
Controler.cs: “Controller” component of MVC design pattern, this script manages the game events (game starts, game ends) and control the model and the view. When the player click on the button play the event GameStartedEvent of this script is invoked and when the player loses, the event GameEndedEvent is invoked. If you want to do an action when the game starts or ends, you must to add a listener to the event by calling the function Controler.RegisterEvent(GameEvent evt, Action callback); inside the function OnEnable of your MonoBehaviour class. Conversely, if you want to remove an action, call the function Controler.UnRegisterEvent(GameEvent evt, Action callback); inside the function OnDisable of your MonoBehaviour class.

Getting Started

A - Pipes

In this game, the pipe object is represented by the script “Pipe.cs”.

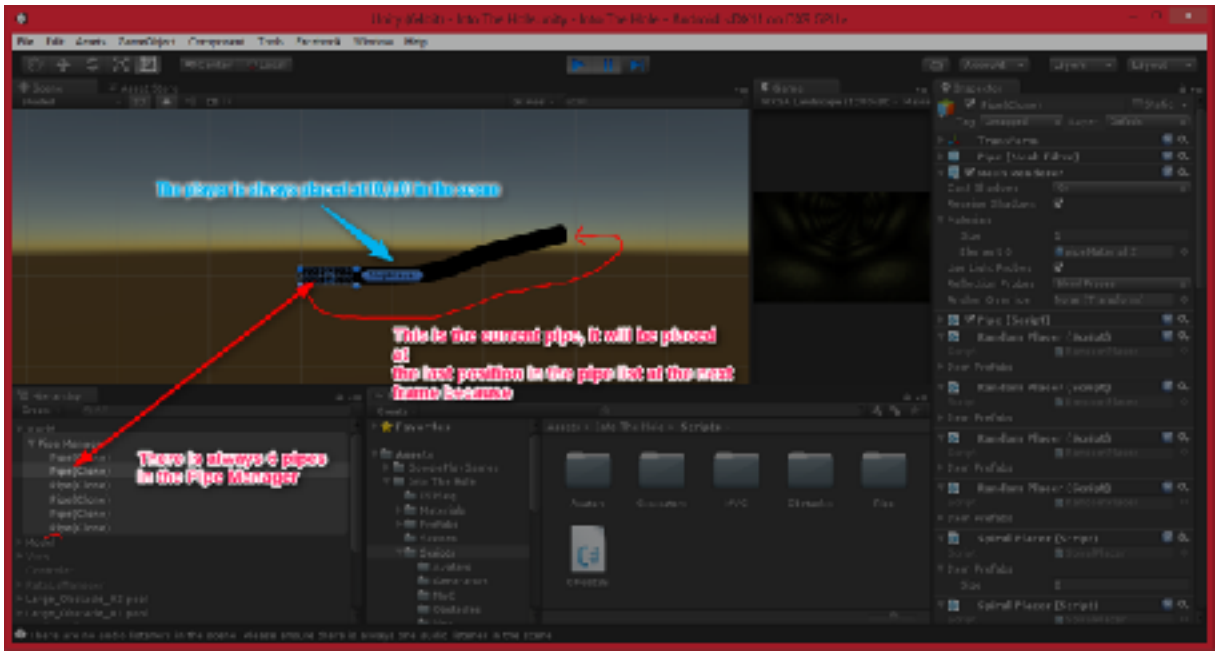
This script allows to generate a procedural pipe. If you want to change the pipe without coding, you can modify the prefab placed at “Assets/Into The Hole/Prefabs/”.



The pipes generated procedurally at the beginning of the game are controlled by the script PipeManager.cs, placed at “Assets/Into The Hole/Pipe/”.

To have infinite game, at the beginning we have 6 pipes. The player is always in position Vector3.zero and he never moves, he just turns around the point (0, 0, 0). Every time the pipe [1] in the list “pipes” of PipeManager.cs script is the current pipe. The current pipe aligns itself with the player every time and when it overtakes the player, the list will be shifted and the

current pipe finds itself in position 6.
For example with the list [1,2,3,4,5,6], we obtain the list [2,3,4,5,6,1].



B – Obstacles

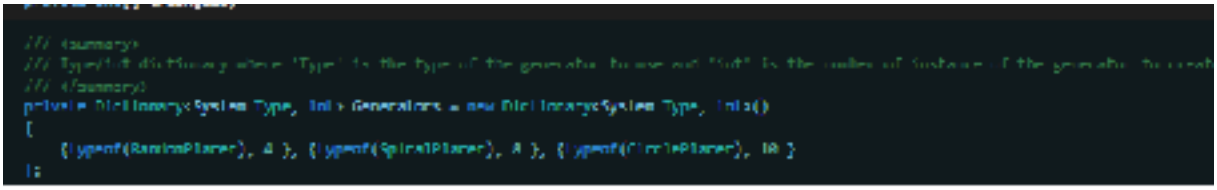
An obstacle is composed of 3 Game Objects.

- A root Game Object which must has a component which inherits from “BaseItem.cs”
- An empty Game Object named “Rotater” and child of the root Game Object.
- A Game Object child of “Rotater” that contains a collider component and have the tag “Item”. This is the Game Object that can enter on collision with the player.

The obstacles are generated by the scripts "RandomPlacer.cs", "SpiralPlacer.cs" and

“CirclePlacer.cs” which inherit from "PipeItemGenerator".

You can easily create new one manners to generate obstacles by inheriting the class “PipeItemGenerator “. If you have created new generator, you must to register it to the Dictionary “Generators” of the class “Pipe”.



How to customize this package?

A - Create New Obstacle

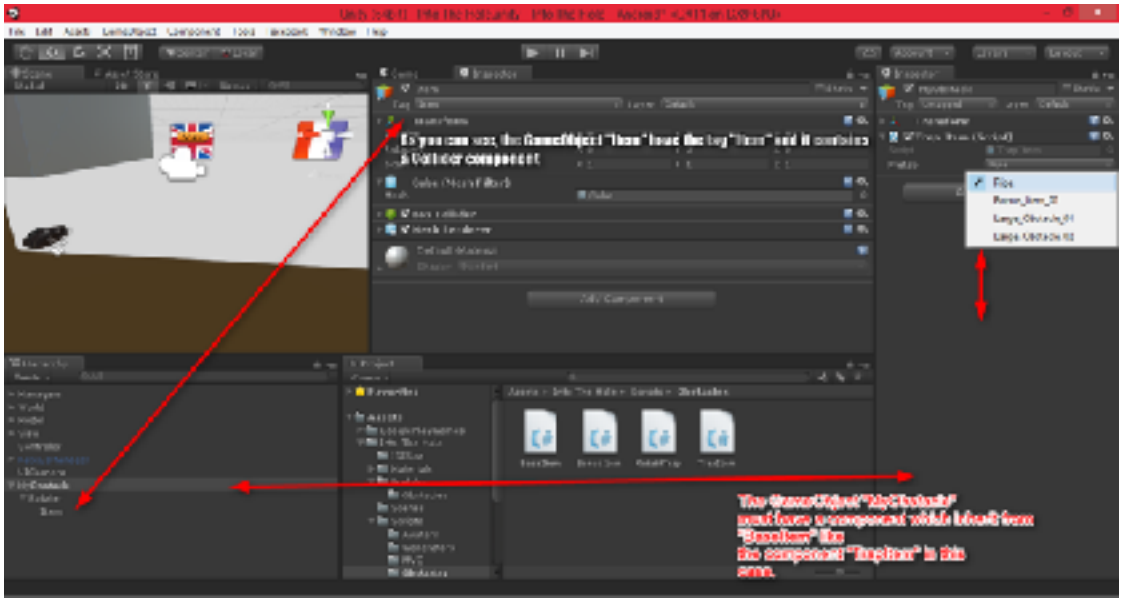
- Create new Game Object and name it like you want (example “ MyObstacle ”).
- Add new child Game Object to the Game Object that you created and name it “Rotater”
- Add new child Game Object to the Game Object “ Rotater”.

This one must have the tag “ Item ” and contains a “Collider component ”.

- Drag and drop your Game Object to the folder “ Assets/Into The Hole/Prefabs/Obstacles ” to create a prefab of your obstacle.
- Press the key “ Alt + U ” or go to the tab “ Tools/ InfinityEngine./Resources Database/Editor ” and click on the button “Update” to update ISI Resource plugin database.

- If your obstacle is a trap, add the component “ TrapItem ” to the Game Object “ MyObstacle ”, if your obstacle is a bonus, add the component “Bonus Item”.

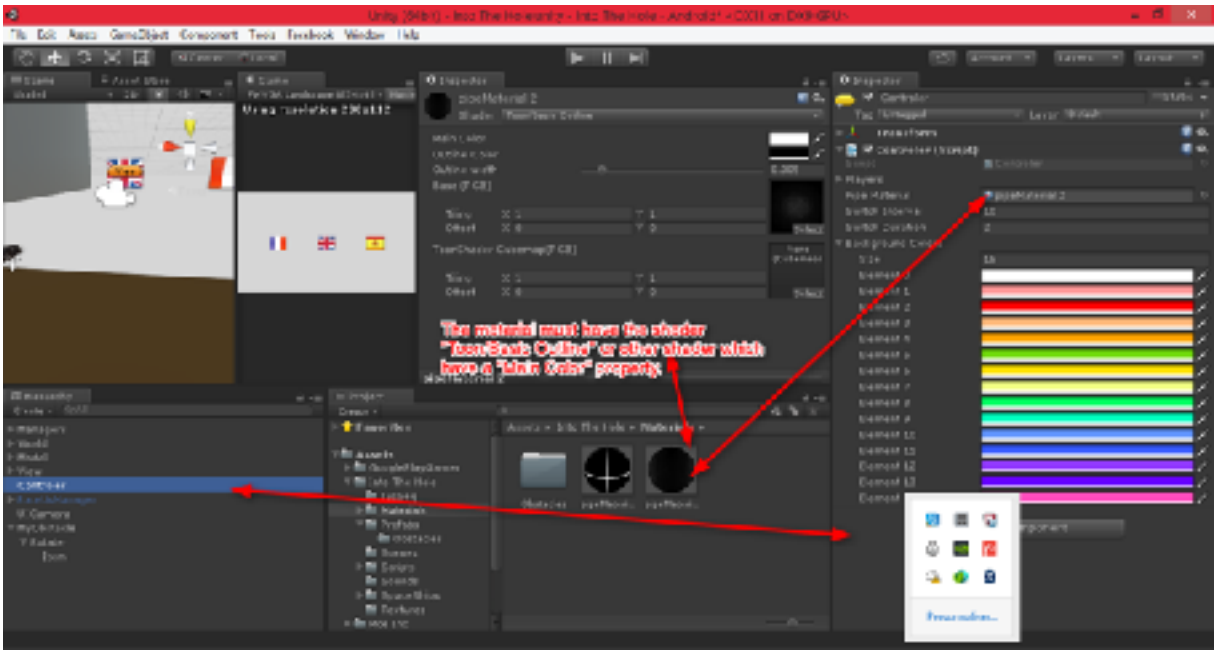
If you want to add a custom behavior to your obstacle, create new script which inherits from “ BaseItem ”.



B - Change The Texture Of The Pipes

If you want to change the texture of the pipes, create new material which has the shader “ Toon/Lit ” or another shader which has a color property because the color of the material is changed at runtime.

After you created your material, add drag and drop it into the field “ Pipe Material ” of the Game Object “ Controler ”



C - Change The Player

By default, there is two playable avatars in this package:

- ShipAvatar
- ParticleAvatar

The avatar switch system is implemented in “ Controler ” class but it is not used in the final game. The only thing you need to do if you want to change the player is register your player to the Controller class in unity inspector and call the function “ ChangePlayer () ” of the class “ Controler ”.

FAQ

How The Menu System Works?

The menu system of the project is controlled by the plugin ISI Interpolation. This plugin provides a powerful tool to creates an inspector driven user interfaces without programming any line

of code. The entire user interface of the game is created thanks to this plugin.

How To Use The Power Of InfinityEngine?

InfinityEngine is very useful tool for unity there is a lot of tools shared with the API, the API is fully documented and easy to use. InfinityEngine is shared with this project as a dll library because there are lots of scripts, using a dll makes it easier to use the API with other projects. Never delete any files in the InfinityEngine folder (unless you know what you are doing). If you want to get the source code of the API, you can send an email at mciissee@gmail.com.

Final Words

Thanks for your purchase, please rate this asset if you like it and consult my asset store page to see more assets.

If you have any question, feel free contact me at mciissee@gmail.com

Good luck for your projects.