# Adventure/Horror Complete Kit
# Doc. Ver. 1.4

First of all thank you so much for buying our kit I really hope to help you develop your game. If you have any question you can contact me here footprint.sftw@gmail.com, remember to send me your invoice in order to allow me to check your purchase.

# 1 - Scene Setting

FPH kit include a *BaseScene* which will allow you to start building your game with no particular effort but even if we created this scene for you, there are a couple of things you must do BEFORE to start playing with our kit or open any of you scenes. These are some layers and tags which needs to be added, without them both UI and interaction won't work.

TAG (these should be added together with the assets but since I'm not sure of it....) :

- *InventoryObject*
- *ObserveObject*
- *InteractObject*
- *ChangeLevelObject*
- *DialogObject*
- *ShowTextObject*
- *BatteryObject*
- *DoorObject*

LAYER (in this order):

- Layer8: *GUILayer*
- Layer9: *HandLayer*
- Layer10: *PuzzleLayer*
- Layer11: *PlayerLayer*

## 2 - Unity5.x Importing

If you are using Unity5.x you will probably note some ugly and strange white lines around some object. This is caused by old legacy shader.
You will only have to switch from the old legacy shader to the new standard shader shipped with Unity5 and everything will be perfect.

Unity5 changed the way LockCursor works, if you are using Unity5 please open "FPH_ControlManager.cs" and from line 84 to line 103 you will find that

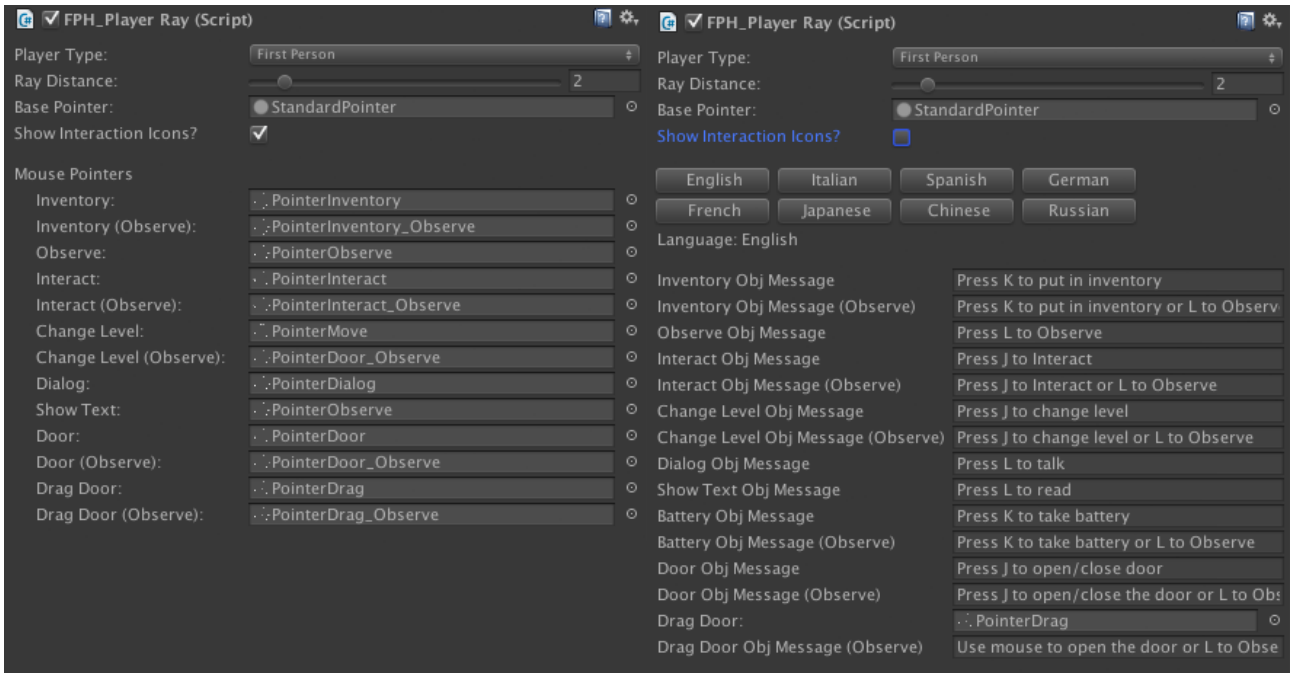
```
if(isScreenLocked){
        // Comment the next line if using Unity5
        Screen.lockCursor = true;
        // Uncomment these lines if using Unity5
        /*
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
        */
}
else{
        // Comment the next line if using Unity5
        Screen.lockCursor = false;
        // Uncomment these lines if using Unity5
        /*
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
        */
}
```

If you need some support while doing that do not hesitate to contact us we will help you doing that.

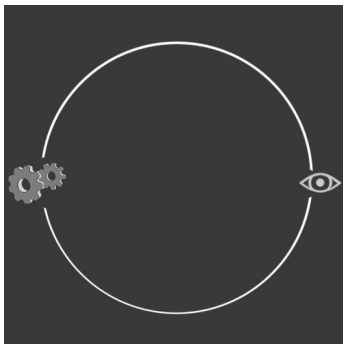
**NOTE:** Since version 1.35 we decided to include two packages: one for Unity4.x and one for Unity5.x. The AssetStore will automatically recognize which version of Unity you are using and will download the right version for you, this mean that you won't need to manually upgrade the project anymore.

# 3 - How to play

As everything in our kit we designed the gameplay to be clean and easy.
Interaction can be displayed in two different way: with InteractionIcons and with a simple information text.

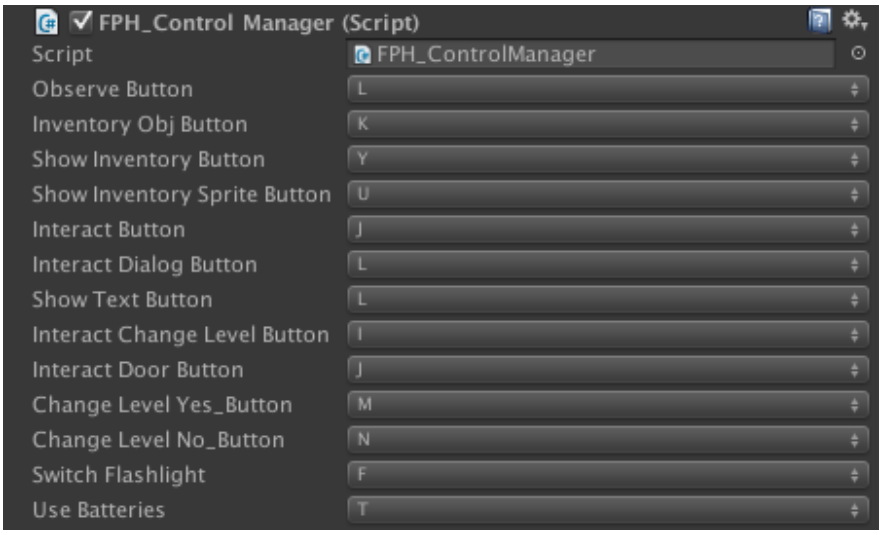



We decided to use four buttons*: **I**, **J**, **K**, **L**. Every button corresponds to an icon position in the InteractionIcons. This mean that if this icon is shown in the game



you will have to press **J** to interact and **L** to observe.

If you want to create you own icon or you simply want to change the various interaction buttons you will only have to select *"-GameManager"* object and change the various var of *"FPH_ControlManager"*



# 4 - Languages

Our kit is thought to be as multi-language as possible. It can (without modification) support English, Spanish, Italian, German, French, Japanese, Chinese, Russian.
Adding an unsupported language is really easy, only a couple of line code are needed.

As first thing you will need to open "*FPH_LanguageManager"* then add language name to "*LanguagesEnum"*. The next thing will be to add this line inside of the "*Awake()"* function of "*FPH_LanguageManager"*

```
else if(Application.systemLanguage == SystemLanguage."LANGUAGE TO SUPPORT"){
    gameLanguage = LanguagesEnum."LANGUAGE TO SUPPORT";
}
```

Now you must add this line inside of any script which will display a text

```
if(FPH_LanguageManager.gameLanguage == FPH_LanguageManager.LanguagesEnum."LANGUAGE TO
SUPPORT"){
    textObject.text = "Localized text";
}
```

If you want to debug a language select *"-GameManager"* object and toggle *"isDebugging"* var of "*FPH_LanguageManager"* now select the language you want to debug.

## 5 - Player setup

You don't really need to setup a player since we have created a prefab for you but in case you want to know how things are working or you want to use a First Person Character Controller from another assets there are some things you need to know.

FPH toggle character/mouse movement using *"CanBeController"* static bool var. This mean that if you have a script which control character (movement/jump or whatever you want) or mouse movement you should place the script logic inside of an if statement

```
if(FPH_ControlManager.canBeControlled){
    // Mouse control or code that should work only when
    // The player can be controlled
}
```

You must assign *FPH_PlayerRay.cs* to player camera in order to interact with objects.

If you want to show a hand create a new Camera (we will call this camera HandCamera for convenience) and make it child of player camera, assign this new HandCamera (and every child of this camera) to *HandLayer*. *Depth* should be lower than GUICamera but higher than player camera. Set *ClearFlags* to *Depth only*. Set *Culling mask* to *HandLayer*, HandCamera should only render HandLayer's object.

Our FPController is just a modified version of the one shipped by unity with their SampleAssets, we will update it in the future but for now Unity's solution is good enough.

## 5.1 - UFPS Integration

UFPS integration will only need of a couple of small tweaks.
The first thing you must do is to drag the *"Camera&Controller"* prefabs from *"UFPS → Base → Content → Prefabs → Player"*.

Let's take a look at *"-GameManager"* object and focus on *"FPH_BatteryManager"*. There are two vars you should be aware of: *"HandObj"* and *"Flashlight_LightObj"*.

The first one store the animated hand object reference to trigger animations (like flashlight switch etc.) if you want to use your own animation system or you just don't want to animate your hands just toggle *"Animate Hand?"* bool var.

The second one store the flashlight light and the flashlight cone (which should be a children of flashlight light) in order to toggle light when the player has no battery or when you switch off the flashlight.

Let's now focus on *"Camera&Controller"* prefabs, duplicate the *"HandCamera"* from FPH Player prefab and make it children of *"FPSCamera"* and reset *"HandCamera"* rotation and position.
You can now delete FPH Player prefab, since you deleted Player prefabs you will have to set some reference again (for example player camera reference in puzzles).

The next step will be to assign *"FPH_PlayerRay"* to *"FPSCamera"*.

Now that all of Player reference has been restored you will only have to change a couple of script.
Open *"vp_FPController"* and *"vp_FPInput"*.
Inside of *"vp_FPController"* modify both *Update()* and *FixedUpdate()* function placing all the logic inside of this if statement

```
if(FPH_ControlManager.canBeControlled){
    // Mouse control or code that should work only when
    // The player can be controlled
}
```

Do the same for *Update()* function inside of *"vp_FPInput"*.

Now rename *"Camera&Controller"* to *"Player"*.

Everything should work now!

**NOTE:** In case you imported UFPS after FPH you may need to reset the layers.

# 6 - Touch and button control

FPH Kit include a very simple and basic TouchManager (can recognize Down, Up and Swipe). If you want to setup a camera to send Touch or Mouse message you only have to select a Camera gameObject and go to *"Tools → FPH Menu → Setup TouchMouse Camera"*.

You must add a script with one of these function if you want an object (which must have a collider) to recognize mouse or touch event.

```
public void OnTouchUp(){}
public void OnTouchDown(){}
public void OnTouchDrag(){}

public void OnCustomMouseUp(){}
public void OnCustomMouseDown(){}
public void OnCustomMouseDrag(){}

public void OnTouchSwipeRight(){}
public void OnTouchSwipeLeft(){}
public void OnTouchSwipeUp(){}
public void OnTouchSwipeDown(){}

public void OnMouseSwipeUp(){}
public void OnMouseSwipeLeft(){}
public void OnMouseSwipeDown(){}
public void OnMouseSwipeRight(){}
```

For swipe message things are a bit different. You must set

```
FPH_TouchManager.swipeDetectObj
```

to the object which should receive swipe event. Remember to reset the var to null when you have done.

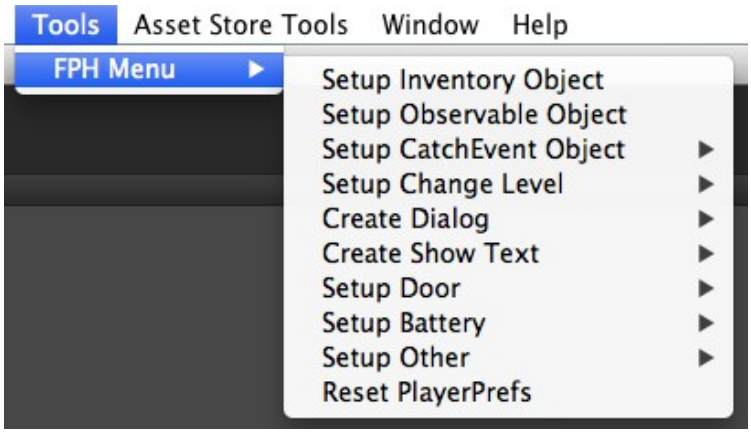
Button control variables are stored inside of *FPH_ControlManager* which is a component of -*GameManager*.

## 7 - Interactive object

We created several object that you can interact with.

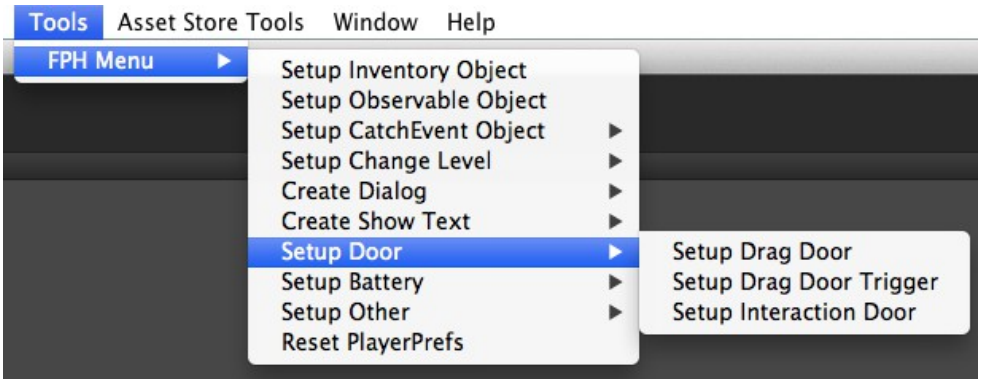
- Door
- Dialog
- Change Level
- Inventory
- Show Textfile
- Observe
- Battery *(go to -GameManager if you want to change drain rate)*
- Numpad Puzzle
- Rotating Puzzle
- Togglable
- 2D Jumpscare
- ToggleObject on Event
- Surveillance Camera System

Configuring one of these object is as easy as pressing a single button you will only have to select the object you want to setup and open the *FPH Menu*, if no object has been selected a new empty one will be created.



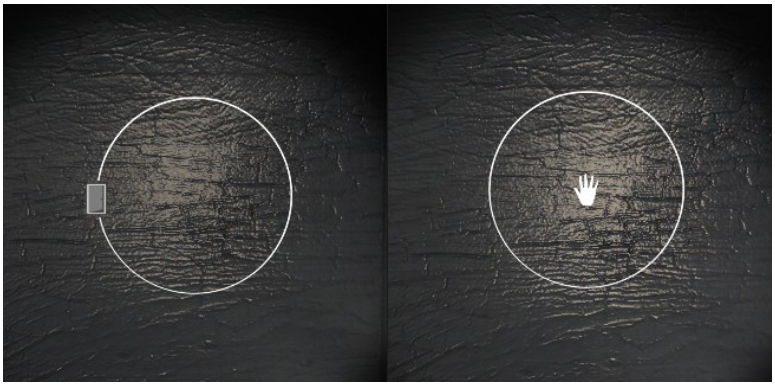

They all work out of the box and we created a lot of sample and setup in order to allow you to do everything without coding but in case you want to create your own object the inventory need some explanation.

# 8 – Door



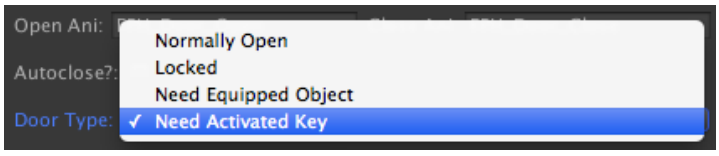

FPH is shipped with two kind of doors: Interaction and Drag.
Interaction door will open when you press the corresponding interaction button.
Drag door will open on cursor drag.

Recognize these two doors is really easy for the gamer since they have two different interaction icons.



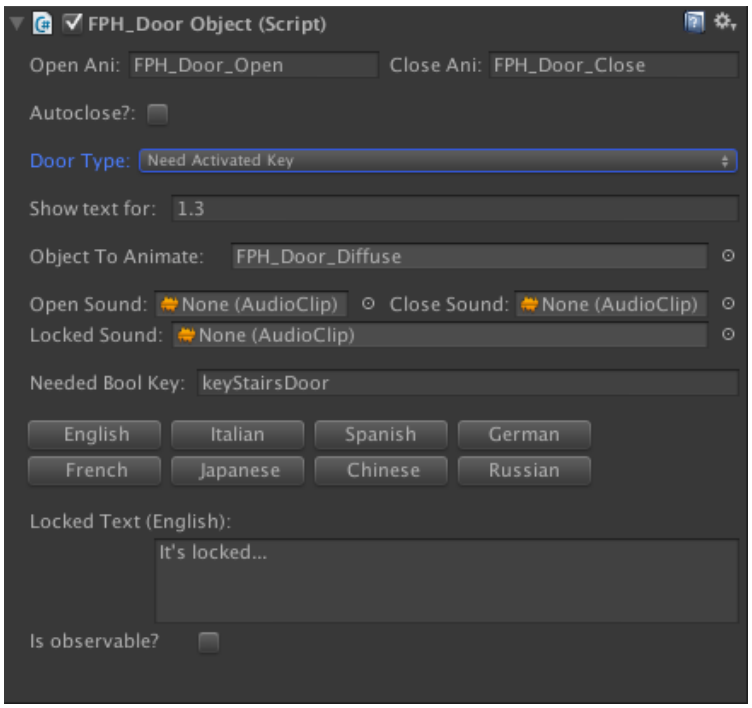
*Left is Interaction and Right is Drag*

Both interaction and drag door has four sub-type all of these type are self-explanatory.



## 8.1 - Interaction Door

Interaction Door are easier to use than drag door and need far less setup. You will only need of two animations: one when you open the door and one when you close the door. I created two simple animation inside of Unity but you can use your favorite software and import the animation inside Unity.

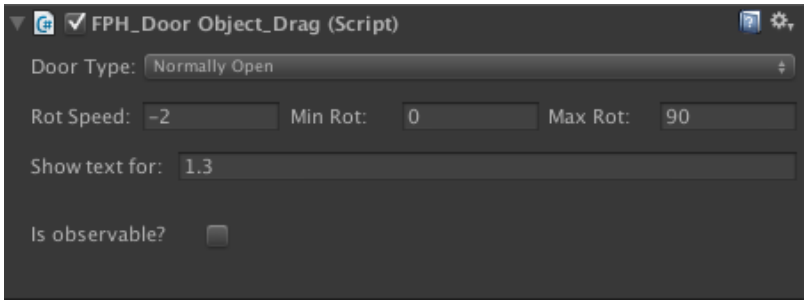



Only the door and not the wall should be selected when you setup a new door because when you click on *"Setup Door → Setup Interaction Door"* a box collider is assigned to the object and only the door should have this collider. In case you want the user to interact with another object in order to trigger open door animations you can setup this object as a door and link *"Object To Animate"* to your animated door.

NOTE: In case you are using our assets you can simply drag a door from *"Prefabs"* folder, no additional setup needed.

## 8.2 - Drag Door

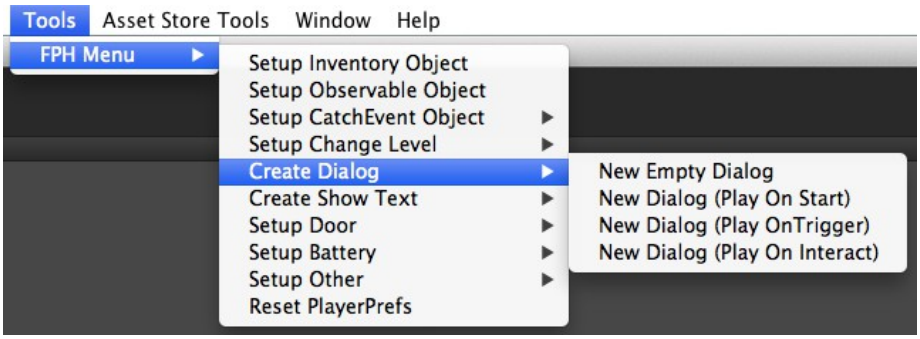
Drag Door are a bit harder to setup but nothing too difficult. After you have imported your door select it and click on *"Setup Door → Setup Drag Door"*.



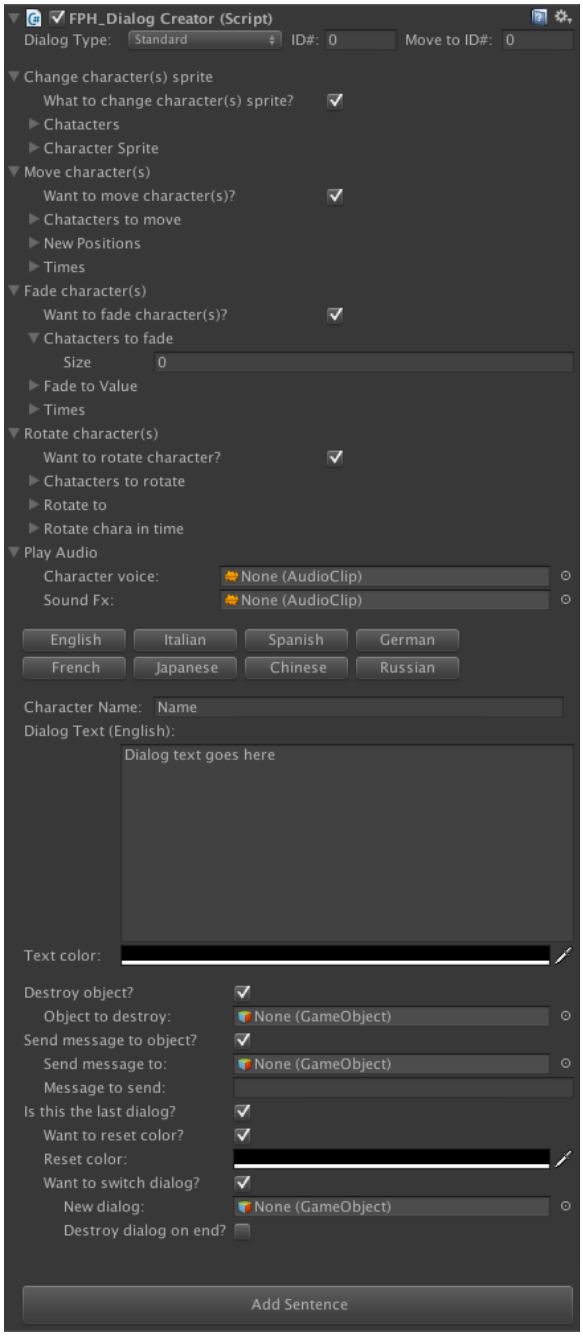

You can now create two *"Triggers"* and position them like in this picture. Now select one trigger and assign "Door Obj".


These triggers will help Drag Door script to understand player position relative to door.

# 9 - Dialog




FPH Dialog system is really powerful but it's also really easy to use and understand.
**NOTE:** Please remember to properly setup both *"_GUI_Camera"* and *"-GameManager"* or you won't be able to use FPH Dialog System.

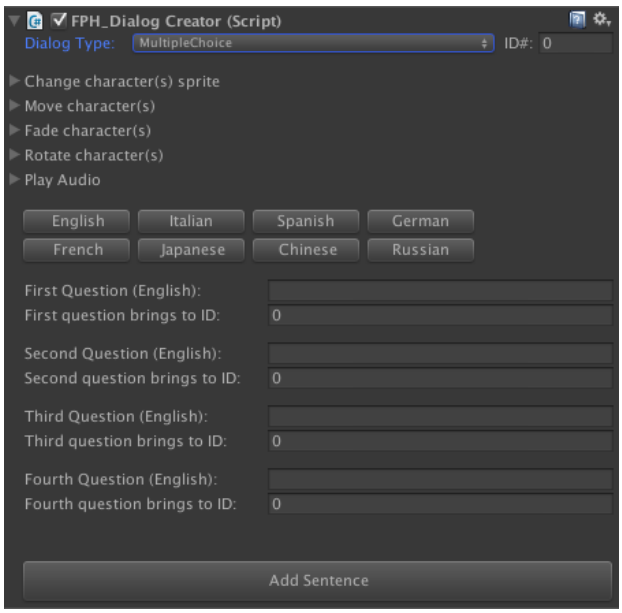



To have an example of what you can do with our Dialog system take a look at this video from our kit "Easy Visual Novel Kit"

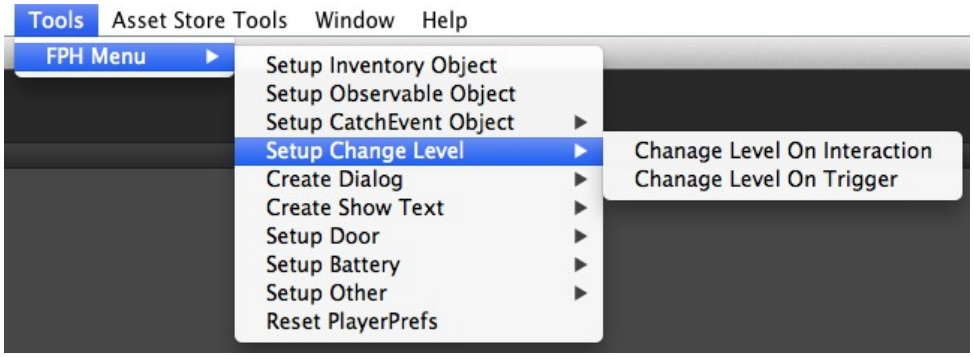
https://www.youtube.com/watch?v=4qwozp_tNiA

In case you want to create a VisualNovel like dialog remember to put characters sprite inside of DialogUI.

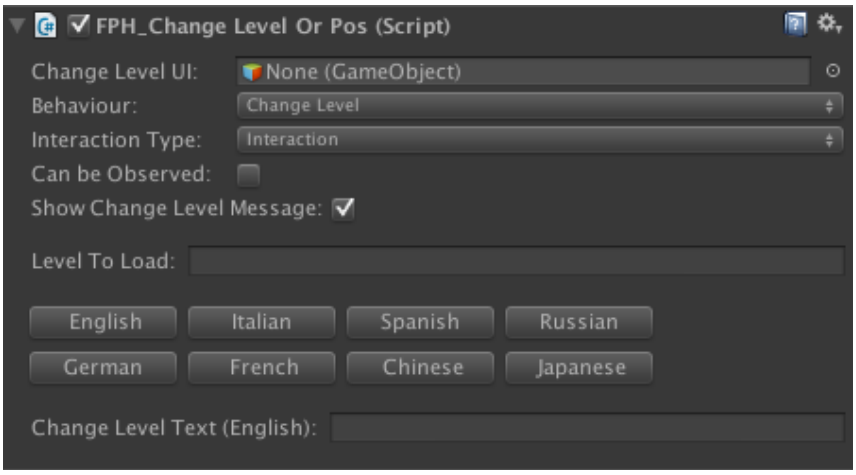
Always remember that the first dialog MUST have an ID of 0 and that even the last dialog should move to an ID even if there's no following dialogs.

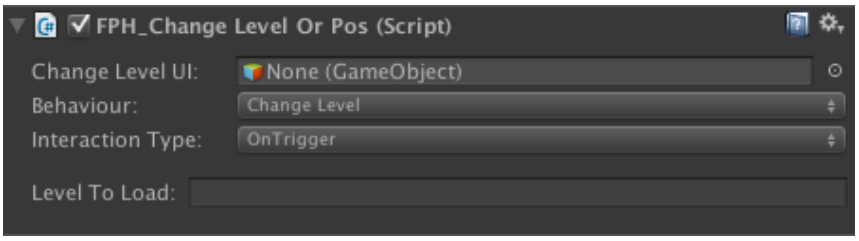


# 10 - Change Level

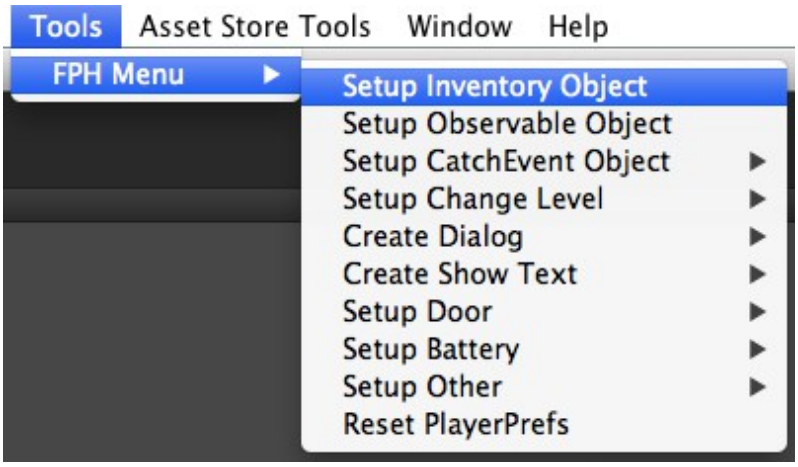



Setup a new change level object is probably the easiest things. You will only have to open FPHMenu.
There are two configuration: *Interaction* and *OnTrigger*.
*OnTrigger* option will be useful in case you want to setup a teleport or something similar.

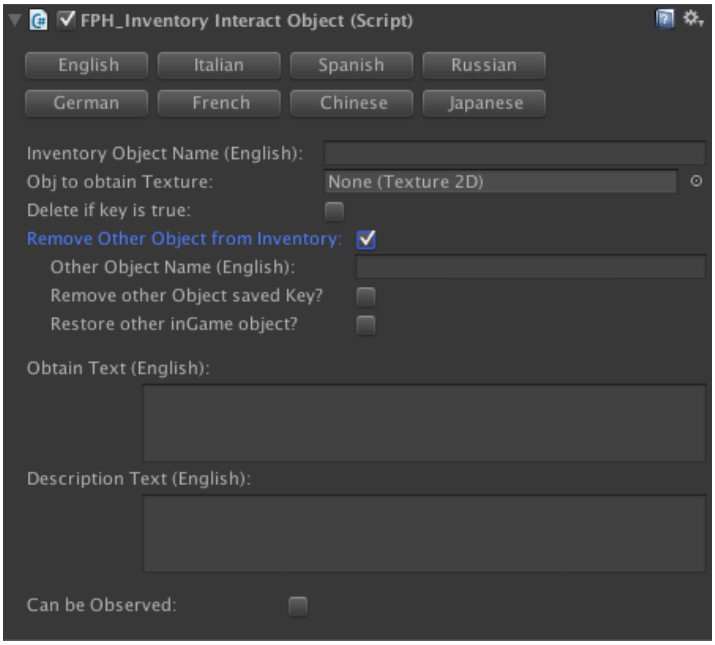


## 11 - Inventory




To setup a new inventory object you only have to open the FPHMenu. Always remember to proper setup game UI.




Inventory object inspector is easy to understand, in case you want to script a custom Inventory object here's some info you should know about.

To add a new item to the inventory you must use this line of code (must be added for every supported language)

```
FPH_InventoryManager.AddInventoryItem(string itemName, string itemTextureGUI, string itemDesc);
```

*itemTextureGUI* is the name of the inventory texture which must be placed inside of *Assets/Resources* folder.

To know if we have an object (must be added for every supported language)

```
FPH_InventoryManager.HasObject(string itemName); //Return bool
```

To remove a single item

```
// You can use selectedIndex or equippedItem_Index
FPH_InventoryManager.RemoveInventoryItem(int index);
```

To remove all items

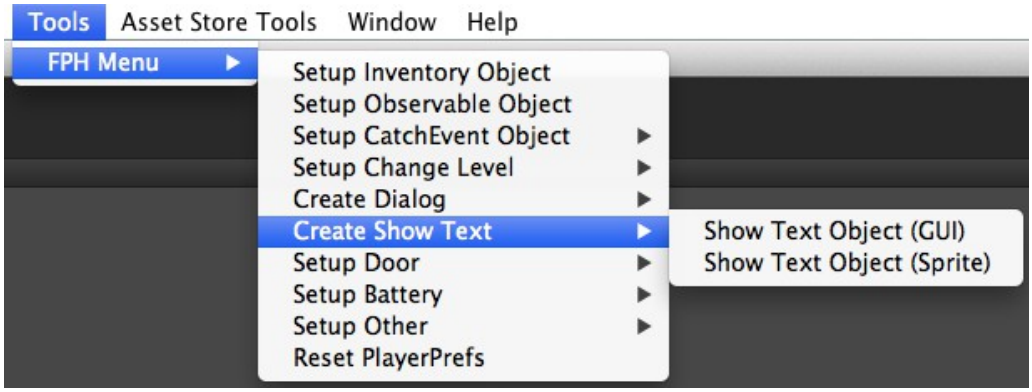
```
FPH_InventoryManager.RemoveAllInventoryItem();
```

To save or load inventory

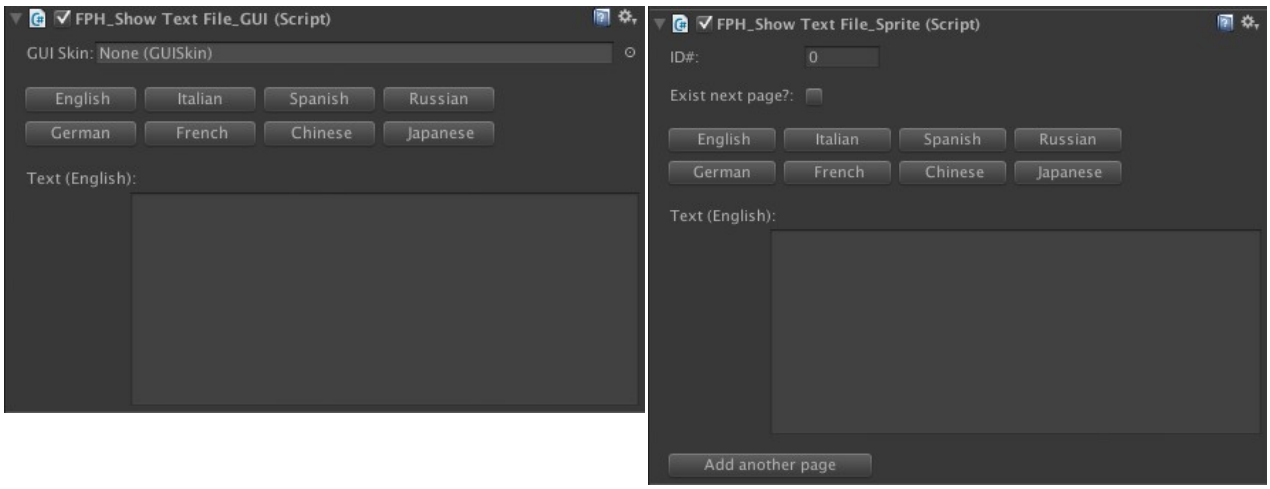
```
FPH_InventoryManager.SaveInventory();
FPH_InventoryManager.LoadInventory();
```

If you don't like our Save/Load inventory code you can use EasySave2. Take a look at the code of the functions “ SaveInventory() ” and “LoadInventory()” inside of “FPH_InventoryManager”.
You will only have to comment/uncomment some lines. In case you want to set an Inventory item in the GUI to only show one item remove “FPH_InventorySprite_ItemButton” from it and assign it “FPH_InventorySpriteDouble_ItemButton” there are some new setting for InventoryInteract Object in case you want to use this new feature.

## 12 - Show TextFile



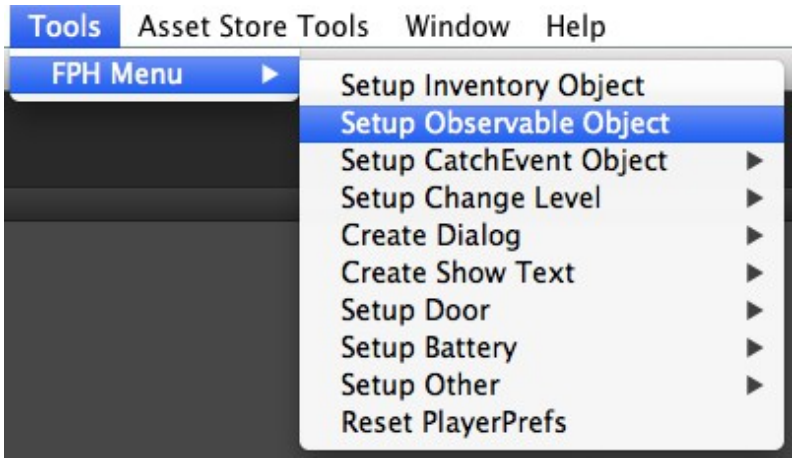

Everything has been setup to work out of the box.
You can show a text file in two different way: using sprite UI or old UnityGUI.




SpriteUI is easier to customize* and have more feature, I would recommend you to choose that instead of UnityGUI.

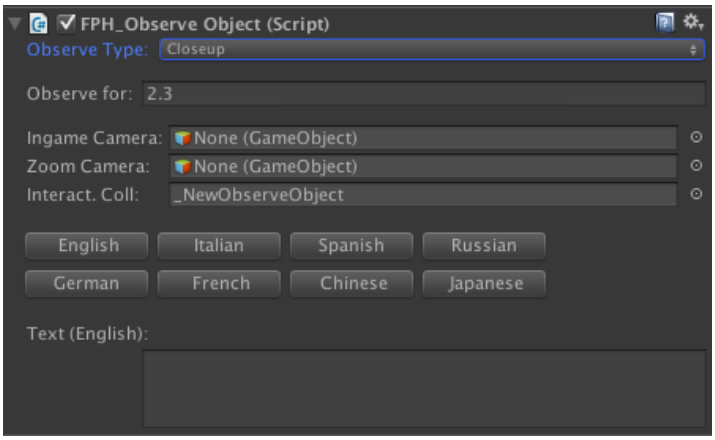
*You can go to *"_GUI_Camera → ShowTextUI"* and change the sprite if you want to customize *"ShowTextUI"*.

## 13 - Observe



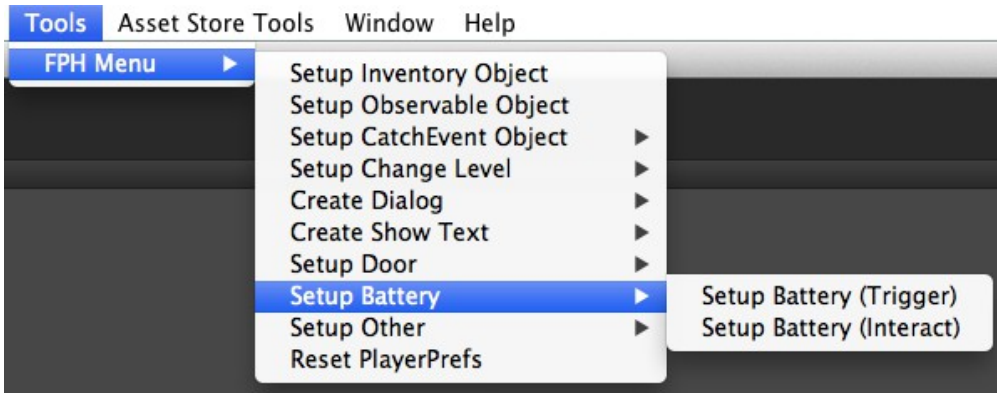

An object can be observed in two different way. One will only display a text when the player interact with it.


The other one will show a camera and a text. You can use this option to show a detail of the object you want the player to observe.

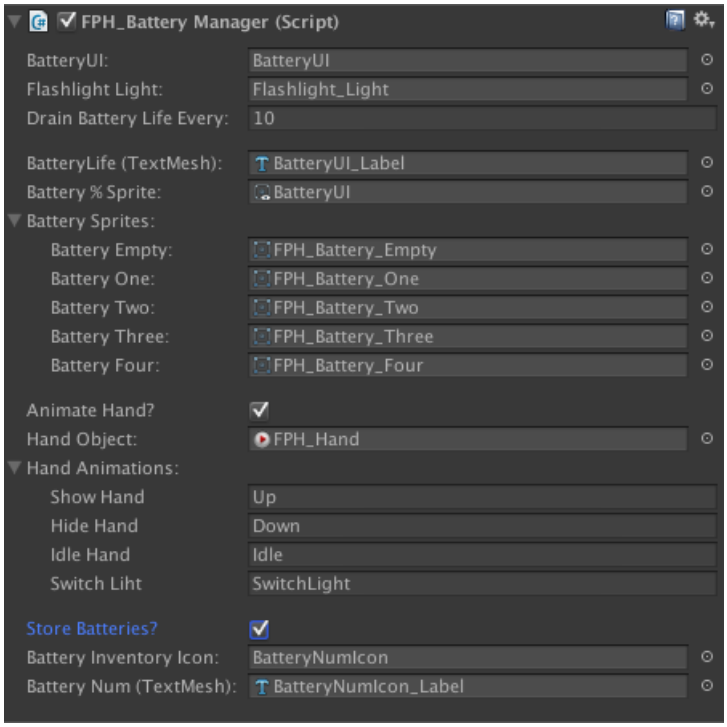



*"IngameCamera"* is player camera: *"First Person Camera"* for default FPH Player or *"_Cameras"* for third person scene sample.

## 14 - Battery




As always to setup a new battery select the object you want to use as battery and open the FPH Menu. Battery's inspector allow you to decide if you want to store the battery or immediately use it. To tweak battery system setting go to *"-GameManager"*.




In case you don't to use our battery system just remove the component from *"-GameManager"*
As you can see every single setting is really easy to understand, as always if you want to tweak buttons just modify *"FPH_ControlManager"* component of *"-GameManager"*.

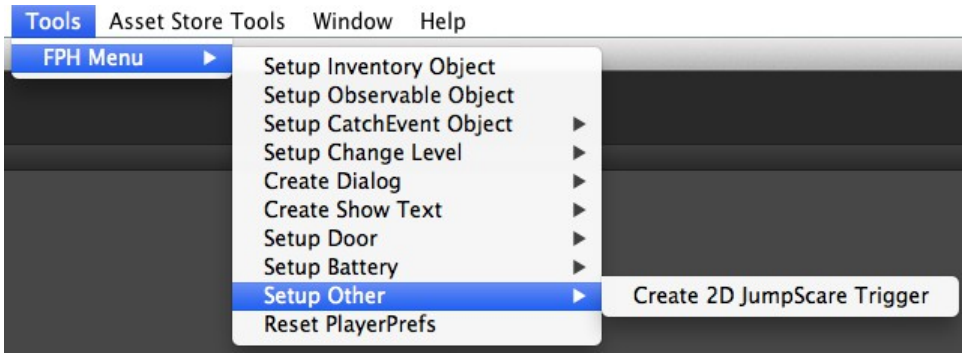
## 15 - Puzzles

Two puzzle are currently available (a new one is coming for version 1.45) to use one of these puzzle just drag a prefabs from *"-FPH → _Other → _Prefabs → GameplayPrefabs"*.

# 16 - Jumpscares

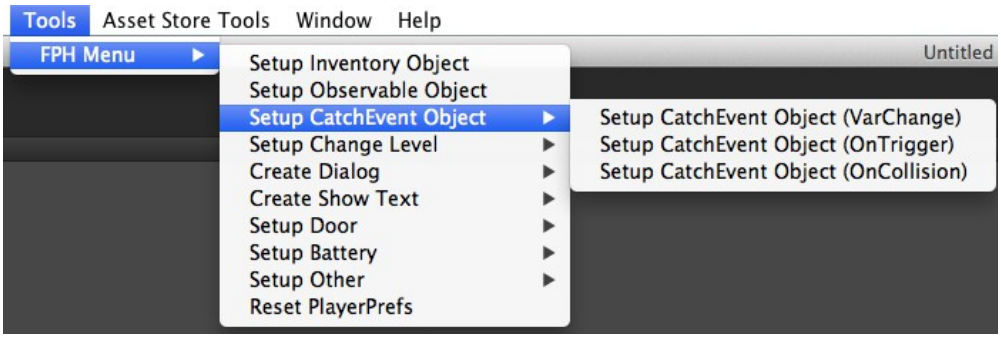
## 16.1 - 2D Jumpscare

At the moment only 2D jumpscares are available but we're working on 3D jumpscares too which will be available in the near future (around 1.45 – 1.55).
Setup a 2D jumpscare is, as always, really easy. Open the FPH Menu under Tools and press *Create 2D JumpScare Trigger*.

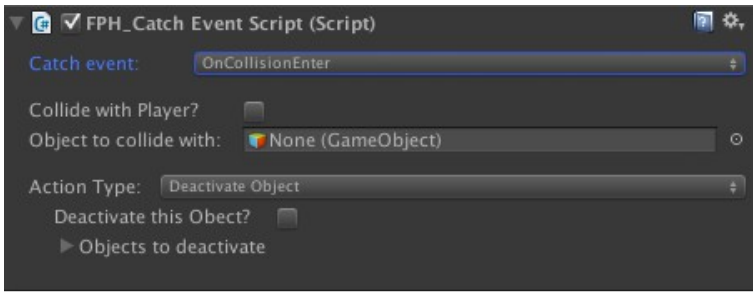


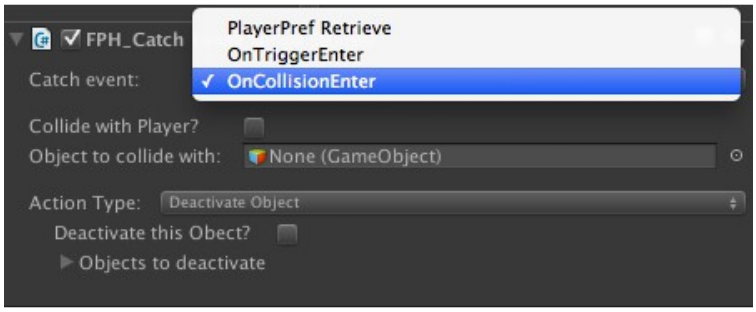

A new GameObject named *_NewJumpScare* will appear in the scene, you will only have to fill the various option and then your jumpscare will be ready.

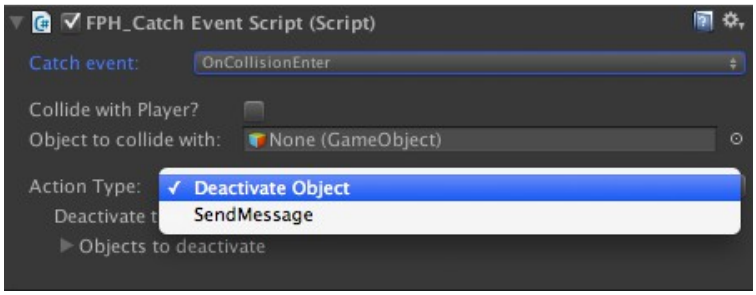
# 17 - ToggleObject on Event




This doesn't really need to much explanation. In case you want to catch an event and perform a particular action just open the FPH Menu.
If you have any request of this object don't hesitate to contact me!

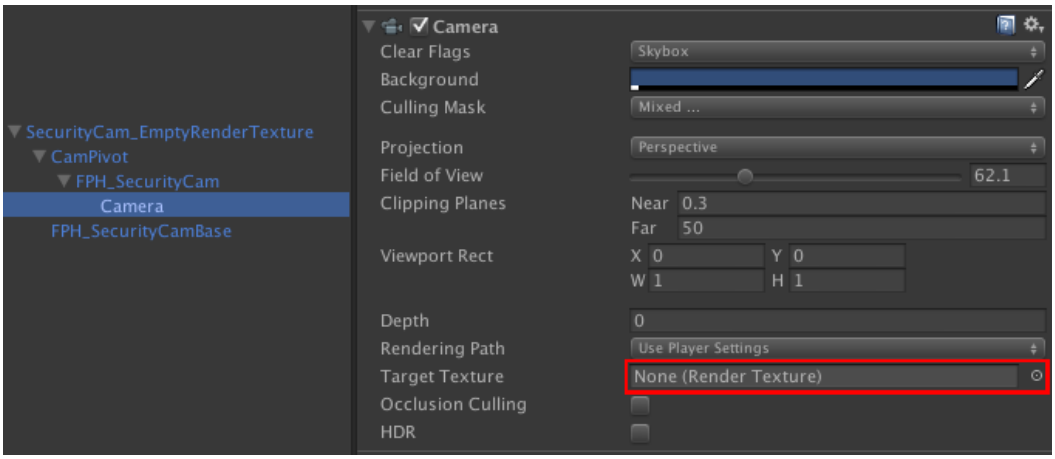


## 18 - Surveillance Camera System and Computer

Setup a new surveillance camera is really easy, you will only have to drag "SecurityCam_EmptyRenderTexture" from the *Prefabs* folder. Position it where you want and apply a new *Render Texture* to the Security Camera's camera (sorry for repeating).

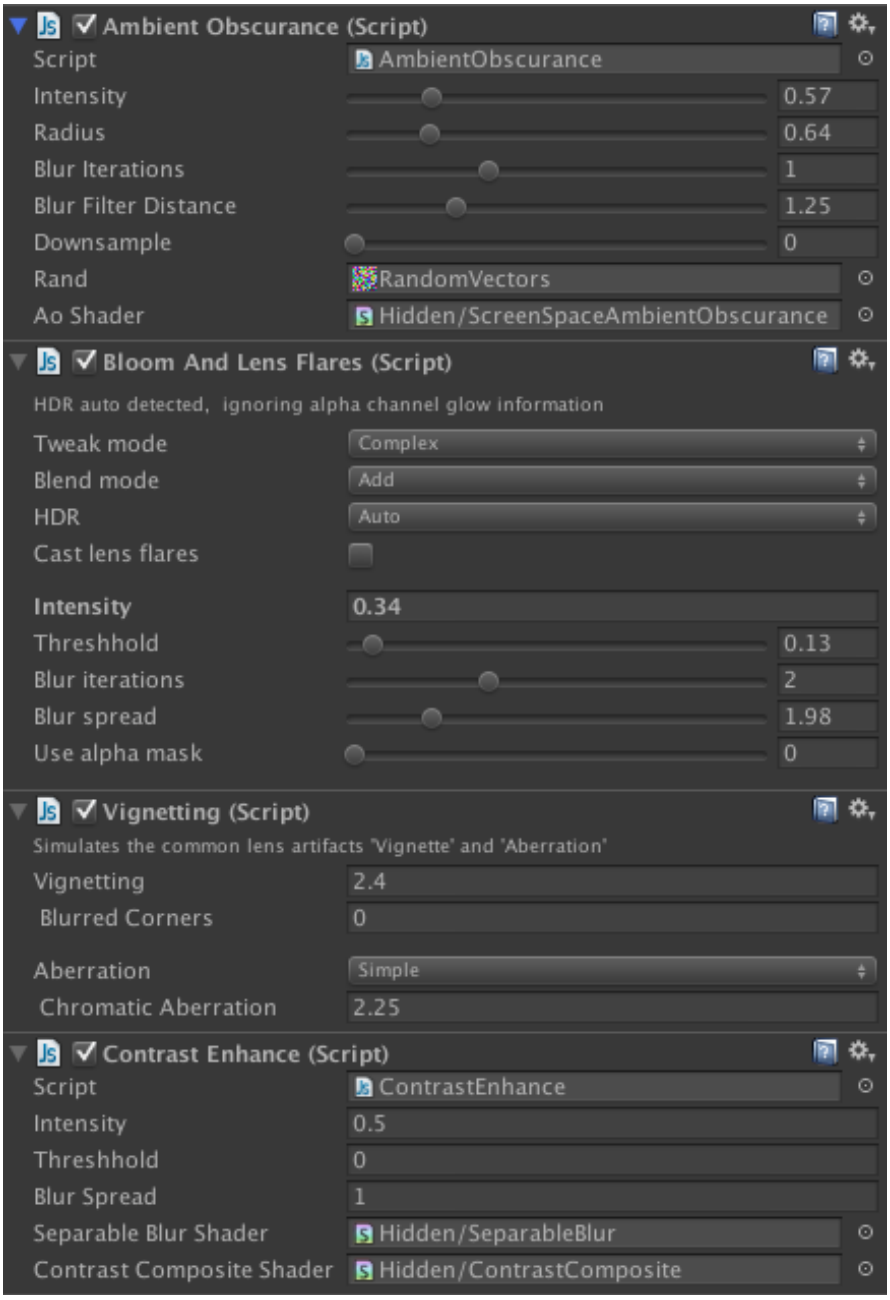



By setting the Render Texture of a Camera what's rendered by the camera will be shown by an object that has the Render Texture as main Texture, this mean that you must create a new material for every single RenderTexture.
Now that you have properly setup your SecurityCameras it's time to setup one (or more) of the PC we included in the kit, don't worry, this is an easy step too.

The first thing you have to do is to drag Computer01 or Computer02 from the *Prefabs* folder, after that remember to click on "Break Prefab Instance". Inside of every Computer you can find a *CameraRender* object, set its Material to one of the RenderTexture material that you will use in the game. By default the Player can interact with the computer to switch between different cameras, if you don't want that you will only have to to change the tag of *FPH_ComputerDesk* from *InteractObject* to *Untagged*.
In case you want the Player to interact with the computer you will only have to set the *CameraMaterials* of both *ComputerButton_BackCam* and *ComputerButton_NextCam*. Two switch between different cameras the player will only have to click (or touch) on the Right or Left arrow of the Computer's keyboard but he can also actually press the Right or Left arrow of his keyboard.

# Appendix 1: ImageEffects setting

We removed all the ImageEffects from the project but if you want to make your project look as good as in our screens you will have to add these effect (these settings are from Unity 4.3.4).



## Appendix 2: Customization

We included a lot of art inside of this kit (and more will come later) but in case you want to use your own UI you will just have to import it as sprite and change it. Again, no need to code.