

HORROR FPS KIT I.IV

MANUAL

THANKS FOR BUYING HORROR FPS KIT!

If you like my assets please go to my channel:

<https://www.youtube.com/c/ThunderWireGamesIndie>

and check out my tutorials and game developments :)

also check out my website:

<http://www.twgamesdev.com>

ABOUT HFPSKIT

HFPS KIT IS ADVANCED AND EASY HORROR TEMPLATE WITH ALL FEATURES NEEDED TO BUILD UP YOUR OWN GAME, INCLUDES ALMOST ALL FEATURES THAT HAVE LOT OF AAA HORROR GAMES. CONTAINS A LOT OF READY TO USE ASSETS, JUST DRAG AND DROP.

SUMMARY

ALL FEATURES (VERSION 1.4)	4
PROJECT SETUP (SETUP NEW SCENE)	5
HOW TO SETUP GAMECONFIG	7
CONFIG MANAGER	8
HOW TO SETUP CONFIG MANAGER TO OTHER SCRIPTS	9
CONFIG MANAGER FUNCTIONS	9
INPUT MANAGER (REBINDABLE INPUT)	10
HOW TO SETUP NEW INPUT	10
USING CONFIG IN BUILDED GAME *	11
HOW TO DESERIALIZE NEW ADDED INPUT	12
TYPE PARSER	13
HOW TO USE PARSER?	13
DYNAMIC OBJECTS (DYNAMIC MAMAGER)	14
SETTING UP DYNAMIC DOOR	14
DYNAMIC DRAWER	15
DYNAMIC LEVER	16
VALVE	17
MOVABLE INTERACT	18
DRAGGABLE OBJECTS	19
INVENTORY	20
BACKPACK PICKUP (INVENTORY EXPAND)	20
INVENTORY TWEAKS	21
INVENTORY ITEM PICKUP	21
COMBINABLE ITEM	23
INVENTORY WEAPONS AND BULLETS	24
CHANGING INVENTORY SETTINGS	25
SAVE/LOAD MANAGER (WIP)	26
ADDING SAVEABLE OBJECTS	26
SAVING CUSTOM DATA	27
SAVING CUSTOM ITEM SWITCHER DATA	28
JUMPSCARES	30
TRIGGER ANIMATION	30
JUMPSCARE ANIMATION	31
AUDIO ZONE TRIGGER	32

ADDING EXAMINE OBJECTS	33
ADDING NEW PAPERS	34
FLOATING ICON	35
ADDING NEW FOOTSTEPS	36
SHOWING CUSTOM NOTIFICATIONS	36
SIMPLE MESSAGE	36
PICKUP MESSAGE	36
WARNING MESSAGE	36
SHOWING CUSTOM HINT MESSAGE	36
BUG, ERROR REPORT	37
CREDITS	37

ALL FEATURES (VERSION 1.4)

PLAYER FUNCTIONS

- FULLY FUNCTIONAL PLAYER CONTROLLER (WALK, RUN, JUMP, CROUCH, LADDER CLIMBING)
- FOOTSTEPS SYSTEM WITH SOUNDS
- DRAG RIGIDBODY SYSTEM (ROTATE, ZOOM, THROW)
- EXAMINE AND PAPER READ SYSTEM (ROTATE, EXAMINE)
- INVENTORY SYSTEM (ADD, REMOVE, MOVE, REPLACE, USE, COMBINE)
- WALL DETECT SYSTEM (HIDE WEAPON)
- WEAPONS (GLOCK18)
- FALL DAMAGE
- PLAYER LEAN
- ZOOM EFFECT
- INTERACT SYSTEM
- UI CROSSHAIR

OBJECT PICKUPS

- CUSTOM OBJECT PICKUP SCRIPT
- FLASHLIGHT PICKUP (BATTERIES)
- CANDLE PICKUP
- LOCKED DYNAMIC OBJECT KEY PICKUP
- INVENTORY ITEM PICKUP
- BACKPACK PICKUP (EXPAND INVENTORY)

DYNAMIC FUNCTIONS

- DYNAMIC FUNCTIONS (DOOR, LEVER, DRAWER, VALVE, MOVABLE INTERACT)
- DYNAMIC OBJECT ANIMATION FUNCTION
- DYNAMIC OBJECT TYPES (NORMAL, LOCKED, JAMMED)
- DRAGGABLE OBJECTS (DOOR, DRAWER, LEVER)
- KEYPAD

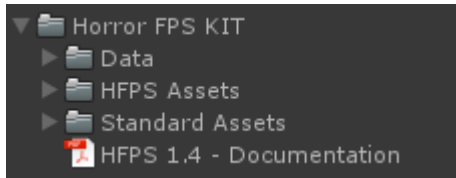
MORE FUNCTIONS

- SAVE/LOAD SYSTEM (WIP, SAVING AND LOADING TRANSFORMS, SCRIPTS...)
- CONFIG MANAGER (SAVE AND READ YOU OWN .CFG FILES)
- CONFIG READER
- HELPERS (INPUT HELPER, TYPE PARSER)
- SAVING GAME OPTIONS TO CONFIG
- REBINDABLE INPUT MANAGER
- AI ZOMBIE SYSTEM (WALK, RUN, ATTACK, PATROL)
- UI MENUS (MAIN MENU, LOAD GAME MENU, PAUSE MENU)
- GAME OPTIONS (GENERAL, GRAPHIC, CONTROLS)
- JUMPSCARE ANIMATION (SCARED BREATHING, SCARED EFFECT)
- LAMPS (NORMAL, FLICKERING)
- FLOATING ICON (ICON FLOATING ON OBJECT)
- SNAPABLE, SEAMLESS WALLS
- PROPS, PICKUPABLE OBJECTS, AND MUCH MORE..
- AMBIENCE SOUND CHANGE
- HINT MANAGER
- PICKUP NOTIFICATIONS (ITEM NAME, MESSAGE, HINT)
- FULLY FUNCTIONAL UI

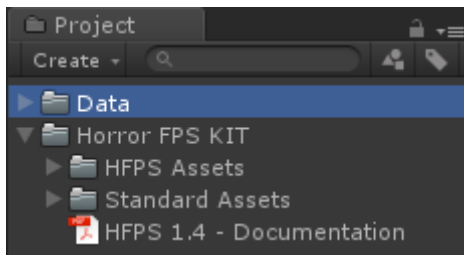
PROJECT SETUP (SETUP NEW SCENE)

IS RECOMMENDED IMPORT HFPSKIT TO EMPTY PROJECT!

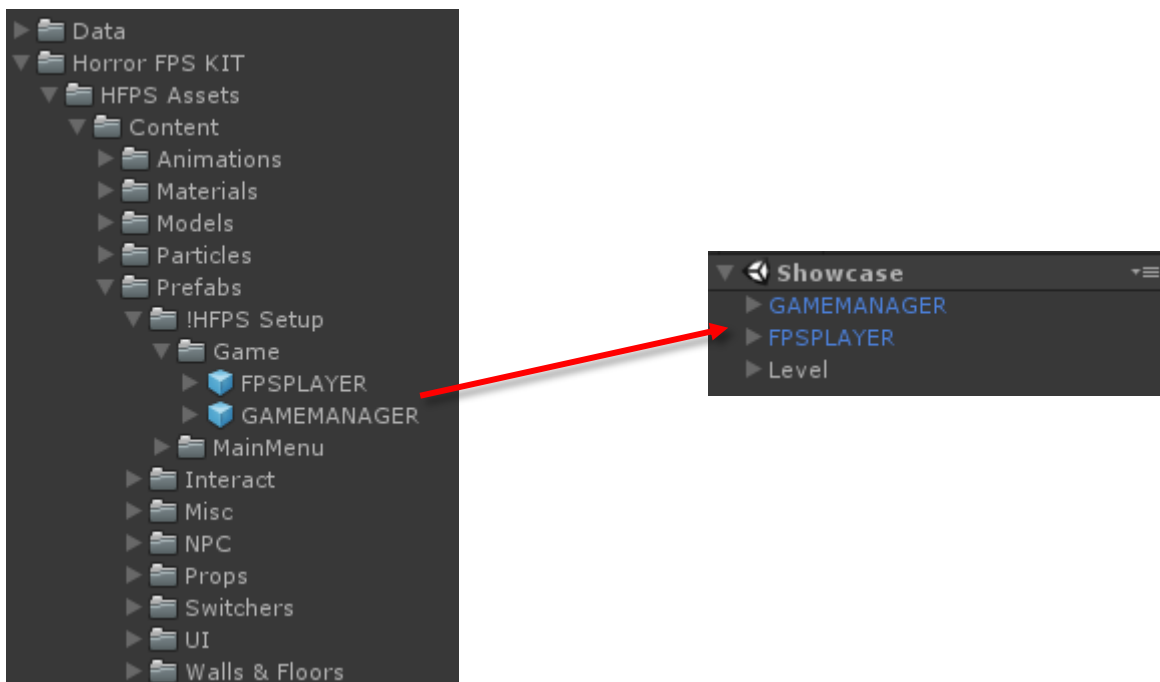
1. IMPORT HFPS KIT TO EMPTY PROJECT (ALL PROJECT SETTINGS WILL BE OVERWRITTEN!)



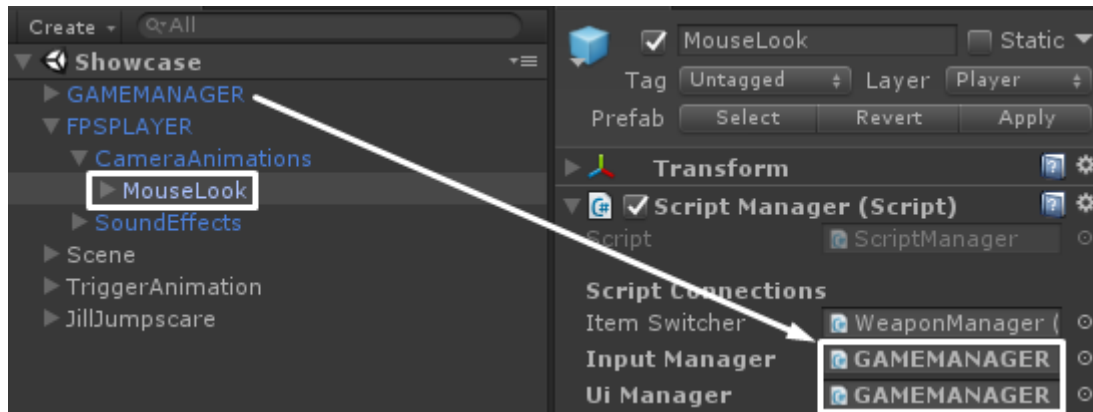
2. MOVE **DATA** FOLDER TO YOUR PROJECT ASSET FOLDER



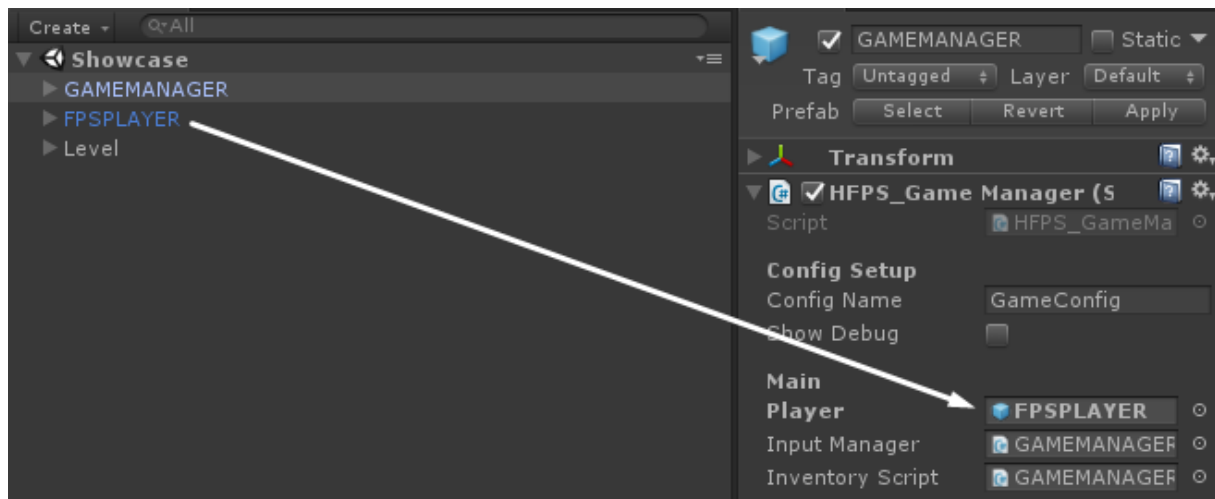
3. THEN GO TO **HFPS ASSETS -> CONTENT -> PREFABS -> !HFPS SETUP -> GAME** AND DRAG **GAMEMANAGER** AND **FPSPLAYER** TO YOUR SCENE



4. THE IMPORTANT STEP IS SETUP SCRIPT CONNECTIONS! (CONNECT **GAMEMANAGER** WITH **FPSPLAYER**)



5. CONNECT FPSPLAYER GAMEOBJECT WITH GAMEMANAGER



6. RUN HFPS FROM MAIN MENU TO SET GAMECONFIG LOCATION!!
(IMPORTANT) WITHOUT IT PLAYER WILL NOT MOVE

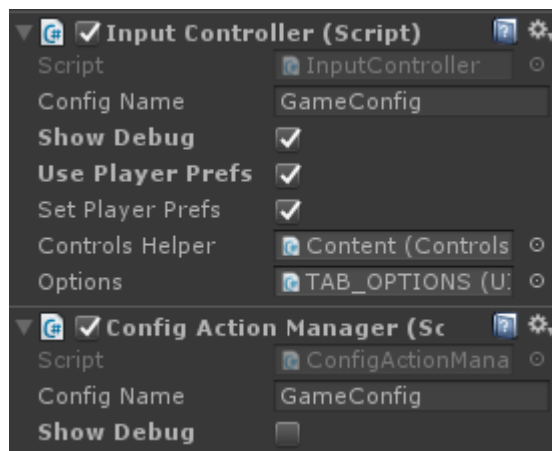
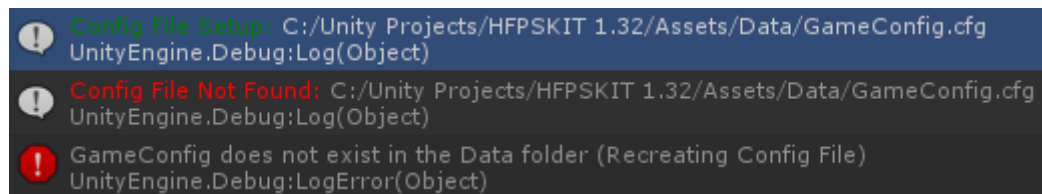


HOW TO SETUP GAMECONFIG

FOLDER NAMED **DATA** IS DEFAULT FOLDER TO STORE GAMECONFIG.

IF **GAMECONFIG** DOES NOT EXIST IN **DATA** FOLDER YOU NEED TU RUN GAME FROM MAIN MENU TO RECREATE IT WITH DEFAULT VALUES.

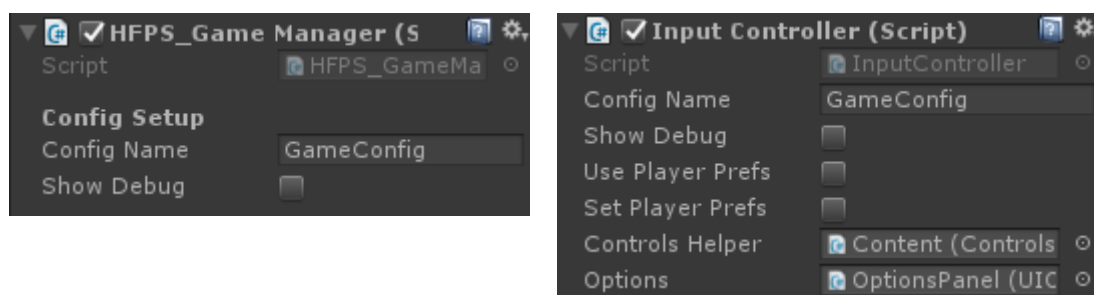
MAIN MENU:



IN MAIN MENU CHOOSE **SETPLAYERPREFS** TO SAVE **CONFIGNAME** AS A DEFAULT CONFIG. WITHOUT IT YOU MUST GO TO ALL GAME SCENES AND CHANGE DEFAULT CONFIGNAME MANUALLY.

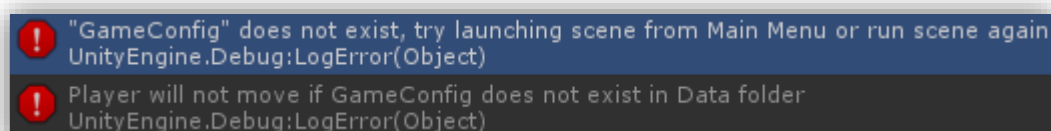
IF YOU SET **USEPLAYERPREFS** TO TRUE GAME WILL SAVE GAMECONFIG NAME AND BOOL TO LOAD FROM PLAYERPREFS

GAME SCENE:



IN GAME KEEP SET AND USE PLAYER PREFS TO FALSE

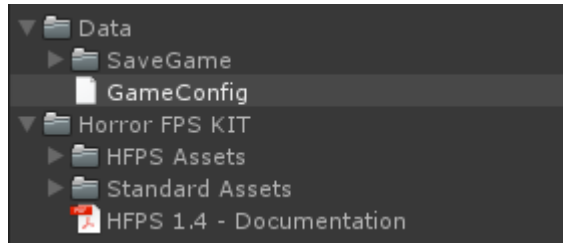
WITHOUT IT YOU WILL GET THESE ERRORS:



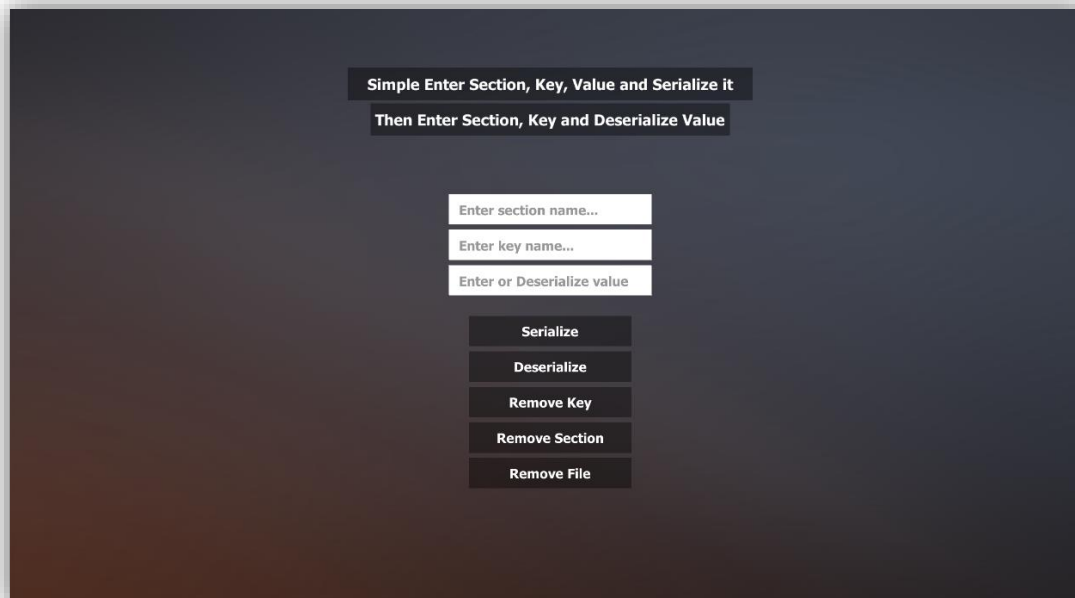
CONFIG MANAGER

IS SIMPLE SERIALIZATION MANAGER TO SAVE AND READ YOUR OWN .CFG FILES

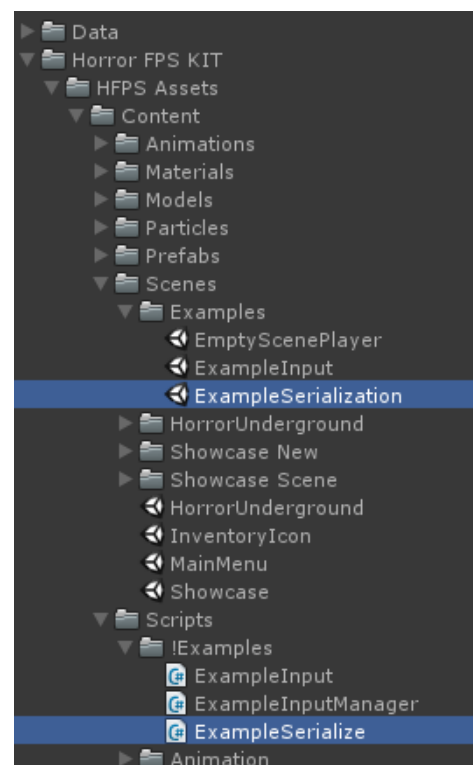
- ALL CONFIG FILES IS STORED INSIDE PROJECT OR INSIDE EXPORTED GAME TO FOLDER NAMED **DATA**



YOU CAN EASILY VIEW OR EDIT CONFIG BY THE **EXAMPLESERIALIZATION** SCENE



ALSO THE EXAMPLE SCRIPT
FOR SERIALIZATION IS
INCULDED IN SCRIPTS
FOLDER



HOW TO SETUP CONFIG MANAGER TO OTHER SCRIPTS

1. SIMPLY BY ADDING NAMESPACE:

```
using ThunderWire.Configuration;
```

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using ThunderWire.Configuration;
```

2. THEN YOU MUST ADD DEFINITION:

```
ConfigManager config = new ConfigManager();
```

```
public class ExampleSerialize : MonoBehaviour {
    ConfigManager config = new ConfigManager();
```

3. AND THE MAIN PART IS SETUP CONFIG FOLDER AND NAME

```
config.Setup(ShowDebug(True, False), "ConfigName");
```

```
void Start () {
    config.Setup(showDebug, ConfigName);
}
```

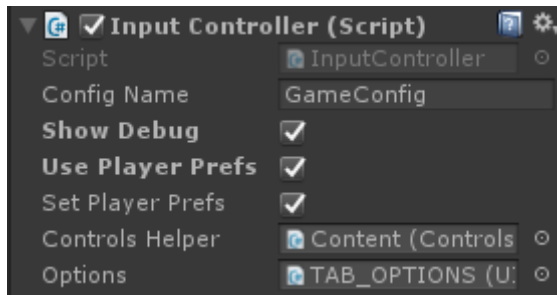
CONFIG MANAGER FUNCTIONS

config.Setup(Debug, "ConfigName"); - SETUP CONFIG DEBUGGING AND NAME
config.SetupFolder(Debug, Folder, "ConfigName") - SETUP CONFIG WITH FOLDER
config.Serialize("Section", "Key", "Value"); - SERIALIZE TO CONFIG FILE
config.Deserialize("Section", "Key"); - DESERIALIZE FROM CONFIG (STRING)
config.ContainsSection("Section"); - CHECK IF CONFIG HAVE SECTION (BOOL)
config.ContainsSectionKey("Section", "Key",); - CHECK IF SECTION HAVE KEY (BOOL)
config.ContainsKeyValue("Section", "Key", "Value",); - CHECK IF KEY HAVE VALUE (BOOL)
config.RemoveSectionKey ("Section", "Key"); - REMOVE KEY FROM SECTION
config.RemoveSection ("Section"); - REMOVE SECTION FROM CONFIG FILE
config.GetSectionKeys ("Section"); - GET COUNT OF SECTION KEYS (INT)
config.ExistFile ("ConfigFolder", "ConfigName "); - CHECK IF CONFIG EXIST (BOOL)
config.ExistFileInFolder ("File", "Folder "); - CHECK IF CONFIG EXIST IN FOLDER (BOOL)
config.ExistFileWithPath("FullPath", "File"); - CHECK IF CONFIG EXIST IN PATH (BOOL)
config.RemoveFile("File"); - REMOVE FILE FROM DATA FOLDER
config.DuplicateFile("File", "Name"); - DUPLICATE FILE
config.GetDataPath(); - GET FULLPATH TO THE DATA FOLDER (STRING)
config.GetFileAndPath("File"); - GET FULLPATH TO THE FILE (STRING)
config. GetFileAndPathFolder("File", "Folder"); - GET FULLPATH TO THE FILE INSIDE FOLDER (STRING)

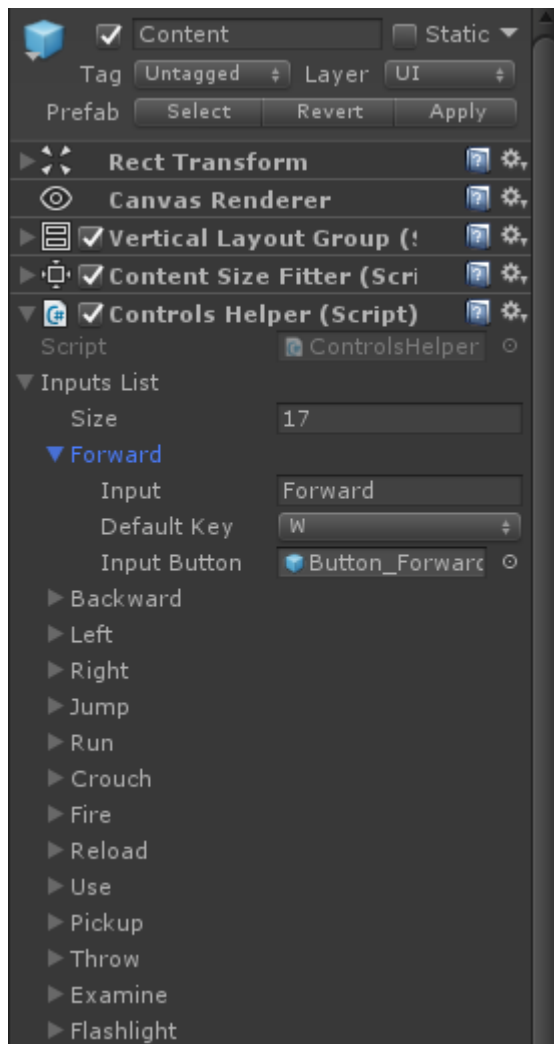
INPUT MANAGER (REBINDABLE INPUT)

HOW TO SETUP NEW INPUT

1. GO TO **HFPS ASSETS -> CONTENT -> SCENES** AND OPEN FOR EXAMPLE **MAINMENU** SCENE
2. SELECT **MENUMANAGER**
3. IF YOU WANT CHANGE CONFIG FILE NAME YOU CAN CHANGE IT IN **InputController.cs** SCRIPT BY CHANGING **CONFIGNAME**



4. NEXT YOU NEED TO CREATE OR EDIT INPUTS IN **TAB_CONTROLS** GAMEOBJECT **WITH SAME CHILD OBJECTS!**
5. IF YOU COMPLETED EDITING INPUTS THE MAIN PART IS ADD **ControlsHelper.cs** SCRIPT TO CONTENT GAMEOBJECT.



6. SCRIPT WILL AUTOMATICALLY ADD INPUTS TO LIST WITH PRE SET VALUES THAT IS SET IN INPUT BUTTONS
7. THEN YOU MUST SET CONTROL HELPER LOCATION TO INPUT CONTROLLER SCRIPT
8. **YOU NEED TO REPEAT ALL STEPS IN GAMEMANAGER**
9. DONE

USING CONFIG IN BUILDED GAME *

THIS IS IMPORTANT PART

WHEN YOU BUILD GAME THE CREATED CONFIG WITH INPUT DOES NOT COME TO BUILDED GAME LOCATION! YOU MUST COPY **DATA** FOLDER TO GAME BUILD LOCATION "**\YOURGAME_DATA**"

OR

WHEN YOU START GAME AND CONFIG DOES NOT EXIST IN THE DATA FOLDER THE **InputController.cs** SCRIPT AUTOMATICALLY CREATE CONFIG FILE SO YOU DOESN'T NEED TO COPY FROM PROJECT.

Data	11.09.2017 18:32	Priečinok súborov	
GI	11.09.2017 18:29	Priečinok súborov	
Managed	11.09.2017 18:29	Priečinok súborov	
Mono	11.09.2017 18:29	Priečinok súborov	
Resources	11.09.2017 18:29	Priečinok súborov	
app.info	11.09.2017 18:28	Súbor INFO	1 kB
boot.config	11.09.2017 18:28	XML Configuratio...	0 kB
globalgamemangers	11.09.2017 18:27	Súbor	37 kB
globalgamemangers.assets	11.09.2017 18:27	Súbor ASSETS	42 kB
level0	11.09.2017 18:27	Súbor	178 kB
level1	11.09.2017 18:27	Súbor	727 kB
level2	11.09.2017 18:27	Súbor	398 kB
level2.resS	11.09.2017 18:27	Súbor RESS	129 kB
resources.assets	11.09.2017 18:28	Súbor ASSETS	4 937 kB
resources.assets.resS	11.09.2017 18:28	Súbor RESS	969 kB
sharedassets0.assets	11.09.2017 18:28	Súbor ASSETS	66 kB
sharedassets0.assets.resS	11.09.2017 18:28	Súbor RESS	9 300 kB
sharedassets1.assets	11.09.2017 18:28	Súbor ASSETS	19 190 kB
sharedassets1.assets.resS	11.09.2017 18:28	Súbor RESS	310 794 kB
sharedassets1.resource	11.09.2017 18:28	Súbor RESOURCE	2 168 kB
sharedassets2.assets	11.09.2017 18:28	Súbor ASSETS	4 002 kB
sharedassets2.assets.resS	11.09.2017 18:28	Súbor RESS	304 534 kB
sharedassets2.resource	11.09.2017 18:28	Súbor RESOURCE	3 399 kB

HOW TO DESERIALIZE NEW ADDED INPUT

- FOR EXAMPLE GO TO SCRIPT EXAMPLES AND OPEN **ExampleInput.cs**
- IN **INPUTMANAGER** GAMEOBJECT YOU HAVE **CONFIG ACTION MANAGER** THAT HAVE ALL CONFIGMANAGER ACTIONS.
- IF YOU HAVE NEW SCRIPT YOU MUST CONNECT IT WITH **ConfigActionManager.cs** OR **InputManager.cs**

1. WRITE PARSER NAMESPACE

```
using ThunderWire.Parser;
```

2. WRITE DEFINITION

```
private ParsePrimitives parser = new ParsePrimitives();
```

3. DEFINE **ConfigActionManager**

```
public ConfigActionManager configActions;
```

4. DEFINE NEW KEY

```
private KeyCode useKey;
```

5. WRITE PARSING SENTENCE TO UPDATE

```
void Update()
{
    if (configActions.GetKeysCount() > 0 && !isSet)
    {
        useKey = parser.ParseType<KeyCode>(configActions.Deserialize("Input", "Use"));
        isSet = true;
    }

    if (Input.GetKeyDown(useKey) && !isPressed)
    {
        Debug.Log("Use Key Pressed!");
        isPressed = true;
    }
    else if (isPressed)
    {
        isPressed = false;
    }
}
```

IF YOU WANT MORE ADVANCED PARSING BY **INPUTMANAGER** OPEN **ExampleInputManager.cs** SCRIPT

TYPE PARSER

- IF YOU NEED PARSE STRING TO A CORRECT TYPE JUST USE MY SIMPLE PARSER
- **SUPPORTED PARSES:**
 - Vector2, Vector3, Vector4, Quaternion, int, uint, Long, uLong, float, double, bool, char, short, byte, Color, KeyCode

HOW TO USE PARSER?

1. WRITE PARSER NAMESPACE

```
using ThunderWire.Parser;
```

2. WRITE PARSER DEFINITION

```
private ParsePrimitives parser = new ParsePrimitives();
```

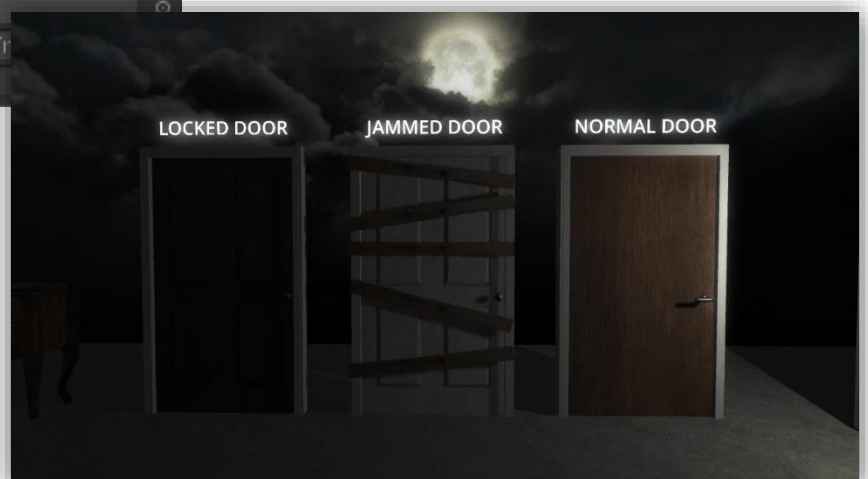
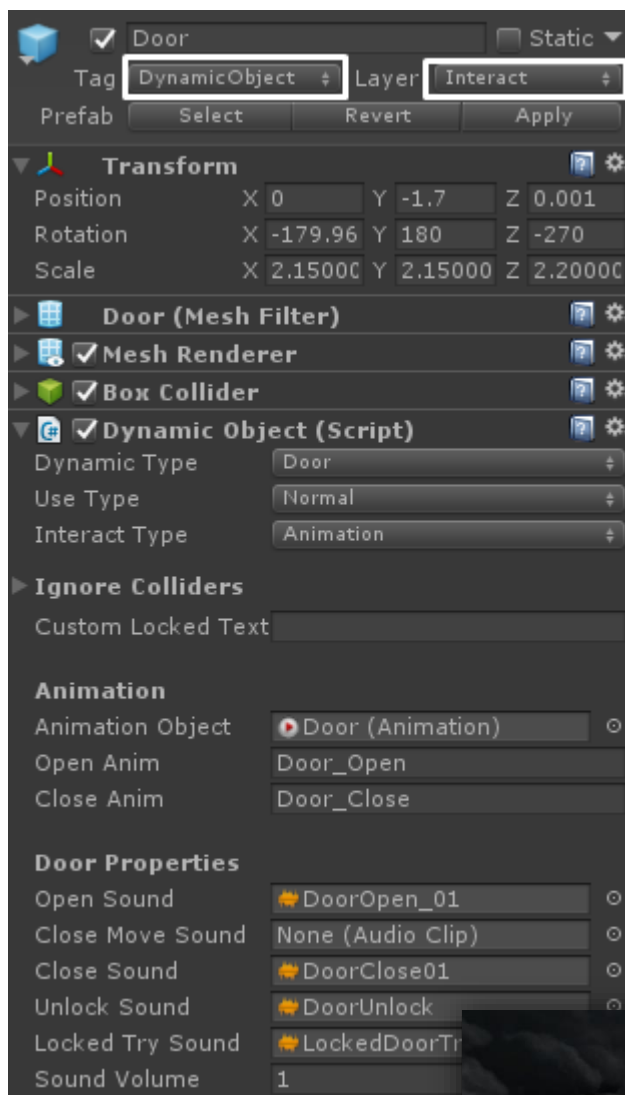
TO PARSE FROM STRING TO CORRECT TYPE USE THIS COMMAND

```
parser.ParseType<TYPE>( "STRING" );
```

DYNAMIC OBJECTS (DYNAMIC MAMAGER)

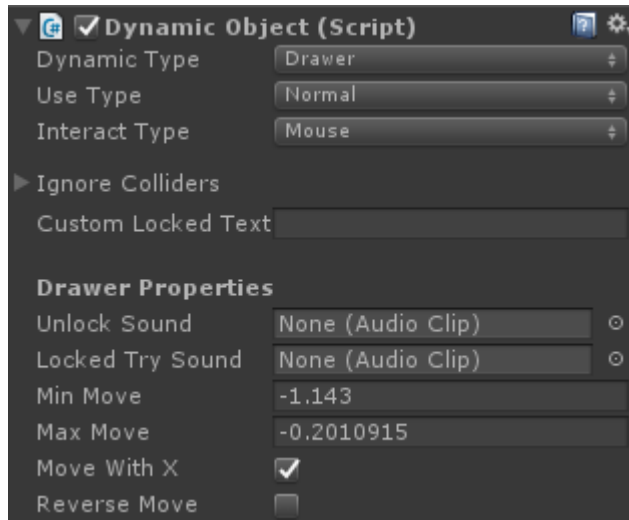
SETTING UP DYNAMIC DOOR

- DYNAMIC OBJECTS MUST HAVE **DynamicObject** TAG AND **Interact** LAYER
- YOU CAN SWITCH BETWEEN USE TYPES (Normal, Locked, Jammed)
- IF LIKE OPENING DOOR WITH **MOUSE** OR **ANIMATION** YOU CAN ALWAYS SWITCH BETWEEN THESE MODES

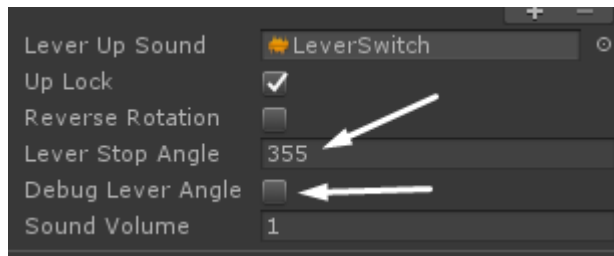


DYNAMIC DRAWER

- DRAWER **MIN** AND **MAX** MOVE POSITIONS IS NORMALLY SET BY **TRANSFORM X POSITION** BUT IF YOU USING CUSTOM DRAWER THAT NEEDS **TRANSFORM Z POSITION** YOU CAN CHECK OFF **MOVE X BOOL** AND USE **TRANSFORM Z POSITION**



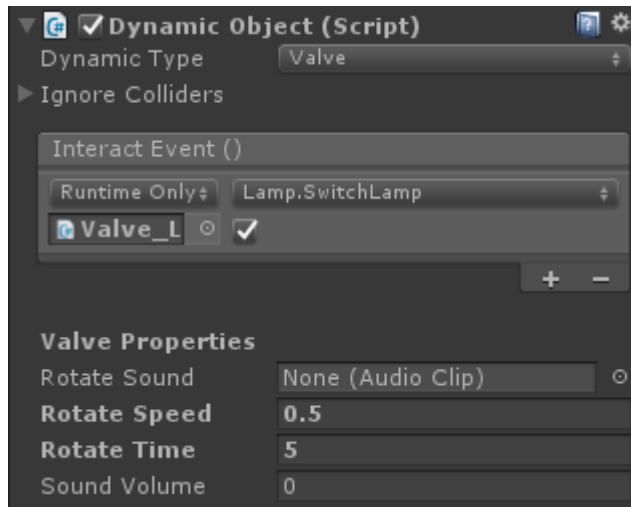
DYNAMIC LEVER



- FIRST YOU MUST DEFINE LEVER STOP ANGLE
- IF YOU SELECT **DEBUG LEVER ANGLE** YOU WILL GET MESSAGE IN DEBUG OF THE CURRENT LEVER ANGLE SO YOU CAN EASILY SET LEVER **ANGLE STOP**
- IF YOU MOVE LEVER UP AND YOU HAVE TICKED **UP LOCK** THE LEVER WILL LOCK ON UP STATE PERMANENTLY SO YOU CANT MOVE LEVER DOWN



VALVE

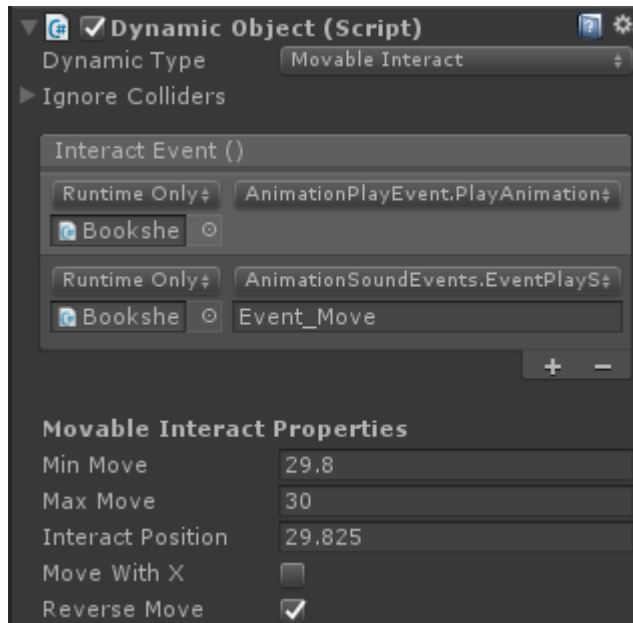


- BY CHANGING **ROTATE SPEED** YOU CAN CHANGE VALVE ROTATING SPEED
- BY CHANGING **ROTATE TIME** YOU CAN SET HOW LONG YOU NEED ROTATE VALVE TO INVOKE INTERACT EVENT

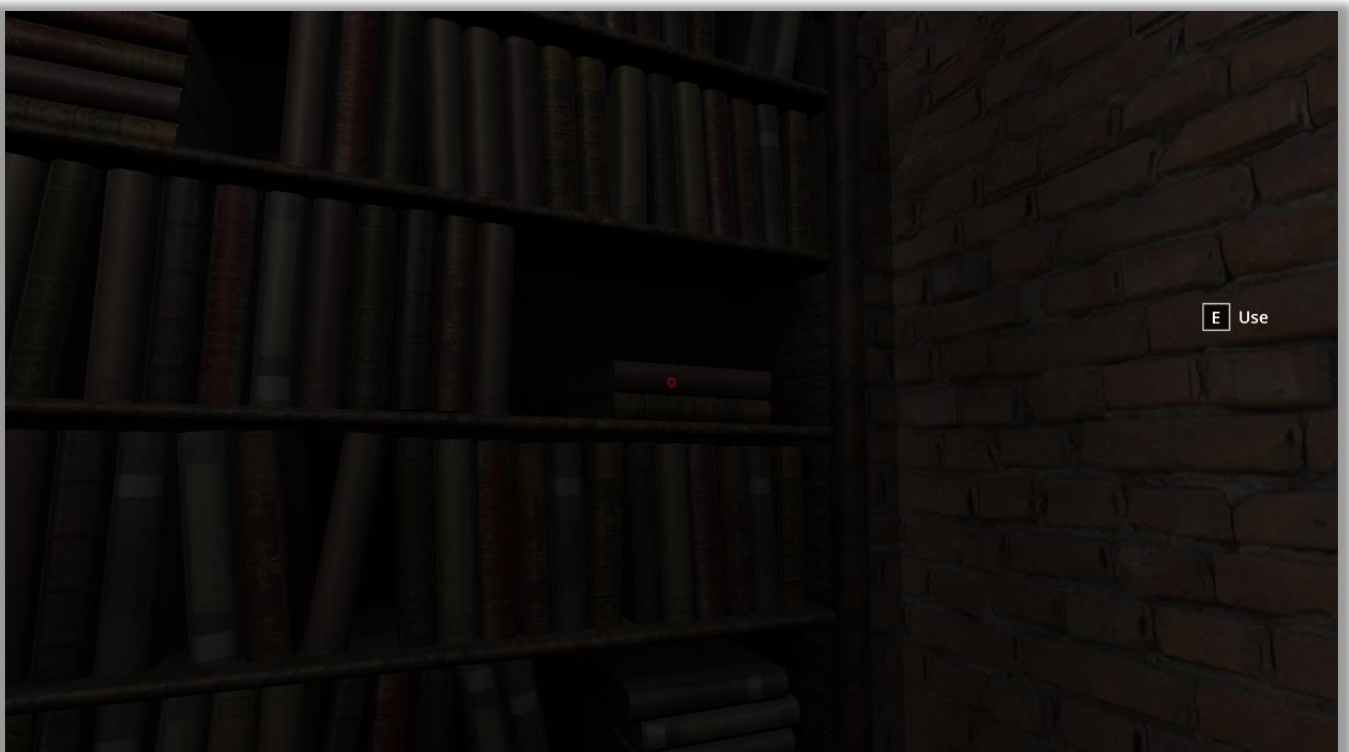


MOVABLE INTERACT

- THIS IS GOOD FOR MAKING SECRET ROOMS



- THIS IS NORMALLY A DYNAMIC DRAWER BUT WITH INTERACT FUNCTION
- WHEN YOU TICK **MOVE WITH X** BOOL THE SCRIPT WILL MOVE WITH TRANSFORM **X** POSITION
- WHEN THE POSITION **Z** OR **X** OF OBJECT IS IN INTERACT POSITION, SCRIPT WILL INVOKE INTERACT EVENT

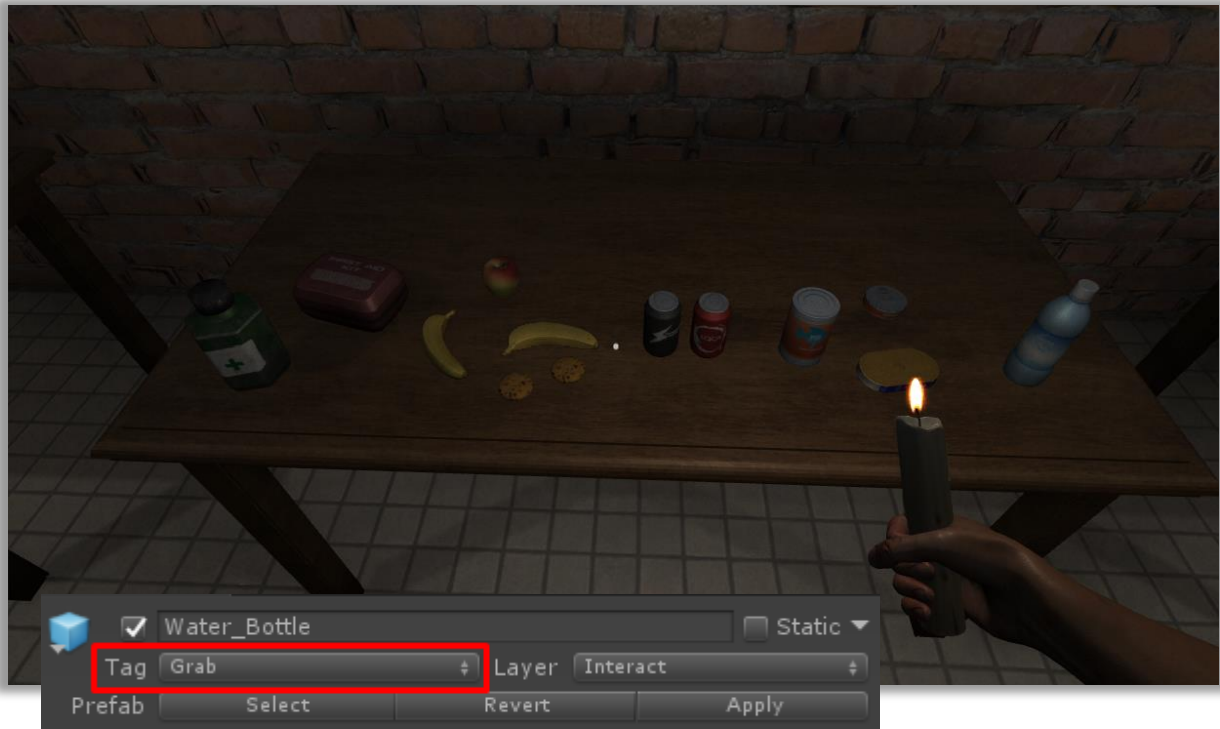


DRAGGABLE OBJECTS

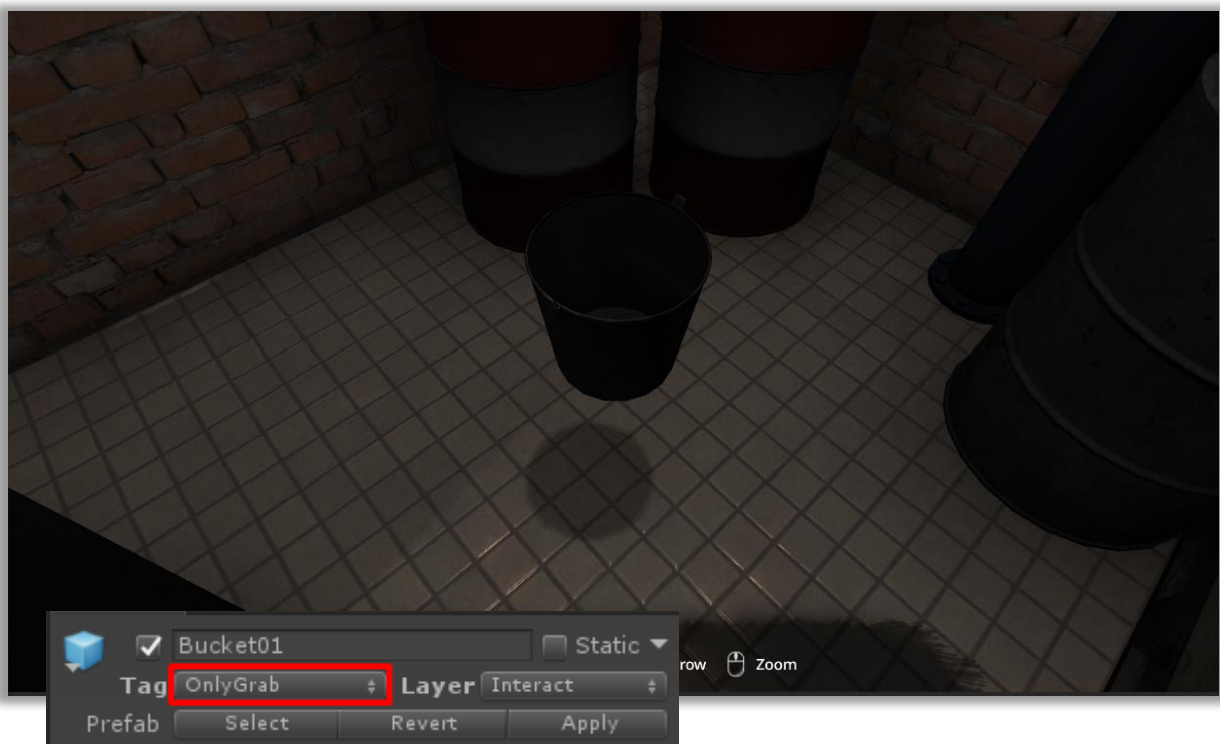
ALL OBJECTS TAGGED WITH **GRAB** OR **ONLYGRAB** TAG WILL BE DRAGGABLE

- YOU CAN ROTATE, ZOOM AND THROW DRAGGED OBJECT
- OBJECTS WITH **GRAB** TAG CAN BE INTERACTED

THE **GRAB** TAG IS FOR DRAGGABLE AND PICKUPABLE ITEMS



AND THE **ONLYGRAB** TAG IS FOR CRATES OR FOR ITEMS THAT CAN BE ONLY DRAGGABLE

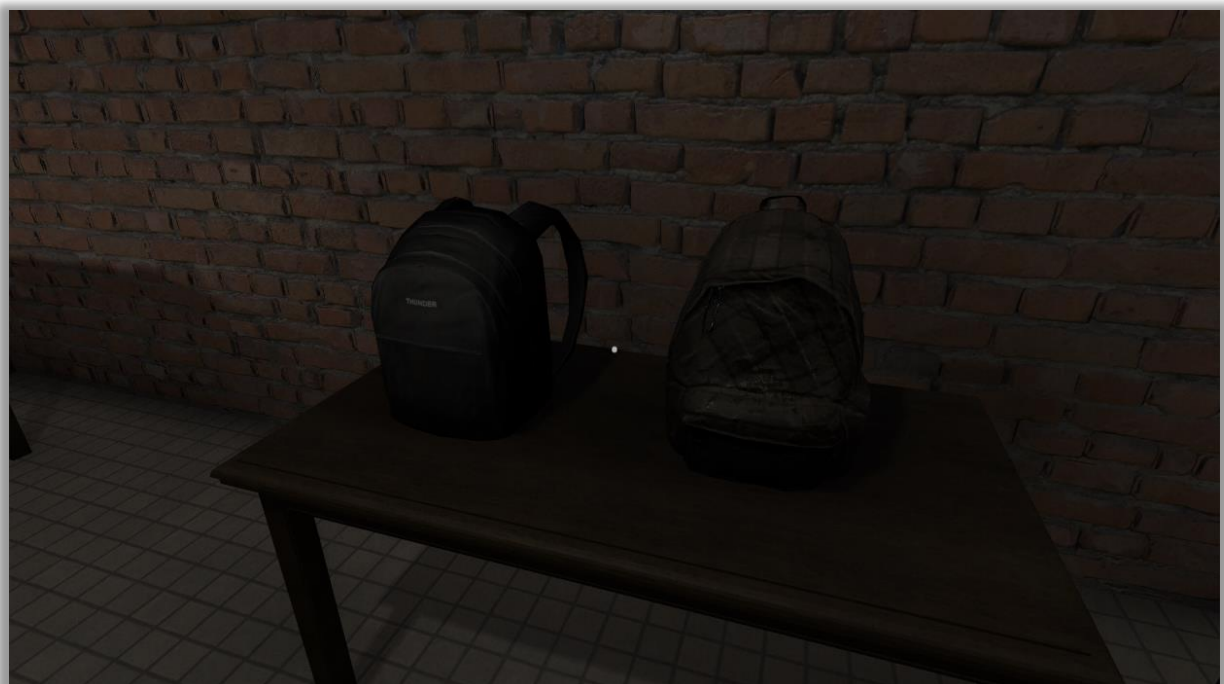


INVENTORY

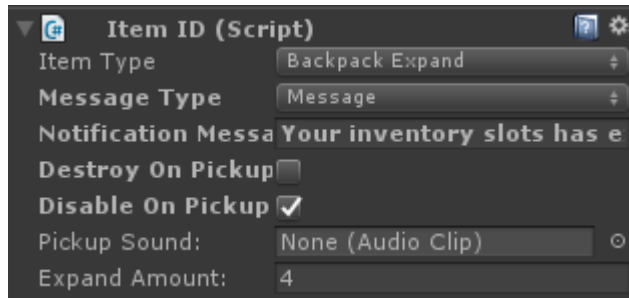


- YOU CAN SHOW INVENTORY MENU BY PRESSING **TAB** BUTTON
- TO CHANGE INVENTORY SHOW BUTTON YOU NEED TO GO TO THE **MAINMENU** AND CHANGE **INVENTORY DEFAULT BUTTON** IN **REBINADBLE INPUT SCRIPT** AND RECREATE CONFIG FILE

BACKPACK PICKUP (INVENTORY EXPAND)



- WHEN YOU PICKUP BACKPACK THE INVENOTRY WILL EXPAND BY **ITEMID EXPAND AMOUNT**

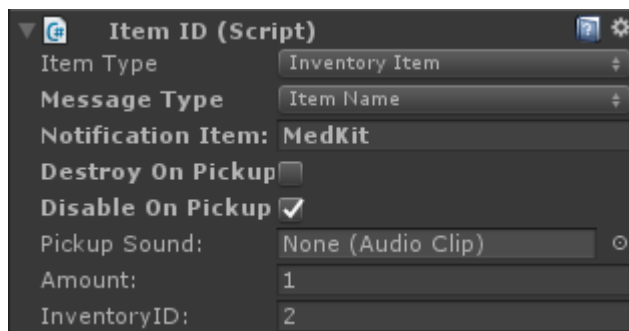


INVENTORY TWEAKS

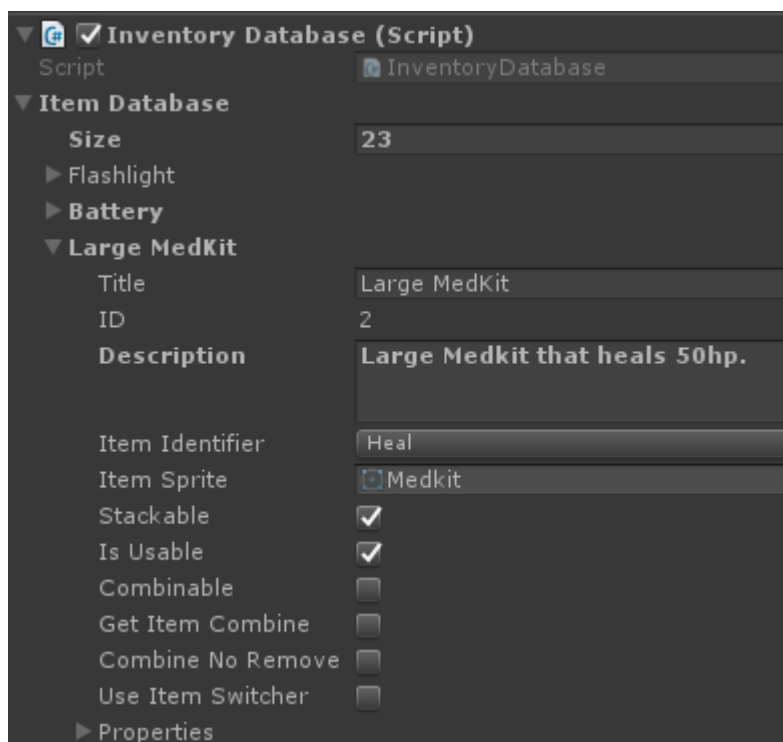
- YOU CAN ADD, REMOVE, USE, COMBINE ITEMS IN INVENTORY
- THE MAIN SCRIPT FOR INVENTORY PICKUPS IS **ItemID.cs**

INVENTORY ITEM PICKUP

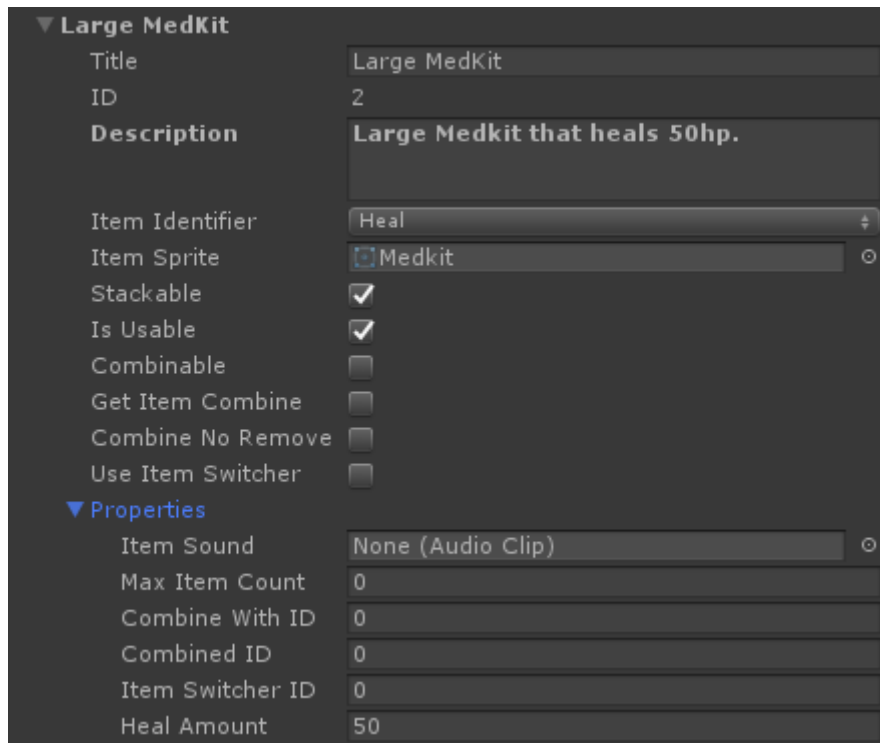
1. ADD **ItemID.cs** SCRIPT TO YOUR OBJECT AND CHANGE **ITEM TYPE** TO **INVENTORY ITEM**



2. THEN YOU NEED TO ADD YOUR ITEM TO **INVENTORY DATABASE** IN **GAMEMANAGER** OBJECT



3. THEN YOU MUST WRITE TITE, LITTLE DESCRIPTION OF YOUR ITEM AND SET ITEM ICON
4. AFTER THAT YOU CAN SET SOME PROPERTIES OF YOUR ITEM IN MY CASE I SET **ITEM IDENTIFER TO HEAL** AND CHANGED **HEAL AMOUNT**



▼ Large MedKit

Title Large MedKit

ID 2

Description Large Medkit that heals 50hp.

Item Identifier Heal

Item Sprite Medkit

Stackable ☒

Is Usable ☒

Combinable ☐

Get Item Combine ☐

Combine No Remove ☐

Use Item Switcher ☐

▼ Properties

Item Sound None (Audio Clip)

Max Item Count 0

Combine With ID 0

Combined ID 0

Item Switcher ID 0

Heal Amount 50

- ITEM ID IS DISPLAYED IN **INVENTORY DATABASE SCRIPT** SO YOU CAN EASY DETERMINE WHICH ID HAVE YOUR ITEM



COMBINABLE ITEM

- THE ONLY THING WHAT YOU NEED TO DO IS SET SOME PROPERTIES IN INVENTORY DATABASE TO MAKE ITEM COMBINABLE

▼ Herb

Title: Herb

ID: 13

Description: A plant containing phytochemicals that help to

Item Identifier: Heal

Item Sprite: Herb

Stackable: ☒

Is Usable: ☒

Combinable: ☒

Get Item Combine: ☒

Combine No Remove: ☐

Use Item Switcher: ☐

▼ Properties

Item Sound: None (Audio Clip)

Max Item Count: 0

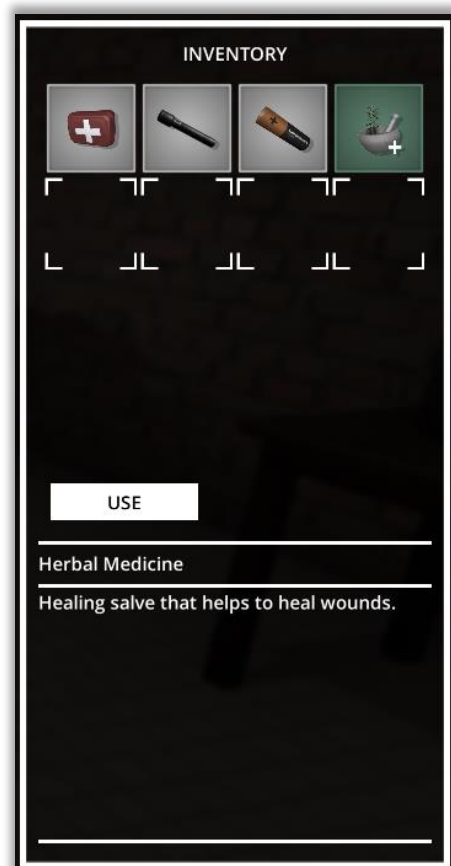
Combine With ID: 12

Combined ID: 14

Item Switcher ID: 0

Heal Amount: 10

- YOU NEED TO TICK **COMBINABLE** BOOL, SET **COMBINE WITH ID** AND **COMBINED ID** IN PROPERTIES SECTION TO TELL WHAT ITEM CAN BE COMBINED WITH THIS ITEM AND WHAT ITEM YOU GET AFTER COMBINE



INVENTORY WEAPONS AND BULLETS

- IF YOU WANT TO ADD NEW WEAPON YOU MUST ADD WEAPON ITEM TO INVENTORY DATABASE, **SET ITEM IDENTIFIER TO WEAPON** AND SET **ITEM SWITCHER ID**

▼ Glock18

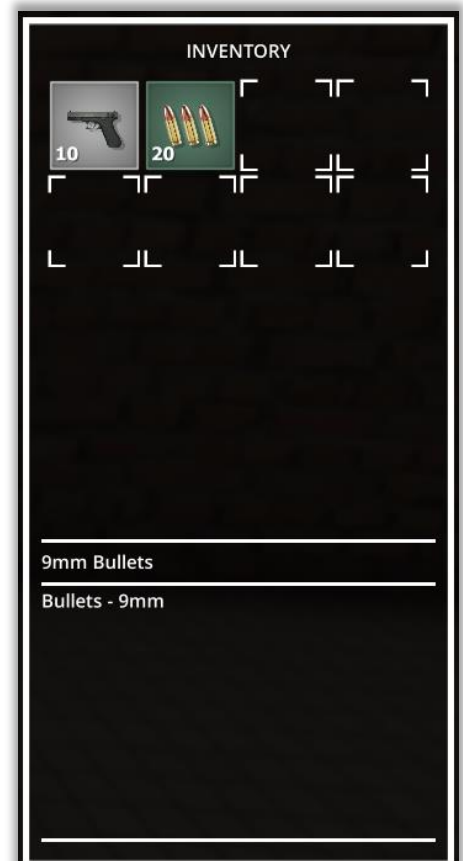
Title	Glock18
ID	21
Description	Glock18 Pistol
Item Identifier	Weapon
Item Sprite	Glock18
Stackable	<input type="checkbox"/>
Is Usable	<input checked="" type="checkbox"/>
Combinable	<input type="checkbox"/>
Get Item Combine	<input type="checkbox"/>
Combine No Remove	<input type="checkbox"/>
Use Item Switcher	<input checked="" type="checkbox"/>
▼ Properties	
Item Sound	None (Audio Clip)
Max Item Count	0
Combine With ID	0
Combined ID	0
Item Switcher ID	2
Heal Amount	0



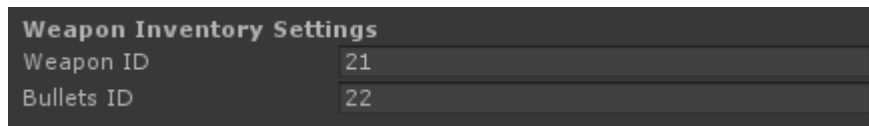
- AFTER THAT YOU NEED TO ADD WEAPON BULLETS AND SET **ITEM IDENTIFIER TO BULLETS**

▼ 9mm Bullets

Title	9mm Bullets
ID	22
Description	Bullets - 9mm
Item Identifier	Bullets
Item Sprite	9mm_bullets
Stackable	<input checked="" type="checkbox"/>
Is Usable	<input type="checkbox"/>
Combinable	<input type="checkbox"/>
Get Item Combine	<input type="checkbox"/>
Combine No Remove	<input type="checkbox"/>
Use Item Switcher	<input type="checkbox"/>
▼ Properties	
Item Sound	None (Audio Clip)
Max Item Count	120
Combine With ID	0
Combined ID	0
Item Switcher ID	0
Heal Amount	0

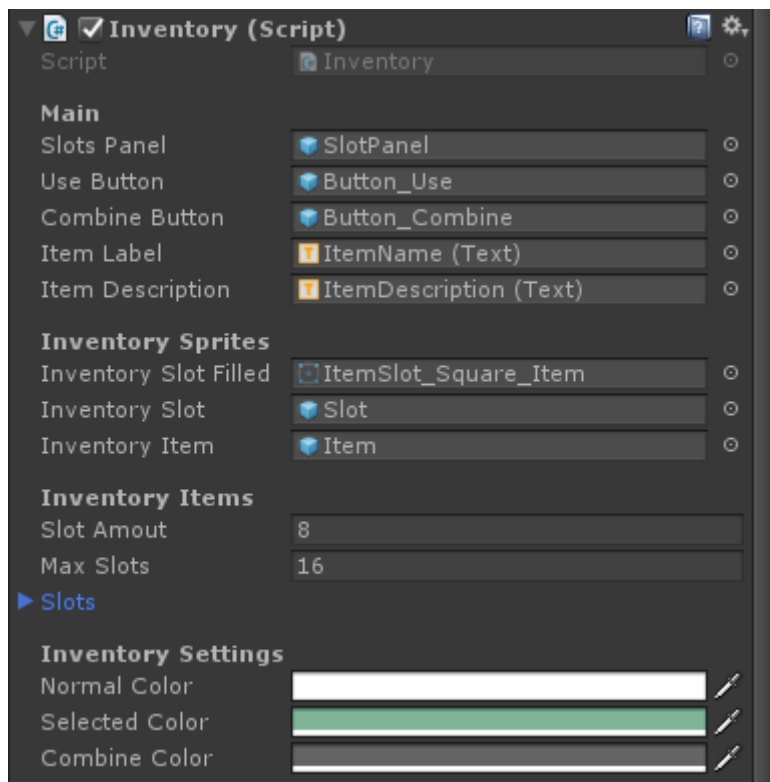


- LAST STEP IS GO TO **WEAPON CONTROLLER** AND SET **INVENTORY SETTINGS**



CHANGING INVENTORY SETTINGS

- IF YOU DON'T LIKE INVENTORY VISUAL YOU CAN CHANGE SOME SETTINGS IN **INVENTORY** SCRIPT THAT YOU CAN FIND IN **GAMEMANAGER** OBJECT

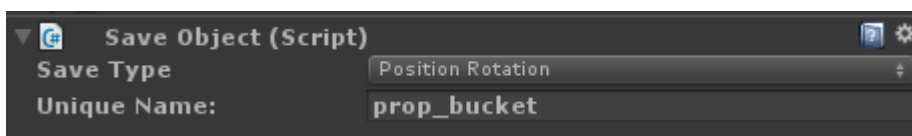


SAVE/LOAD MANAGER (WIP)

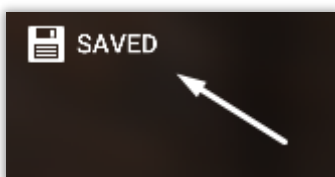
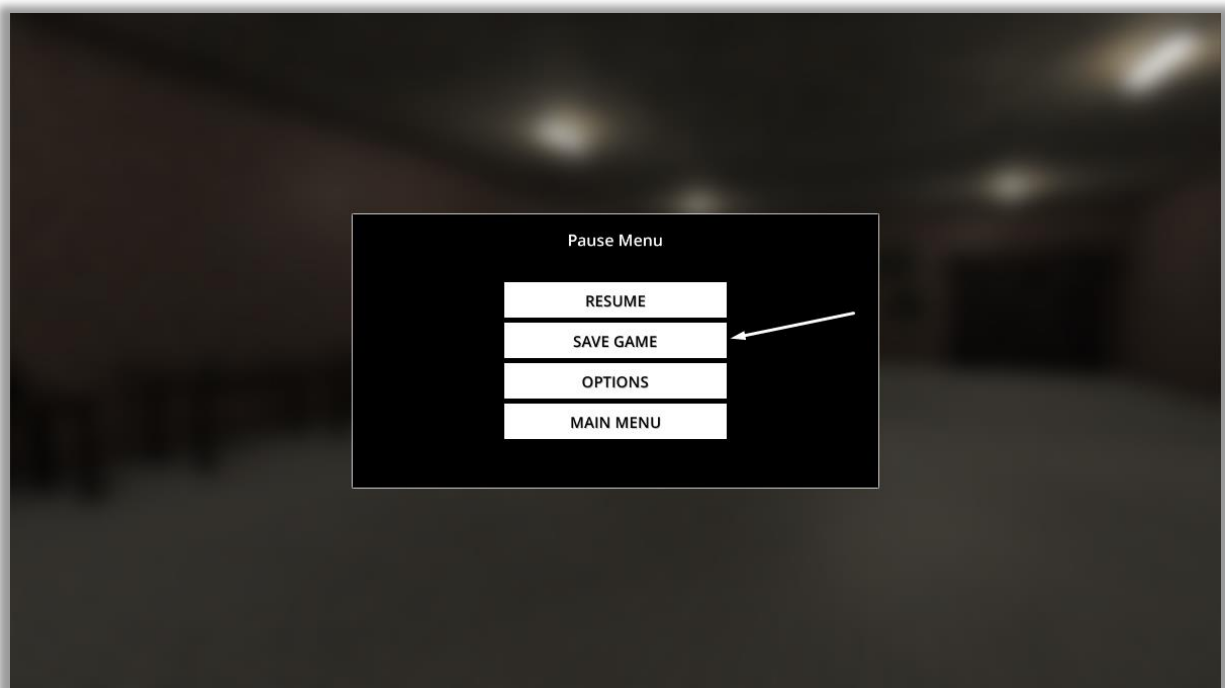
- SAVE AND LOAD MANAGER IS STILL IN WORK IN PROGRESS SO IF YOU FOUND PROBLEM PLEASE SEND ME BUG REPORT

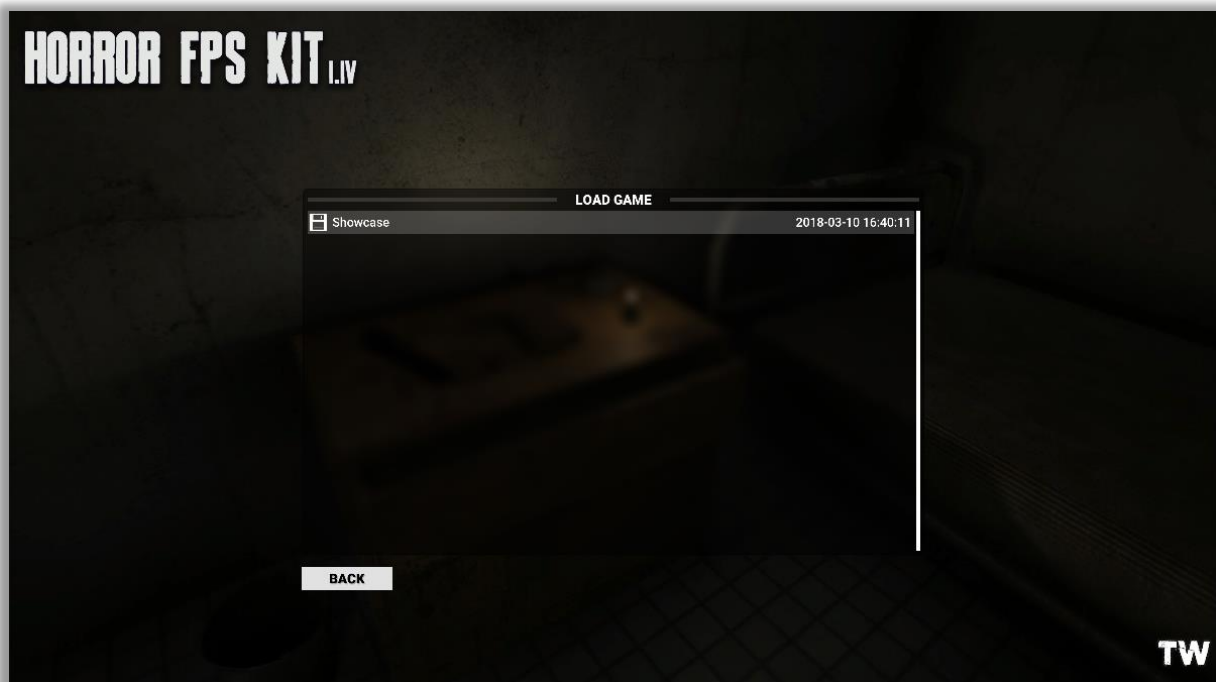
ADDING SAVEABLE OBJECTS

- IF YOU WANT SAVE OBJECT YOU MUST ADD **SaveObject.cs** SCRIPT TO OBJECT WHICH YOU WANT TO BE SAVED AND LOADED
- NEXT YOU NEED TO SET UNIQUE NAME
- **UNIQUE NAME CANNOT BE A DUPLICATE OF OTHER!**



- THEN IF EVERYTHING IS OK YOU CAN SAVE YOUR CURRENT GAME DATA LIKE (POSITIONS, ROTATIONS, SCRIPTS...)





- DEFAULT SAVED GAME LOCATIONS IS
\YOURGAME_DATA\DATA\SAVEDGAME
- SAVED GAME DATA IS NOT ENCRYPTED, FEATURE FOR ENCRYPTING DATA WILL BE ADDED IN FUTURE UPDATE

SAVING CUSTOM DATA

- IF YOU NEED SAVE YOUR OWN DATA THEN YOU NEED SIMPLE CODING
- **SAVE GAME HANDLER** IS LOCATED IN GAMEMANAGER GAMEOBJECT
- BEFORE SAVE YOU NEED TO **UPDATE GAME DATA**

OPEN **SAVE GAME HANDLER** AND LOCATE **SAVE SECTION**

TO UPDATE GAME DATA USE THIS SENTENCE:

```
saveManager.UpdateGameData("your_key", new List<string>()
{
    your_int.ToString(),
    your_float.ToString()
});
```

THEN TO **LOAD** YOUR CUSTOM GAME LOCATE **LOAD SECTION** AND USE THIS SENTENCE:

```
string[] yourArray = saveManager.DeserializeArray("your_key");
int your_int = int.Parse(yourArray[0]);
float your_float = float.Parse(yourArray[1]);
```

- **PAY ATTENTION TO THE SAVING ORDER (0,1,2,3...)**

SAVING CUSTOM ITEM SWITCHER DATA

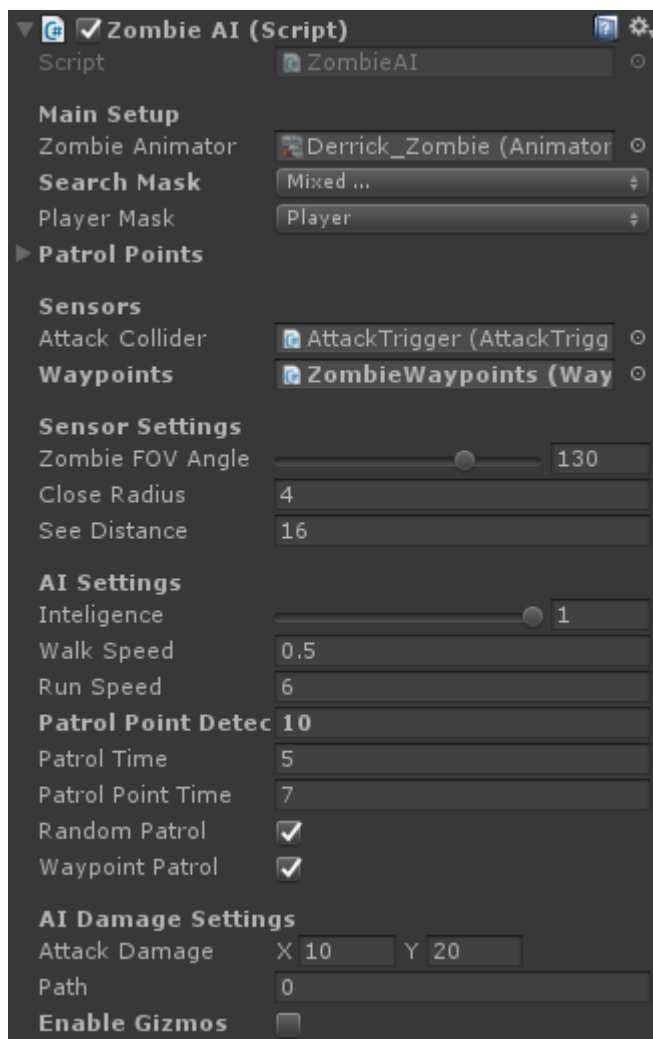
1. ADD **SaveHelper.cs** UNDER YOUR CUSTOM SCRIPT
2. **SAVE HELPER** MUST BE IN ITEM SWITCHER OBJECT
3. OPEN YOUR CUSTOM SCRIPT IN SCRIPT EDITOR
4. ADD THIS SENTENCES WITH YOUR OWN VALUES TO YOUR SCRIPT

```
public void SendValues()
{
    if (GetComponent<SaveHelper>())
    {
        GetComponent<SaveHelper>().SetValues(
            new List<string>() {
                Candle.transform.localScale.y.ToString()
            });
    }
}

public void SetSavedValues(List<string> values)
{
    Vector3 scale = Candle.transform.localScale;
    scale.y = float.Parse(values[0]);
    Candle.transform.localScale = scale;
}
```

- THIS WORKS SAME AS **SAVING CUSTOM DATA**
- SAVE HELPER SENDS MESSAGE **SendValues()** TO YOUR SCRIPT TO GET CUSTOM DATA FROM IT
- AND FOR LOAD **SAVE HELPER** SENDS **SetSavedValues(String List)**

AI SYSTEM



- IF YOU WANT SET NEW ZOMBIE NPC YOU MUST CREATE NEW ANIMATOR WITH SAME PROPERTIES AS HAVE MY PRE SET ZOMBIE ANIMATOR
- YOU CAN CHANGE ALL NEEDED AI SETTINGS IN **ZOMBIE AI** SCRIPT
- **INTELLIGENCE** SLIDER SETS MAIN ZOMBIE INTELLIGENCE SETTINGS AS IS (ATTRACTED STATE AND GO TO PATROL POINT STATE)
- PATROL POINT IS POINT WHERE ZOMBIE GO IF DISTANCE BETWEEN **LAST SEEN POSITION** AND **POTROL POINT POSITION** IS IN RANGE OF **PATROL POINT DETECT**
- TO CREATE NEW WAYPOINT YOU NEED JUST CREATE EMPTY GROUP GAMEOBJECT WITH **WaypointGroup.cs** SCRIPT AND THEN ADD EMPTY GAMEOBJECTS INSIDE THE GROUP (WAYPOINTS WILL BE SET AUTOMATICALLY)

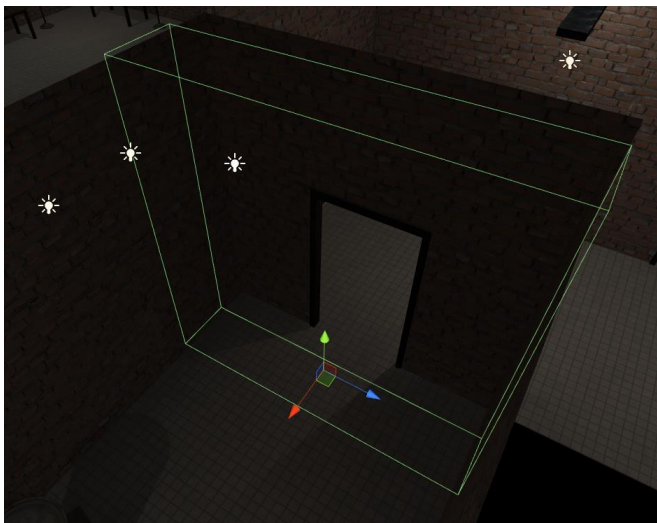
JUMPSCARES

TRIGGER ANIMATION

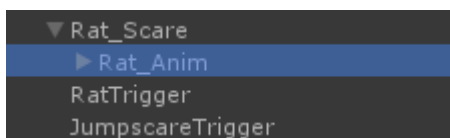
- USE THIS TYPE OF JUMPSCARE TO MAKE OBJECT OR CREATURE MOVE WHEN YOU GO TO THE TRIGGER



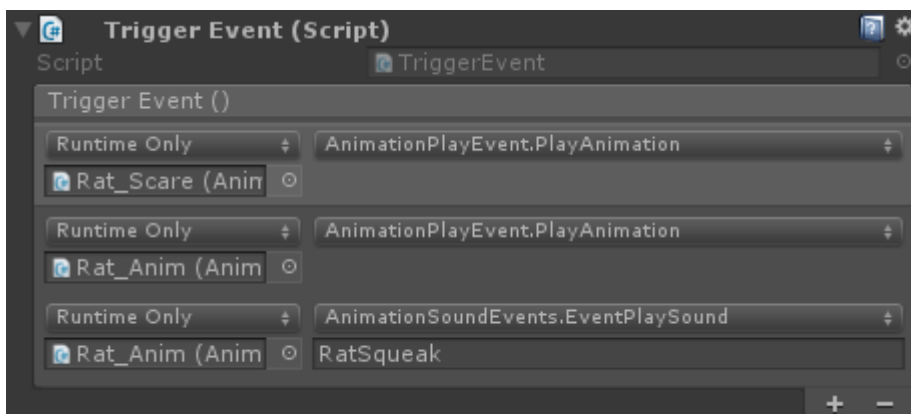
1. FIRST YOU MUST CREATE TRIGGER



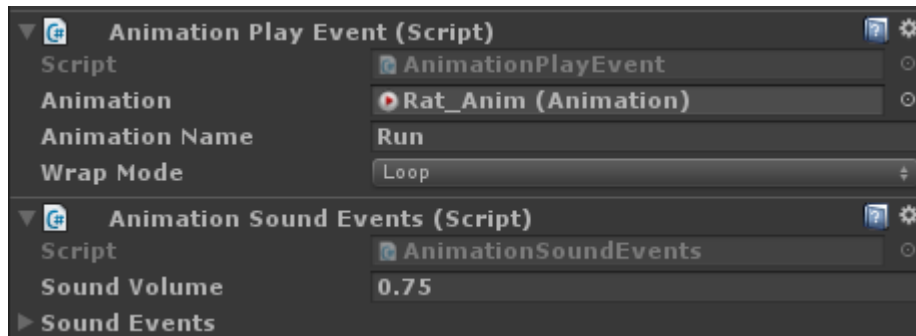
2. THEN CREATE EMPTY GAMEOBJECT AND MOVE CREATURE TO IT



3. ADD **TriggerEvent.cs** TO TRIGGER GAMEOBJECT

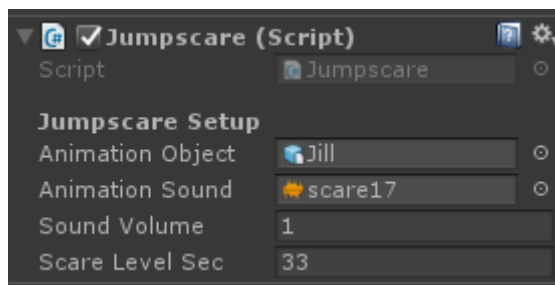


4. CREATE CREATURE MOVE ANIMATION
5. THEN ADD EVENT LISTENERS TO SET WHAT HAPPEND IF YOU GO TO TRIGGER IN MY CASE I ADDED ANIMATION AND SOUND PLAY EVENT

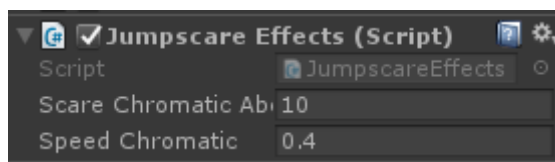


JUMPSCARE ANIMATION

- JUMPS CARE ANIMATION IS SAME AS TRIGGER ANIMATION BUT WITH SPECIAL SCARE EFFECTS
- YOU CAN CREATE IT WITH SAME STEPS AS TRIGGER ANIMATION BUT INSTEAD OF ADDING **TriggerEvent.cs** ADD **Jumpscare.cs**



- YOU CAN SET HOW LONG WILL BE PLAYER SCARED BY SETTING **SCARE LEVEL SEC** IN SECONDS (SCARED BREATHING)
- THIS SCRIPT IS LINKED WITH PLAYER **JumpscareEffects.cs** SCRIPT IN MOUSELOOK GAMEOBJECT
- **JUMPS CARE EFFECTS** CONTROL CAMERA SHAKE, SCARED BREATH AND CHROMATIC ABERATION

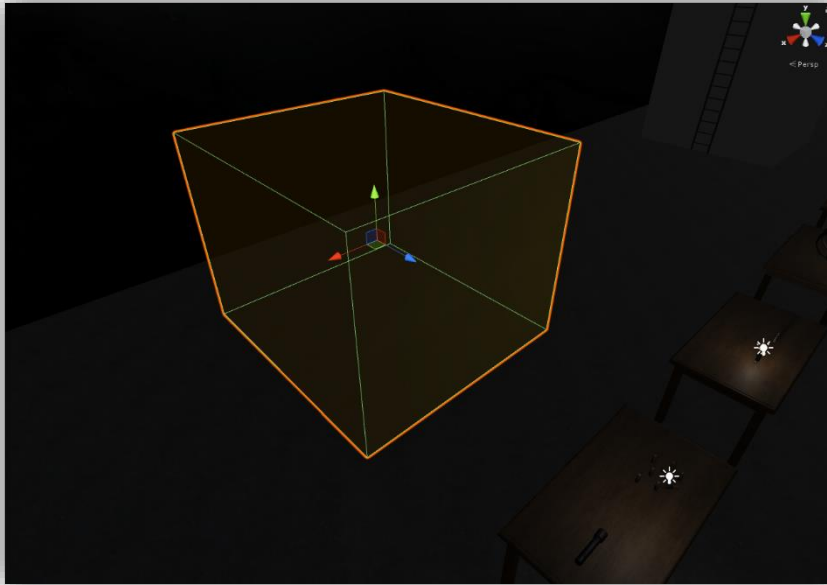


- IF YOU HAVE PROBLEMS WITH CREATING JUMPS CARE ANIMATION YOU CAN GO TO MY YOUTUBE CHANNEL AND WATCH MY JUMPS CARE TUTORIAL: [JUMPS CARE TUTORIAL](#)

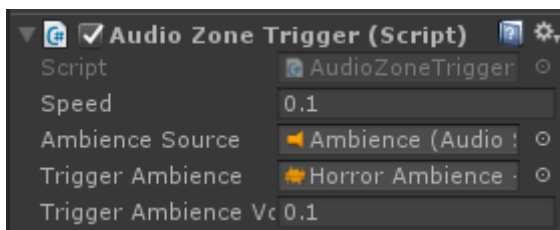
AUDIO ZONE TRIGGER

- THIS IS GOOD FOR GOING TO OTHER ROOMS TO CHANGE AMBIENCE AUDIO

1. FIRST CREATE TRIGGER



2. AND ADD **AudioZoneTrigger.cs** TO TRIGGER OBJECT

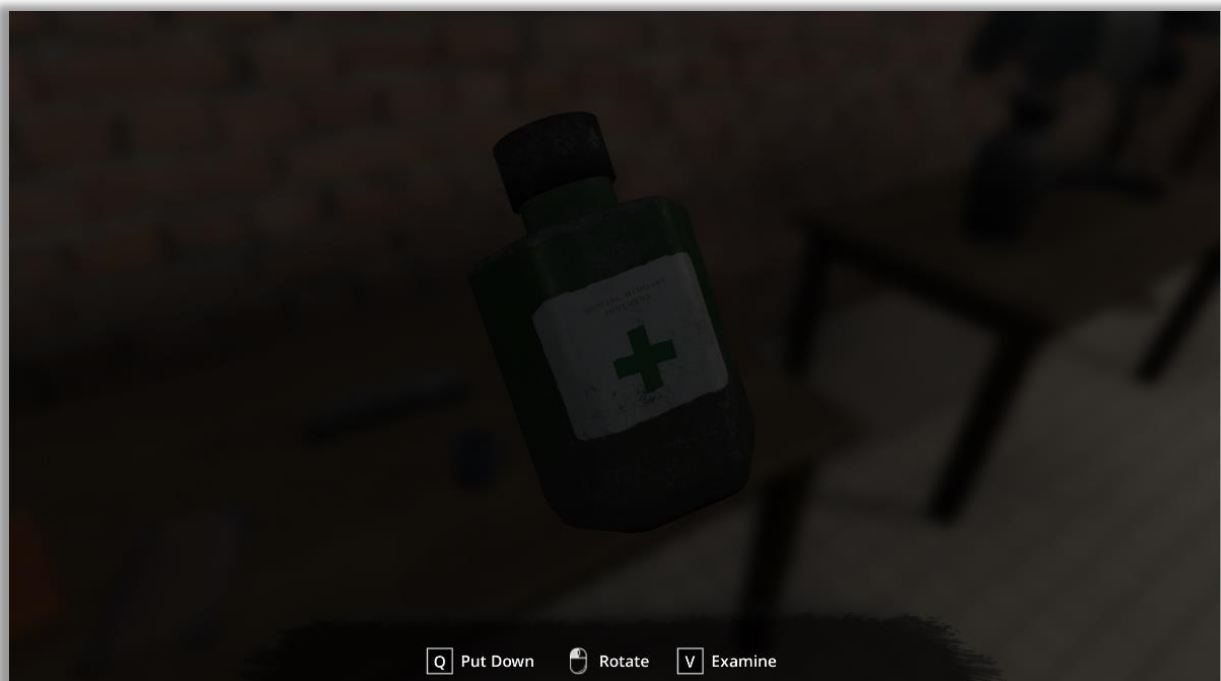
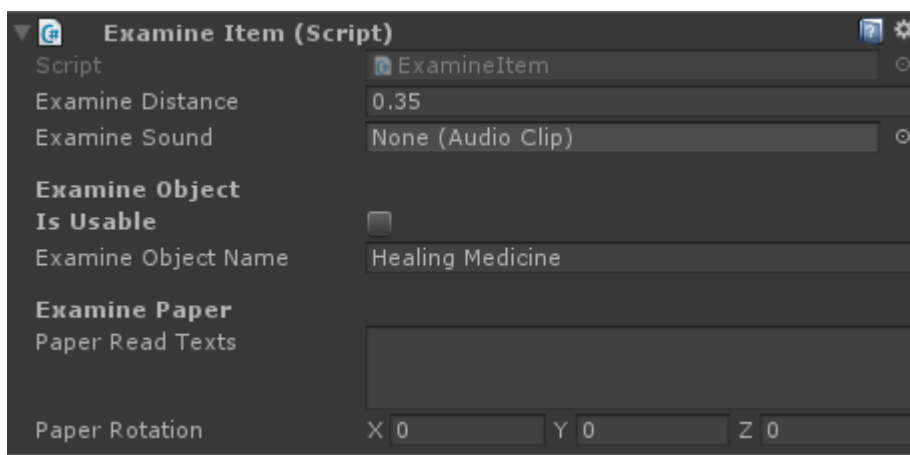


- YOU CAN CHANGE TRANSITION SPEED, AMBIENCE SOUND AND AMBIENCE VOLUME

IF YOU WANT TO CHANGE STARTING AMBIENCE GO TO **FPSPLAYER -> SOUND EFFECTS -> AMBIENCE** AND DRAG YOUR AMBIENCE TO AUDIO SOURCE

ADDING EXAMINE OBJECTS

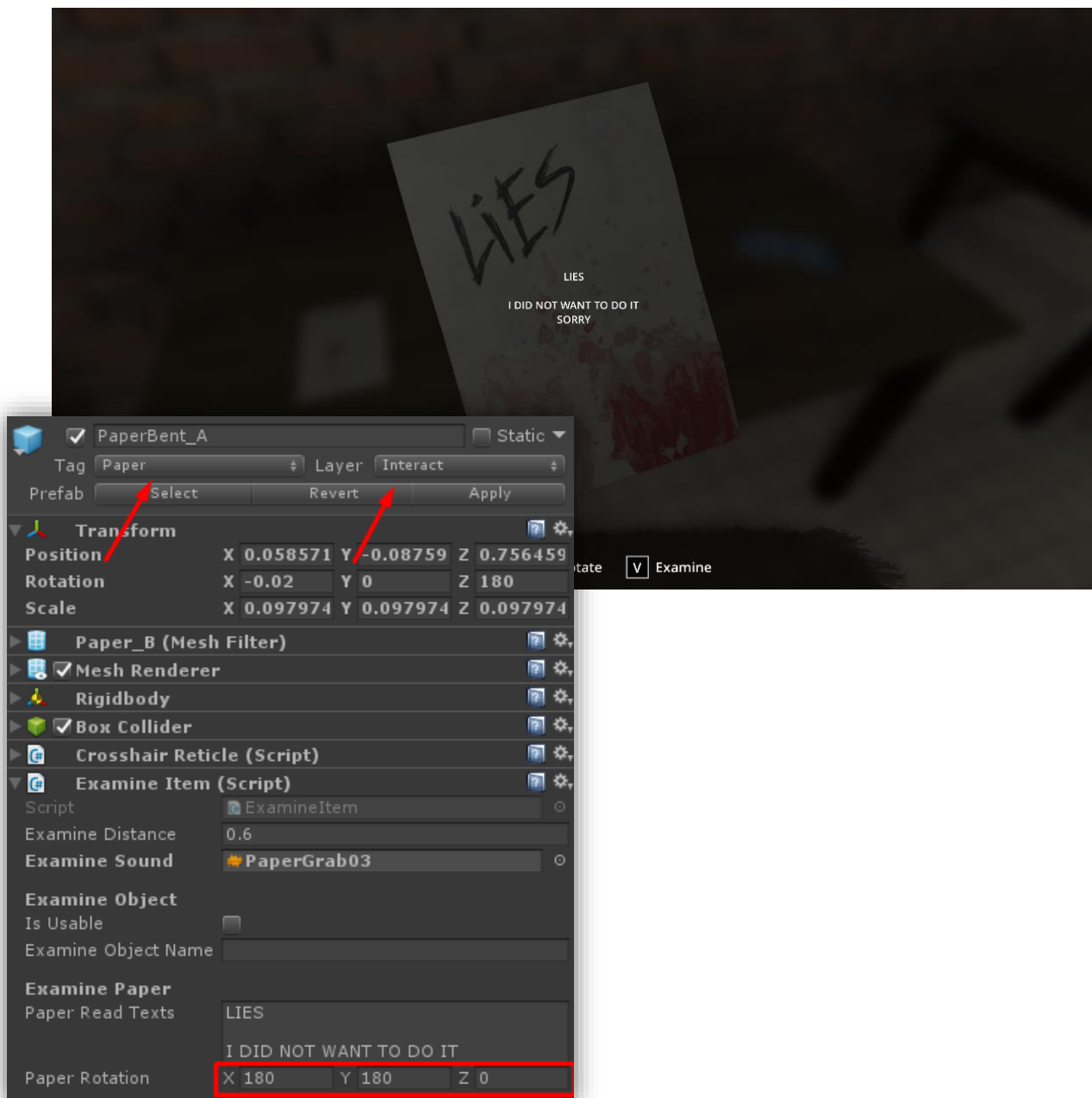
- **OBJECT MUST HAVE RIGIDBODY AND COLLIDER**
 - YOU CAN ROTATE AND EXAMINE OBJECT
1. CHANGE EXAMINE OBJECT TAG TO "**Examine**" AND LAYER TO "**Interact**"
 2. THEN ADD **ExamineItem.cs** TO EXAMINE OBJECT
- YOU CAN CHANGE OBJECT NAME IN **EXAMINE OBJECT NAME**
 - AND YOU CAN ADJUST OBJECT **EXAMINE DISTANCE**
 - IF YOU WANT MAKE OBJECT EXAMINABLE AND USABLE TICK **IS USABLE** BOOL



ADDING NEW PAPERS

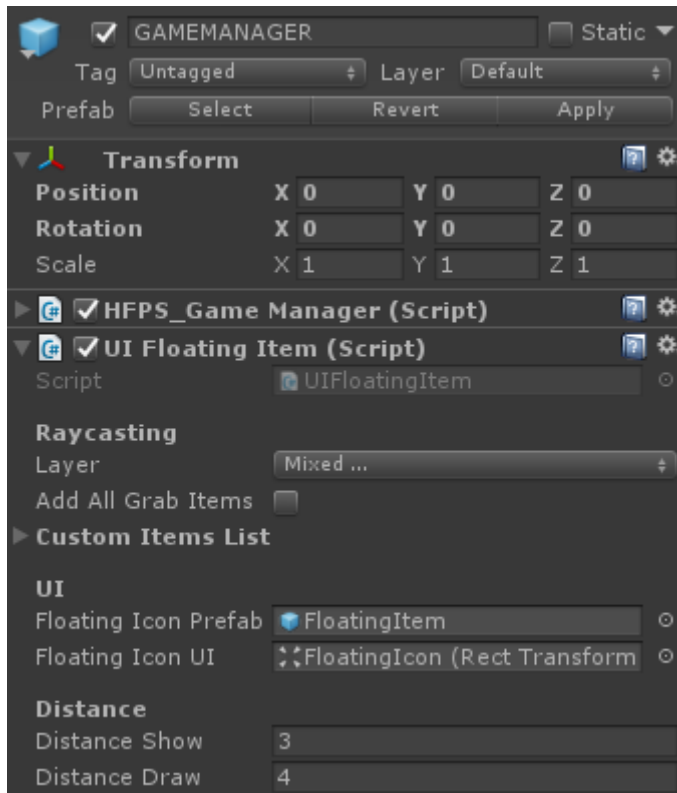
THE PAPER PICKUP SYSTEM WORKS LIKE PAPER EXAMINE METHOD (LIKE IN OTHER POPULAR HORROR GAMES)

- **OBJECT MUST HAVE RIGIDBODY AND COLLIDER**
 - YOU CAN ROTATE AND READ PAPERS
1. JUST DRAG AND DROP **ExamineItem.cs** SCRIPT TO PAPER
 2. THEN CHANGE PAPER TAG TO "**Paper**" and LAYER TO "**Interact**"
 3. MAIN PART IS SET **PAPER ROTATION** TO CORRECT ROTATION WHEN YOU EXAMINE PAPER
 4. THEN YOU CAN CHANGE PAPER READ TEXT AND DISTANCE GRAB



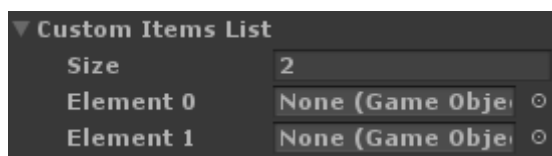
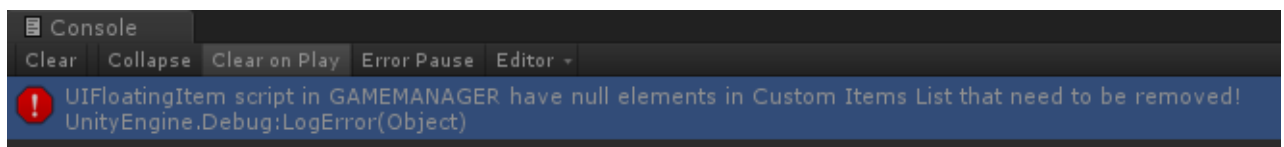
FLOATING ICON

- THIS SCRIPT IS LOCATED IN GAMEMANAGER
- YOU CAN ADD ALL GRABBABLE OBJECT BY CHECKING **ADD ALL GRAB ITEMS**
- IF YOU WANT ADD CUSTOM ITEMS JUST DRAG OBJECT TO **CUSTOM ITEMS LIST**



- IF YOU WANT CHANGE FLOATING ICON CLICK ON **FloatingItem** PREFAB AND CHANGE TO YOUR OWN ICON

IF YOU HAVE **ERROR** ON CONSOLE FROM THIS SCRIPT, IS BECAUSE THE CUSTOM ITEMS LIST HAS NULL ELEMENTS (SET SIZE TO **0** TO FIX IT)



ADDING NEW FOOTSTEPS

REMEMBER IN **FOOTSTEPS.CS** SCRIPT FOOTSTEPS **ELEMENT 0** IS ALWAYS **UNTAGGED** AND **ELEMENT 1** IS **LADDER**

1. JUST ADD NEW ELEMENT AND CHANGE **GROUND TAG** TO YOUR NEW FOOTSTEP GROUND TYPE NAME
2. OPEN FOOTSTEP DROPDOWN AND ADD HOW MUCH FOOSTEPS YOU WANT

SHOWING CUSTOM NOTIFICATIONS

IF YOU WANT TO SHOW MESSAGE WHEN YOU PICKUP OBJECT OR IF YOU WANT TO SHOW INFO MESSAGE OR WARNING MESSAGE CONNECT YOUR SCRIPT WITH **GAMEMANAGER SCRIPT**

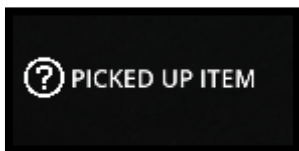
SIMPLE MESSAGE

```
uiManager.AddMessage ("Simple Message");
```



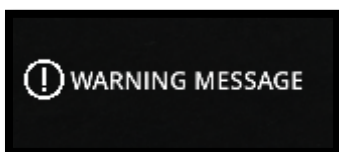
PICKUP MESSAGE

```
uiManager.AddPickupMessage ("Item");
```



WARNING MESSAGE

```
uiManager.WarningMessage ("Warning Message");
```



SHOWING CUSTOM HINT MESSAGE

IF YOU WANT TO SHOW CUSTOM HINT MESSAGE WHEN YOU GO TO TRIGGER USE THIS

1. FISRT LINK YOUR SCRIPT WITH **GAMEMANAGER SCRIPT**
2. IF YOU WANT TO SHOW HINT MESSAGE USE:

```
uiManager.ShowHint ("CustomHint");
```

BUG, ERROR REPORT

IF YOU FOUND BUG OR ERROR PLEASE SEND ME MESSAGE TO THIS EMAIL ADDRESS: thunderwiregames@gmail.com

OR VISIT MY WEBSITE [CUSTOMER SUPPORT PAGE](#) OR [CONTACT PAGE](#)

CREDITS

THIS KIT IS DEVELOPED AND DESIGNED BY THUNDERWIRE GAMES© ALL RIGHTS RESERVED

ALMOST ALL ASSETS ARE CREATED BY THUNDERWIRE GAMES EXPECT ONES DOWNLOADED WITH ROYALTY FREE LICENSE

