DOCUMENTATION

THANKS FOR BUYING HORROR FPS KIT!

If you like my assets please visit 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 AN ADVANCED AND EASY-TO-USE HORROR GAME TEMPLATE WITH MANY FEATURES ESSENTIAL TO CREATING YOUR OWN HORROR GAME, INCLUDING GAMEPLAY FEATURES SEEN IN AAA HORROR GAMES OF THE LAST DECADE. IT CONTAINS A LOT OF READY-TO-USE ASSETS, JUST DRAG AND DROP THEM INTO A SCENE.

SUMMARY

ALL FEATURES (VERSION 1.42)

PLAYER FUNCTIONS

- PLAYER CONTROLLER (WALK, RUN, JUMP, CROUCH, CRAWL, 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, DROP)
- WALL DETECT SYSTEM (HIDE WEAPON)
- WEAPONS (GLOCK18)
- FALL DAMAGE
- PLAYER LEAN (WALL DETECTION)
- ZOOM EFFECT
- INTERACT SYSTEM
- UI CROSSHAIR

OBJECT PICKUPS

- CUSTOM OBJECT PICKUP SCRIPT
- FLASHLIGHT PICKUP (BATTERIES)
- CANDLE PICKUP
- OIL LAMP 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 TYPES (NORMAL, LOCKED, JAMMED – CAN UNJAM)
- DRAGGABLE OBJECTS (DOOR. DRAWER, LEVER)
- KEYPAD

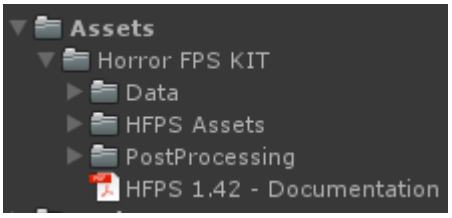
MORE FUNCTIONS

- SAVE/LOAD SYSTEM (SAVING AND LOADING SCENE DATA, ENCRYPT SAVE)
- SAVING PLAYER DATA BETWEEN SCENES
- CONFIG MANAGER (SAVE AND READ YOU OWN .CFG FILES)
- HELPERS (INPUT HELPER, TYPE PARSER, RANDOM GENERATOR)
- LOADING SCREEN (BACKGROUND, TIPS, SCENE NAME)
- REBINDABLE INPUT MANAGER
- AI ZOMBIE SYSTEM (WALK, RUN, ATTACK, PATROL, ATTRACT)
- WATER BUOYANCY
- UI MENUS (MAIN MENU, LOAD GAME MENU, PAUSE MENU, DEAD 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, COLLECTABLE OBJECTS, AND MUCH MORE..
- AMBIENCE SOUND CHANGE
- HINT MANAGER
- PICKUP NOTIFICATIONS (ITEM NAME, MESSAGE, HINT)
- FULLY FUNCTIONAL UI

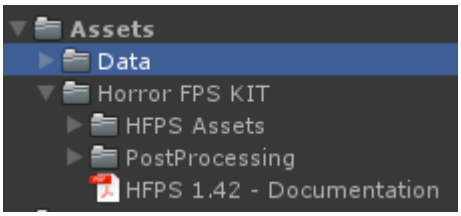
PROJECT SETUP

IS RECOMMENDED IMPORT HFPSKIT TO EMPTY PROJECT!

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

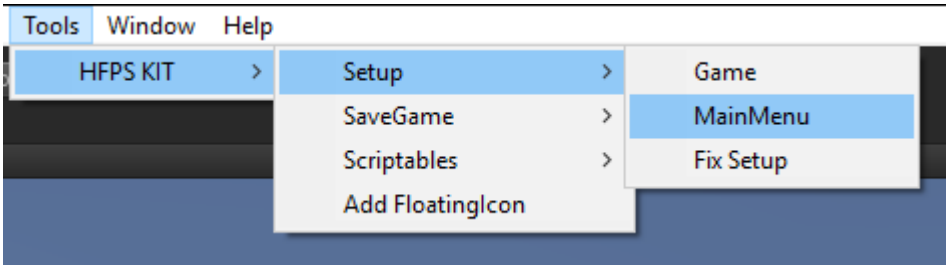


2. MOVE **DATA** FOLDER TO YOUR PROJECT ASSET FOLDER



MAIN MENU SCENE SETUP

1. OPEN NEW EMPTY SCENE
2. GO TO **TOOLS - > HFPS KIT -> SETUP -> MAINMENU**

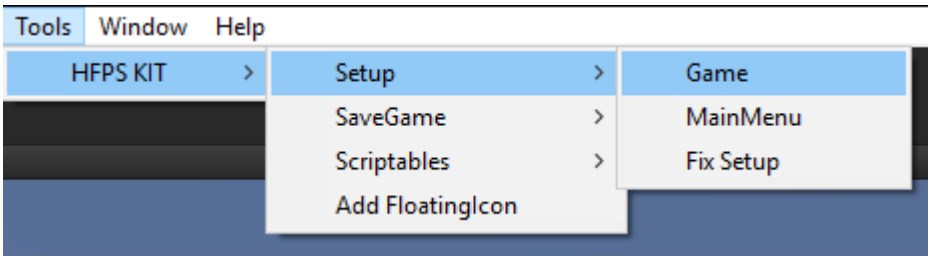


3. NOW CHECK IF YOU HAVE **MENUMANAGER** OBJECT IN YOUR SCENE

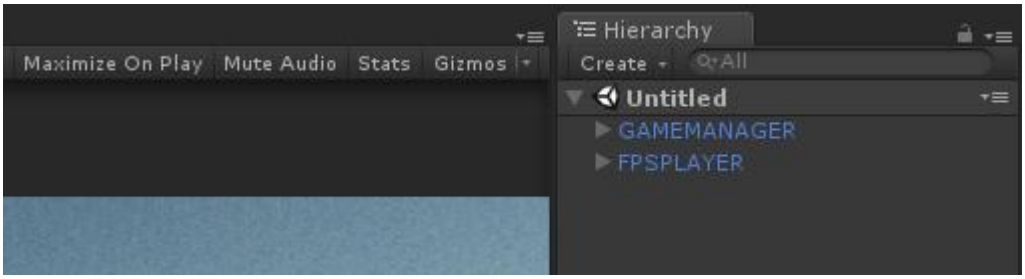
4. **DONE,** NOW YOU CAN PLAY SCENE WITH MAINMENU TEMPLATE

GAME SCENE SETUP

1. OPEN NEW EMPTY SCENE
2. GO TO **TOOLS - > HFPS KIT -> SETUP -> GAME**

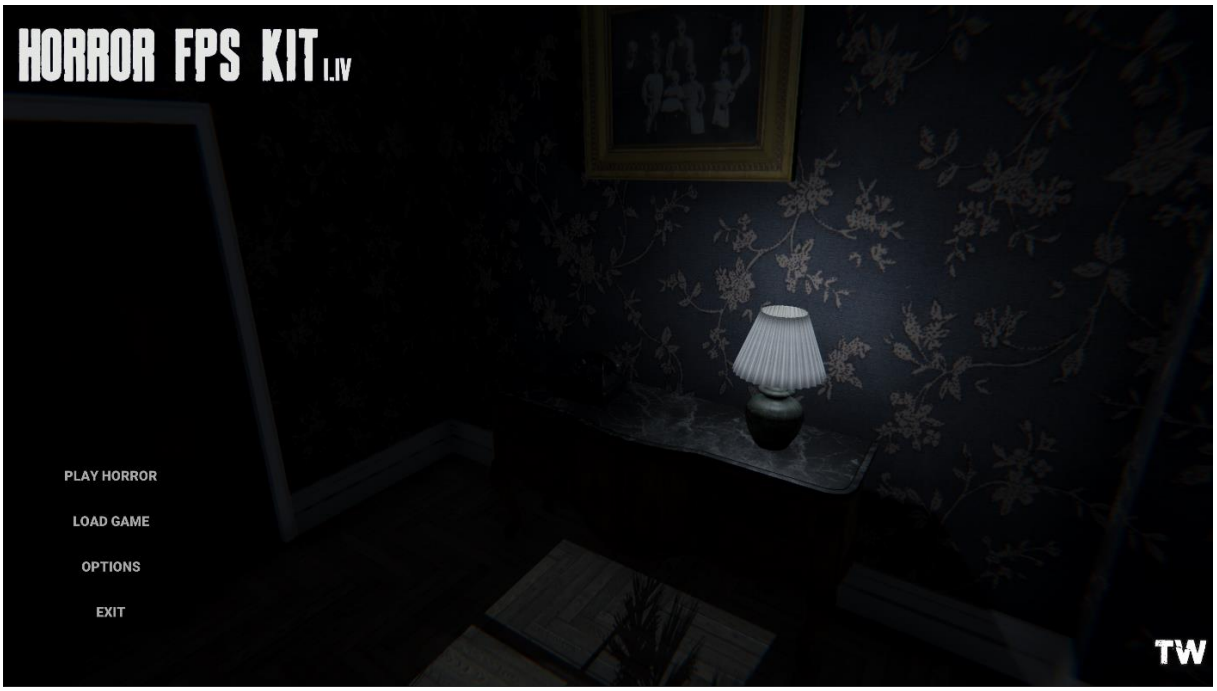


3. CHECK IF YOUR SCENE HAVE **FPSPLAYER** AND **GAMEMANAGER**



4. MOVE FPSPLAYER TO FLOOR

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

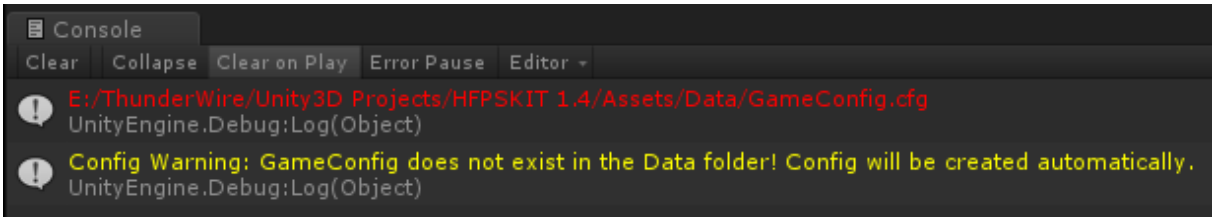


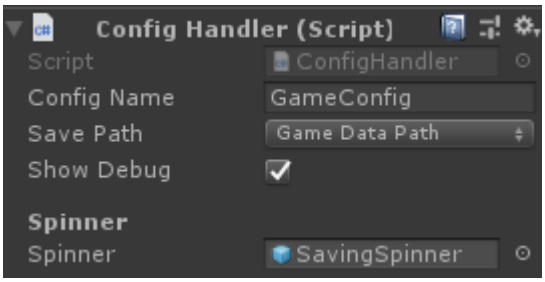
HOW TO SETUP GAMECONFIG

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

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

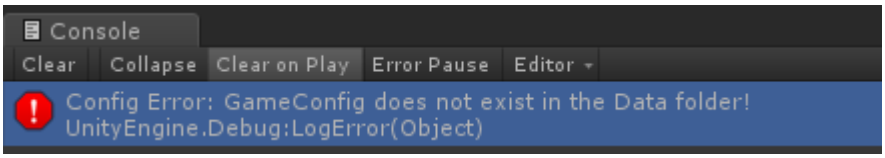
MAIN MENU:





CONFIG HANDLER IS MAIN SCRIPT WHICH HANDLES ALL CONFIG ACTIONS

IF **CONFIG HANDLER** DOES NOT FIND CONFIG FILE YOU WILL GET THESE ERROR:



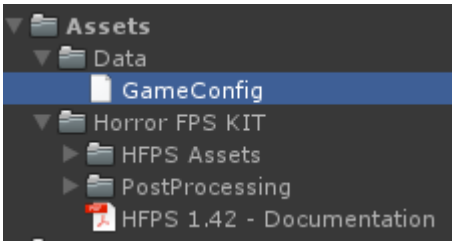
AND PLAYER WILL NOT BE ABLE TO MOVE

IN THIS CASE YOU WILL NEED TO PLAY SCENE AGAIN WITH RECREATED CONFIG FILE WHICH CONTAINS DEFAULT INPUTS

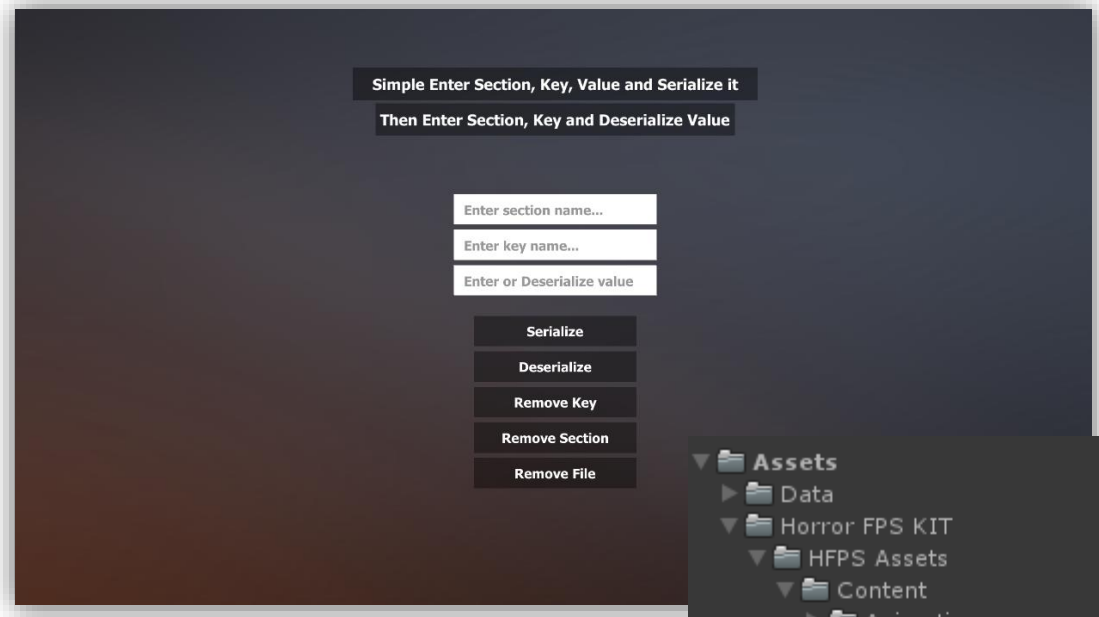
CONFIG MANAGER

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

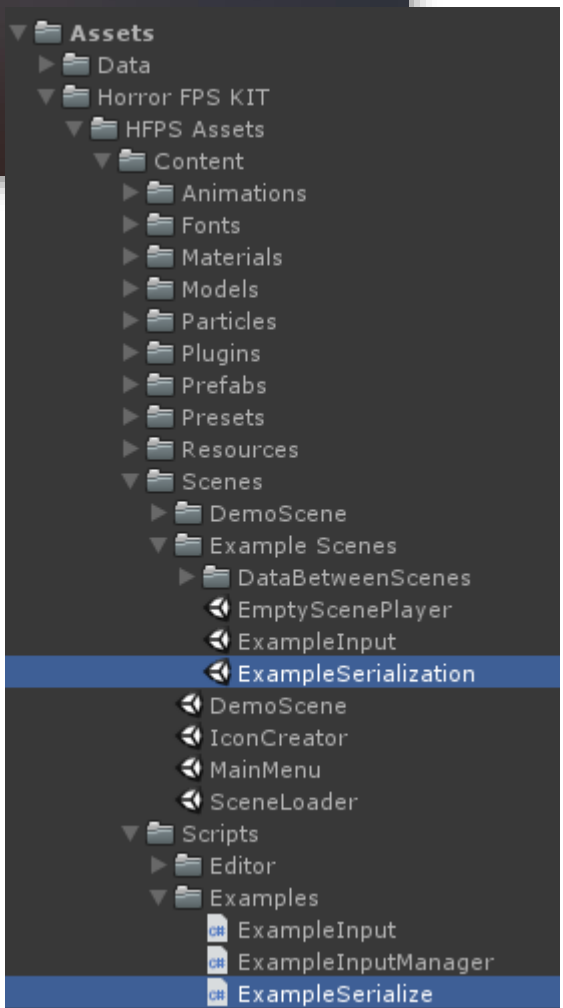
- ALL CONFIG FILES IS STORED INSIDE PROJECT OR INSIDE EXPORTED GAME TO FOLDER NAMED **DATA**



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 UnityEngine.UI;
using ThunderWire.Configuration;
```

2. THEN YOU CAN SETUP CONFIG FOLDER AND NAME

```
ConfigManager.SetFilePath(FilePath.GameDataPath);
ConfigManager.SetFilename("Config");
```

```
void Start () {
    ConfigManager.SetFilePath(FilePath.GameDataPath);
    ConfigManager.SetFilename("Config");
}
```

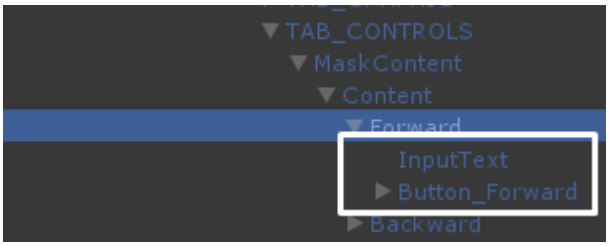
CONFIG MANAGER FUNCTIONS

ConfigManager.EnableDebugging(bool); - ENABLE CONFIG DEBUGGING
ConfigManager.SetFilename(string); - SET CONFIG FILENAME
ConfigManager.SetFilePath(FilePath); - SET CONFIG PATH
ConfigManager.Serialize("Section", "Key", "Value"); - SERIALIZE TO CONFIG FILE
ConfigManager.Deserialize("Section", "Key"); - DESERIALIZE FROM CONFIG
ConfigManager.Deserialize<TYPE>("Section", "Key"); - DESERIALIZE TO TYPE
ConfigManager.ContainsSection("Section"); - CHECK IF CONFIG HAVE SECTION
ConfigManager.ContainsSectionKey("Section", "Key",); - CHECK IF SECTION HAVE KEY
ConfigManager.ContainsKeyValue("Section", "Key", "Value",); - CHECK IF KEY HAVE VALUE
ConfigManager.RemoveSectionKey ("Section", "Key"); - REMOVE KEY FROM SECTION
ConfigManager.RemoveSection ("Section"); - REMOVE SECTION FROM CONFIG FILE
ConfigManager.GetSectionKeys ("Section"); - GET COUNT OF SECTION KEYS
ConfigManager.ExistFile ("ConfigFolder", "ConfigName "); - CHECK IF CONFIG EXIST
ConfigManager.ExistFileInFolder ("File", "Folder "); - CHECK IF CONFIG EXIST IN FOLDER
ConfigManager.ExistFileWithPath("FullPath", "File"); - CHECK IF CONFIG EXIST IN PATH
ConfigManager.RemoveFile(FilePath, "File"); - REMOVE FILE FROM FILEPATH
ConfigManager.DuplicateFile(FilePath, "File", "Name"); - DUPLICATE FILE IN FILEPATH
ConfigManager.GetFolderPath(FilePath); - GET FOLDER PATH
ConfigManager.GetFilepathRoot(FilePath); - GET FOLDER PATH ROOT
ConfigManager.GetFileAndPath(FilePath, "File"); - GET FULLPATH TO THE FILE
ConfigManager.GetFileAndPathFolder(FilePath, "Folder", "File"); - GET FILE IN FILEPATH AND FOLDER

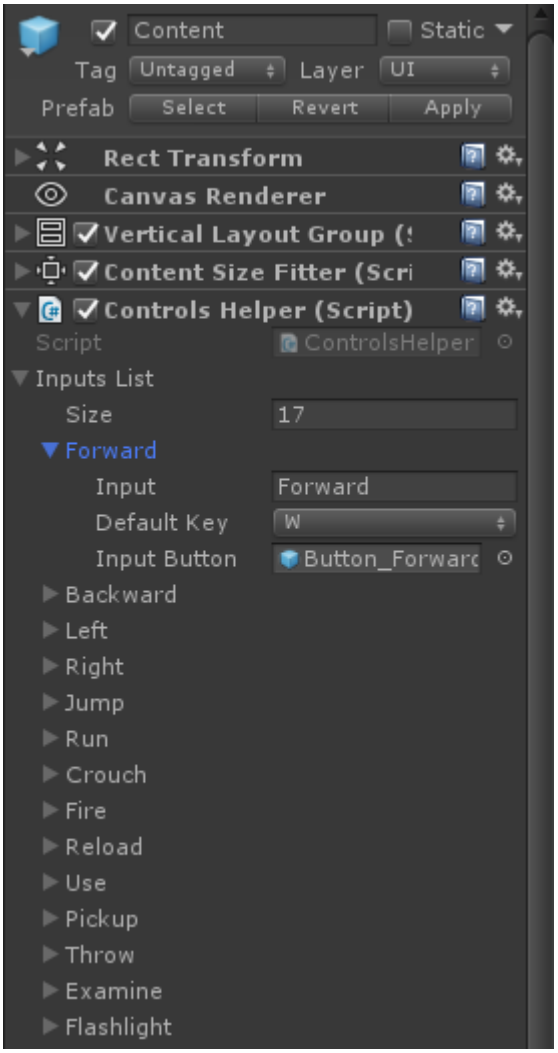
INPUT MANAGER (REBINDABLE INPUT)

HOW TO ADD NEW INPUT

1. GO TO **HFPS ASSETS -> CONTENT -> SCENES** AND OPEN FOR EXAMPLE **MAINMENU** SCENE
2. SELECT **MENUMANAGER**
3. NEXT YOU NEED TO CREATE OR EDIT INPUTS IN **TAB_CONTROLS** GAMEOBJECT **WHICH HAVE SAME OBJECTS INSIDE (TEXT, BUTTON)!**



4. 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 WHICH IS SET IN INPUT BUTTON

7. THEN YOU MUST SET CONTROL HELPER LOCATION TO INPUT CONTROLLER SCRIPT

8. **YOU NEED TO REPEAT ALL STEPS IN GAMEMANAGER**

9. DONE

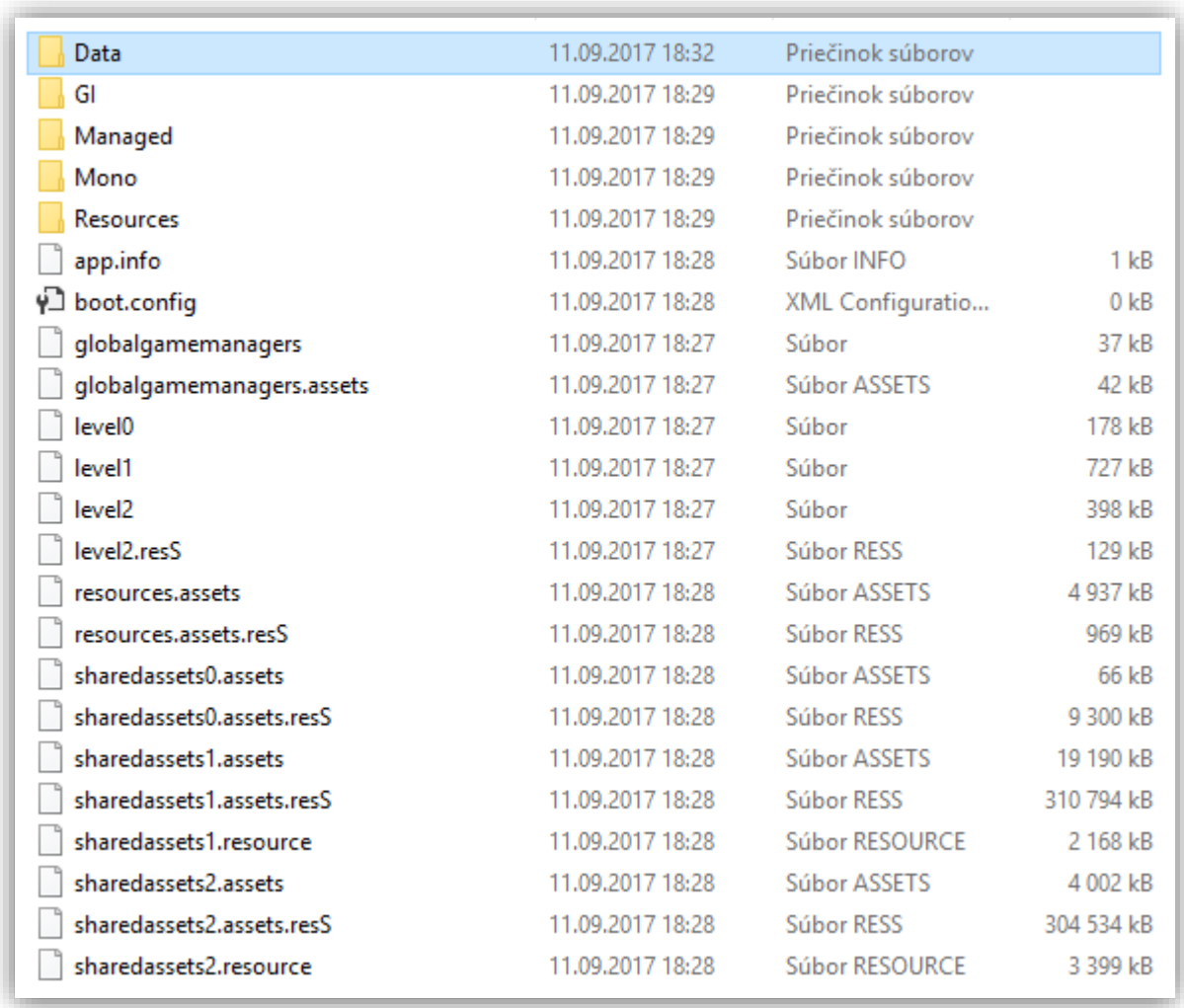
USING CONFIG IN BUILDED GAME !

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
globalgamemanagers	11.09.2017 18:27	Súbor	37 kB
globalgamemanagers.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**
- IF YOU HAVE NEW SCRIPT YOU MUST CONNECT IT WITH **ConfigHandler.cs**

1. WRITE USING PARSER NAMESPACE

```
using UnityEngine;
using ThunderWire.Helper.Parser;
```

2. DEFINE **ConfigHandler**

```
public ConfigHandler configHandler;
```

3. DEFINE NEW KEY

```
private KeyCode useKey;
```

4. WRITE PARSING SENTENCE TO UPDATE

```
void Update()
{
    if (configHandler.GetKeysCount() > 0 && !isSet)
    {
        useKey = Parser.Convert<KeyCode>(configHandler.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 UnityEngine;
using ThunderWire.Helper.Parser;
```

TO PARSE FROM STRING TO CORRECT TYPE USE THIS COMMAND

```
Parser.Convert<TYPE>("STRING");
```

VECTOR 2:

```
Parser.Convert(string x, string y);
```

VECTOR 3:

```
Parser.Convert(string x, string y, string z);
```

VECTOR 4 or QUATERNION:

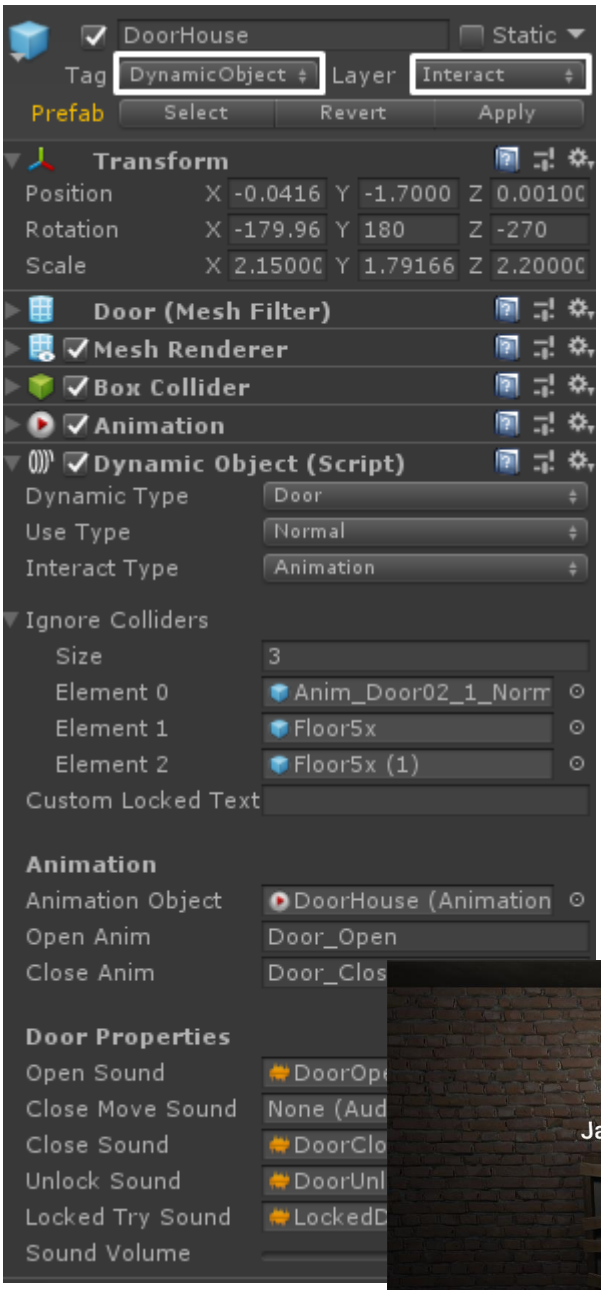
```
Parser.Convert<TYPE>(string x, string y, string z, string w);
```

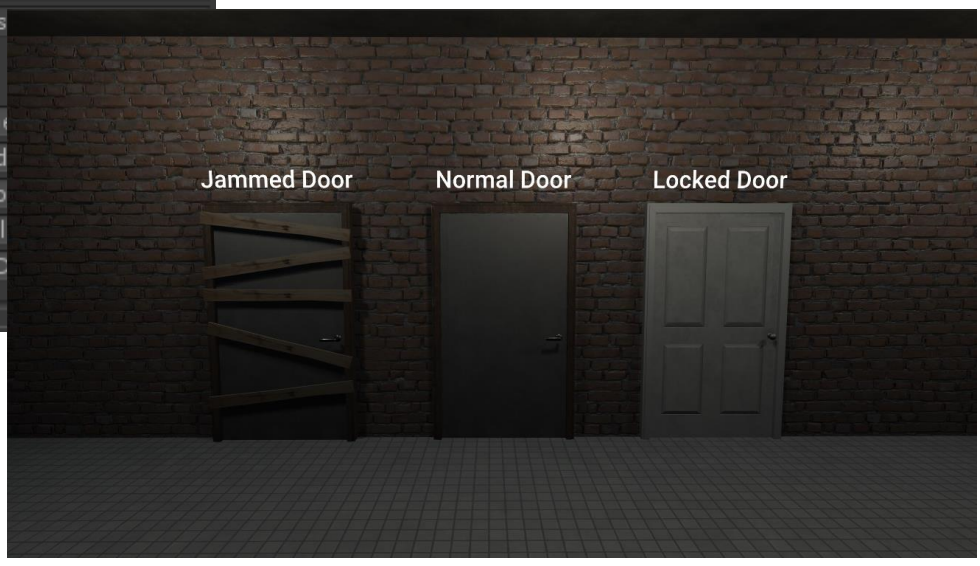
TYPE MUST BE VECTOR4 or QUATERNION! OTHERWISE YOU WILL GET A ERROR.

DYNAMIC OBJECTS (DYNAMIC MAMAGER)

SETTING UP DYNAMIC DOOR

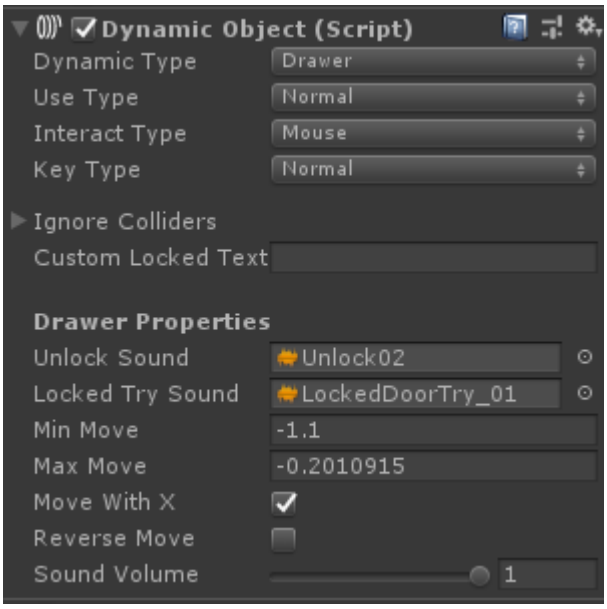
- DYNAMIC OBJECTS MUST HAVE **DynamicObject** TAG AND **Interact** LAYER
- YOU CAN SWITCH BETWEEN USE TYPES (Normal, Locked, Jammed)
- IF LIKE OPENING DOOR WITH **MOUSE** OR **ANIMATION** YOU CAN ALWAYS SWITCH BETWEEN THESE MODES
- YOU CAN ALSO CHANGE KEY UNLOCK TYPE BETWEEN **NORMAL** BY KEY SCRIPT OR BY **INVENTORY** ID.



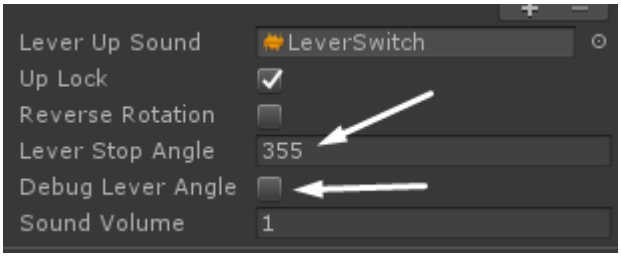


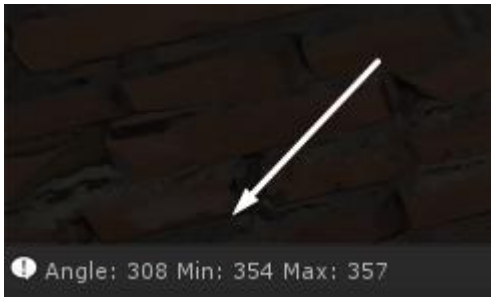
DYNAMIC DRAWER

- 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**



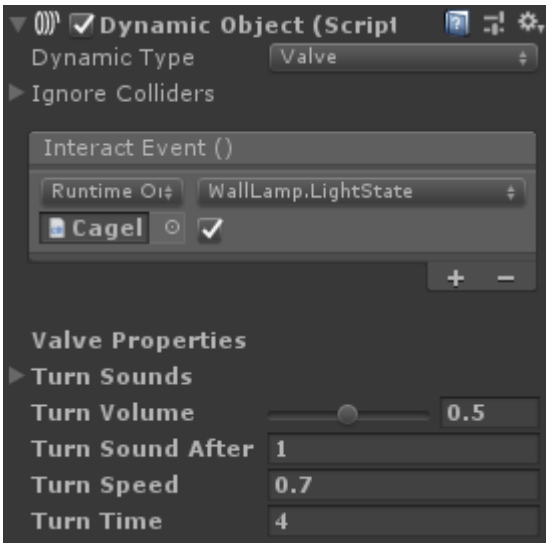
DYNAMIC LEVER





- 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

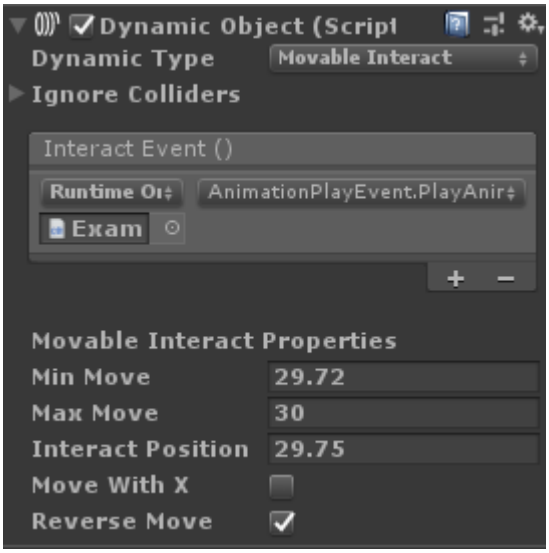
VALVE



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

MOVABLE INTERACT

- THIS IS GOOD FOR MAKING SECRET ROOMS



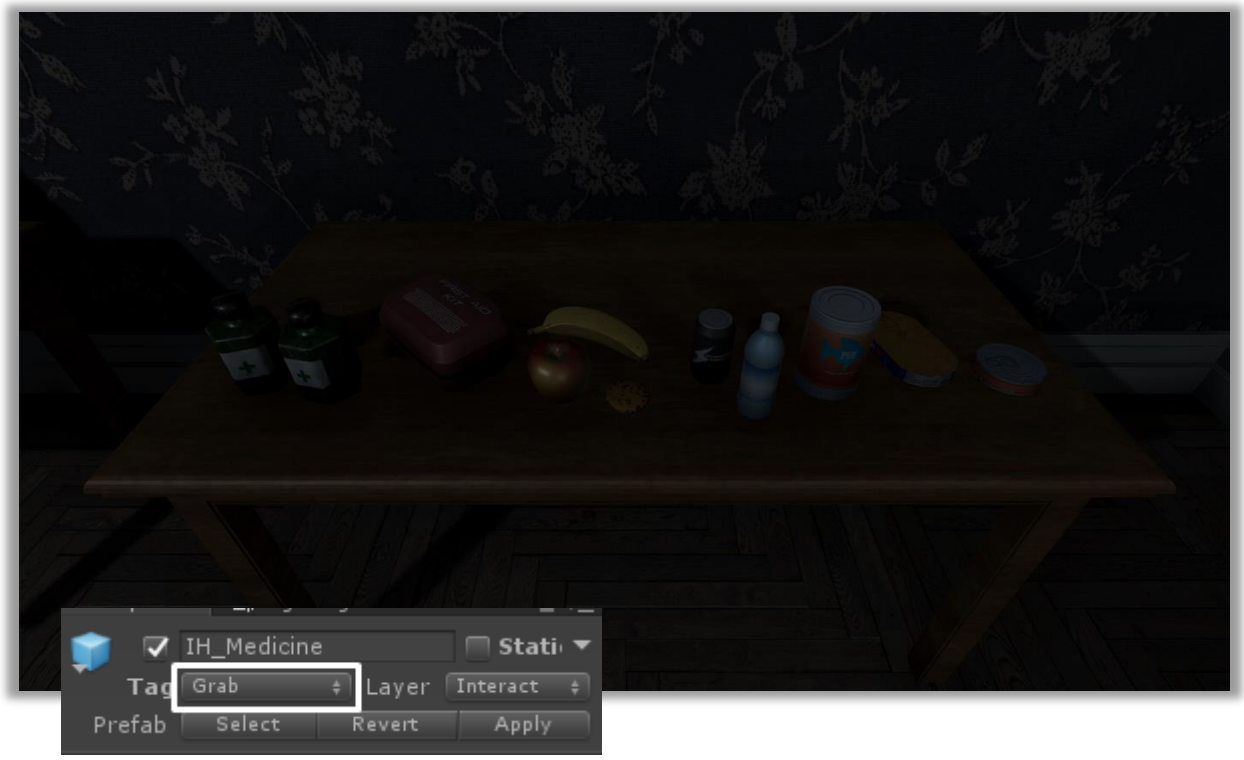
- THIS IS NORMALLY A DYNAMIC DRAWER BUT WITH INTERACT FUNCTION
- WHEN YOU TICK **MOVE WITH X** BOOL THE SCRIPT WILL MOVE WITH TRANSFORM **X** POSITION
- WHEN THE POSITION **Z** OR **X** OF OBJECT IS IN INTERACT POSITION, SCRIPT WILL INVOKE INTERACT EVENT

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

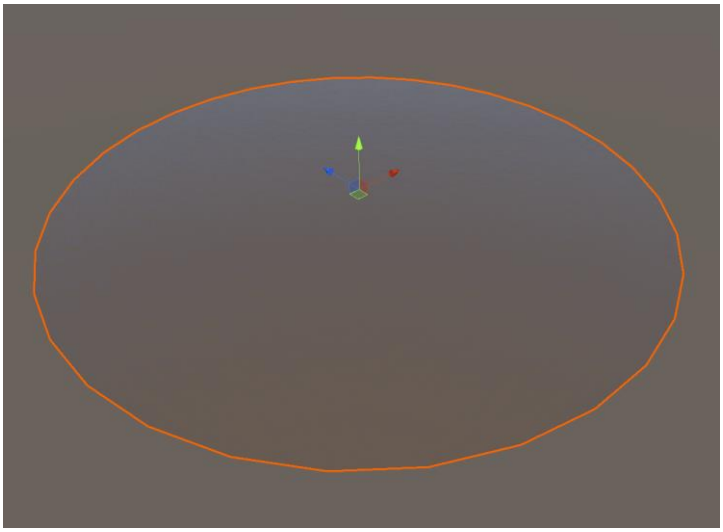
THE **GRAB** TAG IS FOR DRAGGABLE AND PICKUPABLE ITEMS



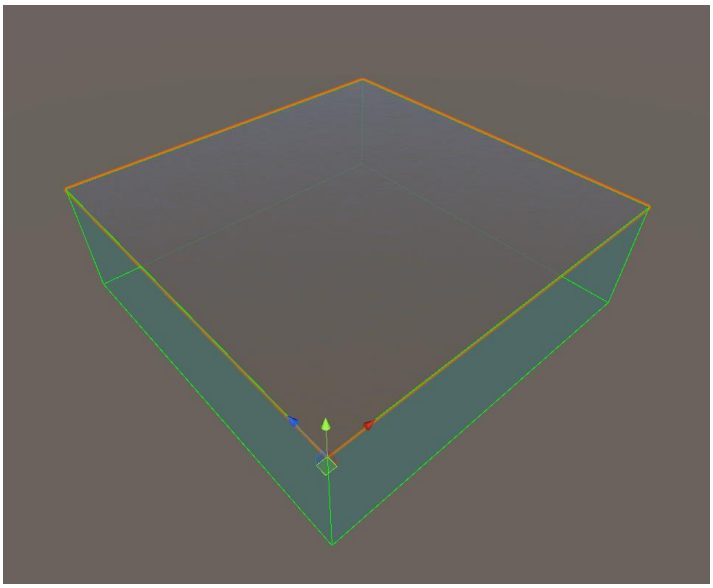
AND THE **ONLYGRAB** TAG IS FOR CRATES OR FOR ITEMS THAT CAN BE ONLY DRAGGABLE

WATER BUOYANCY

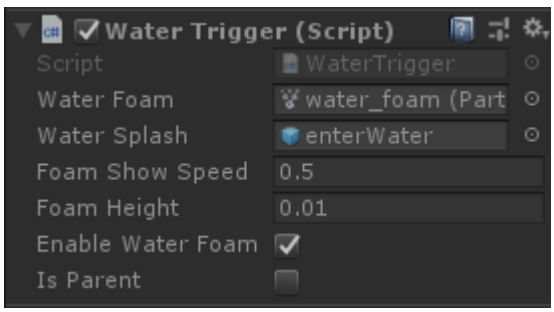
1. DRAG & DROP WATER OBJECT



2. ADD **WaterVolume.cs** SCRIPT AND CHANGE WATER OBJECT TAG TO A **Water Volume**
3. SCRIPT WILL AUTOMATICALLY CREATE PLANE INSTANCE OF A OBJECT



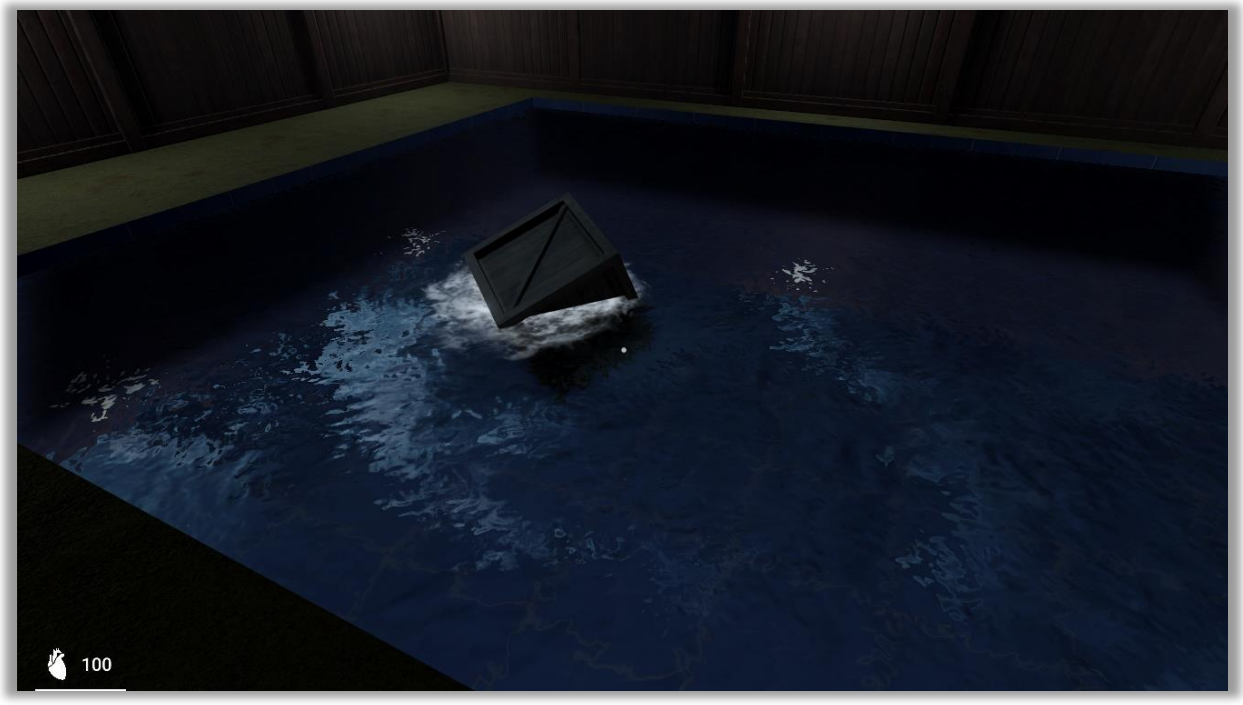
4. TO CHANGE PLANE DIMENSIONS USE SCRIPT **ROWS** AND **COLUMNS**
5. ADD **WaterTrigger.cs** SCRIPT WHICH CONTROLS OBJECT FOAMS AND PLAYER IN WATER STATE



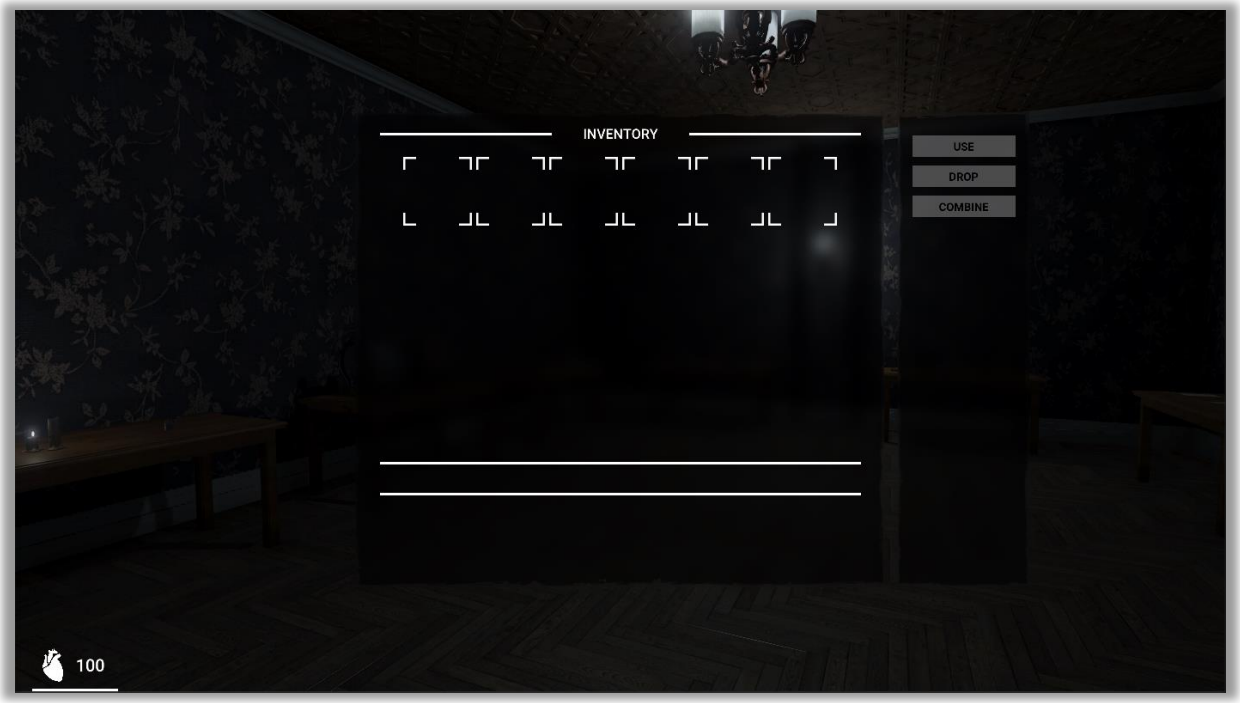
6. ADD **FloatingObject.cs** SCRIPT TO OBJECTS WHICH YOU WANT TO FLOAT ON WATER

7. YOU CAN CHANGE OBJECT DENSITY

OBJECT DENSITY > WATER DENSITY = LOWER OBJECT BUOYANCY

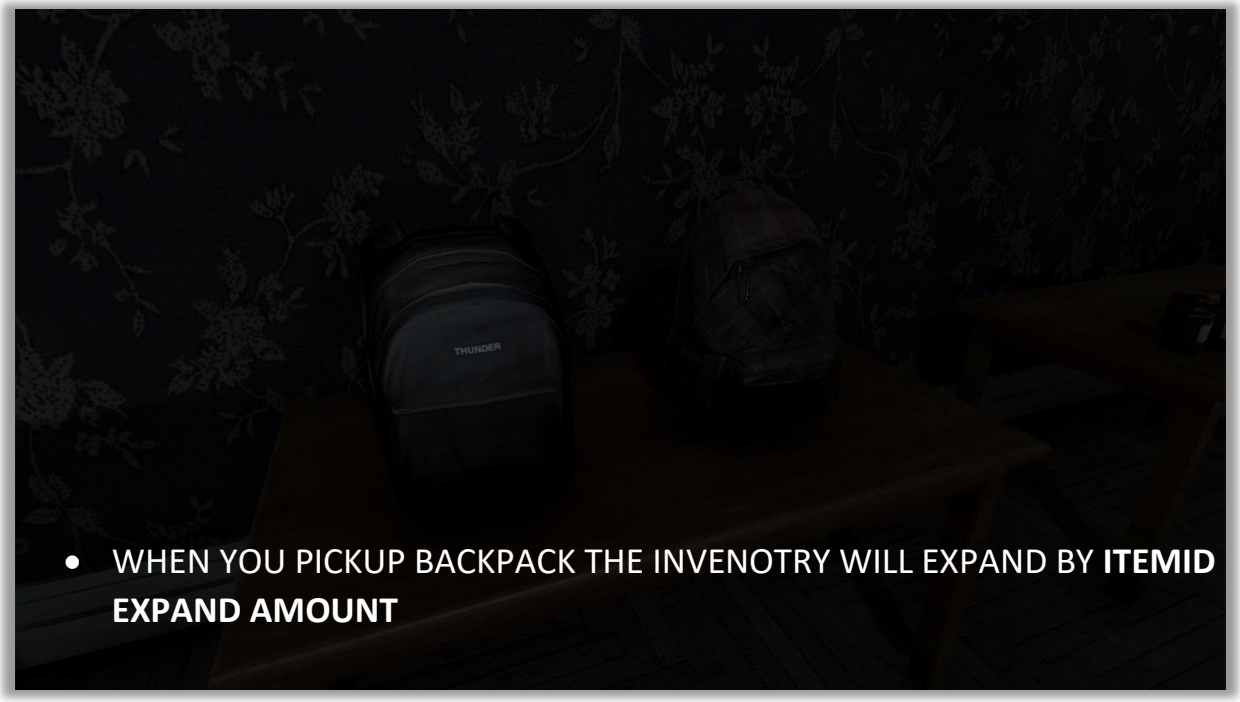


INVENTORY

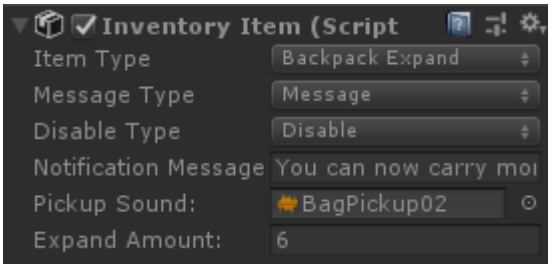


- YOU CAN SHOW INVENTORY MENU BY PRESSING **TAB** BUTTON
- IF YOU WANT CHANGE INVENTORY SHOW BUTTON YOU CAN EASILY CHANGE IT IN **MAIN** OR **PAUSE** MENU.
- DEFAULT SLOT AMOUNT IS SET TO 6 BUT YOU CAN CHANGE IT IN INVENTORY SCRIPT

BACKPACK PICKUP (INVENTORY EXPAND)



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

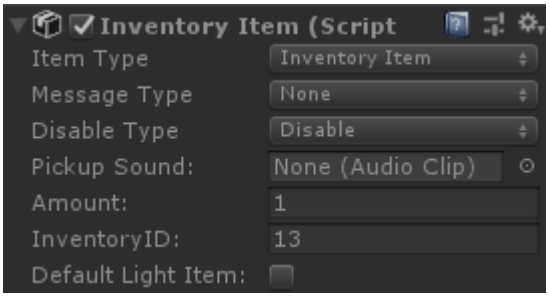


INVENTORY TWEAKS

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

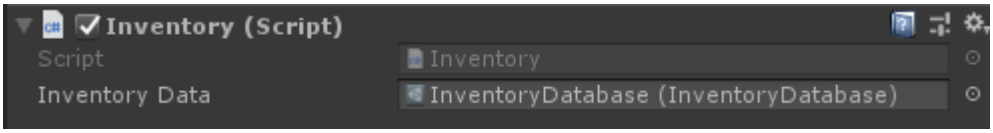
INVENTORY ITEM PICKUP

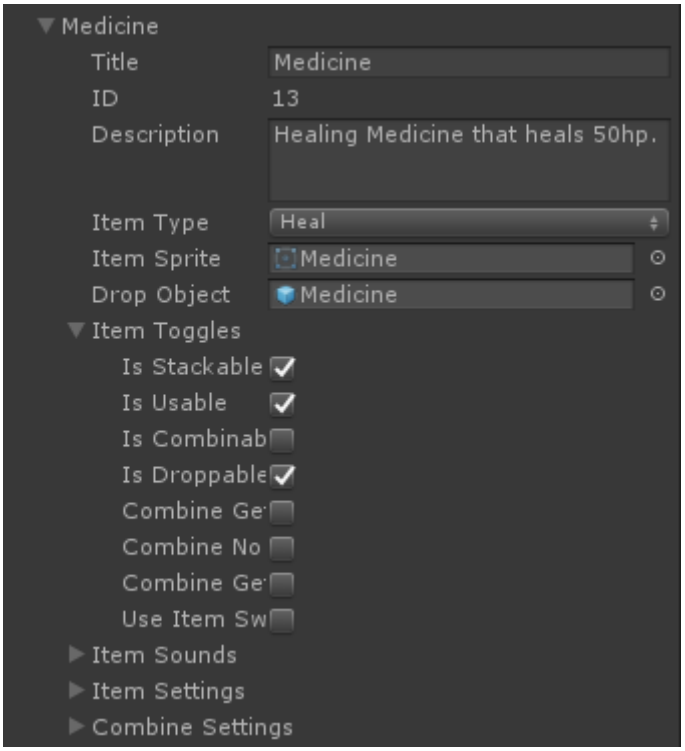
1. ADD **InventoryItem.cs** SCRIPT TO YOUR OBJECT AND CHANGE **ITEM TYPE** TO **INVENTORY ITEM**



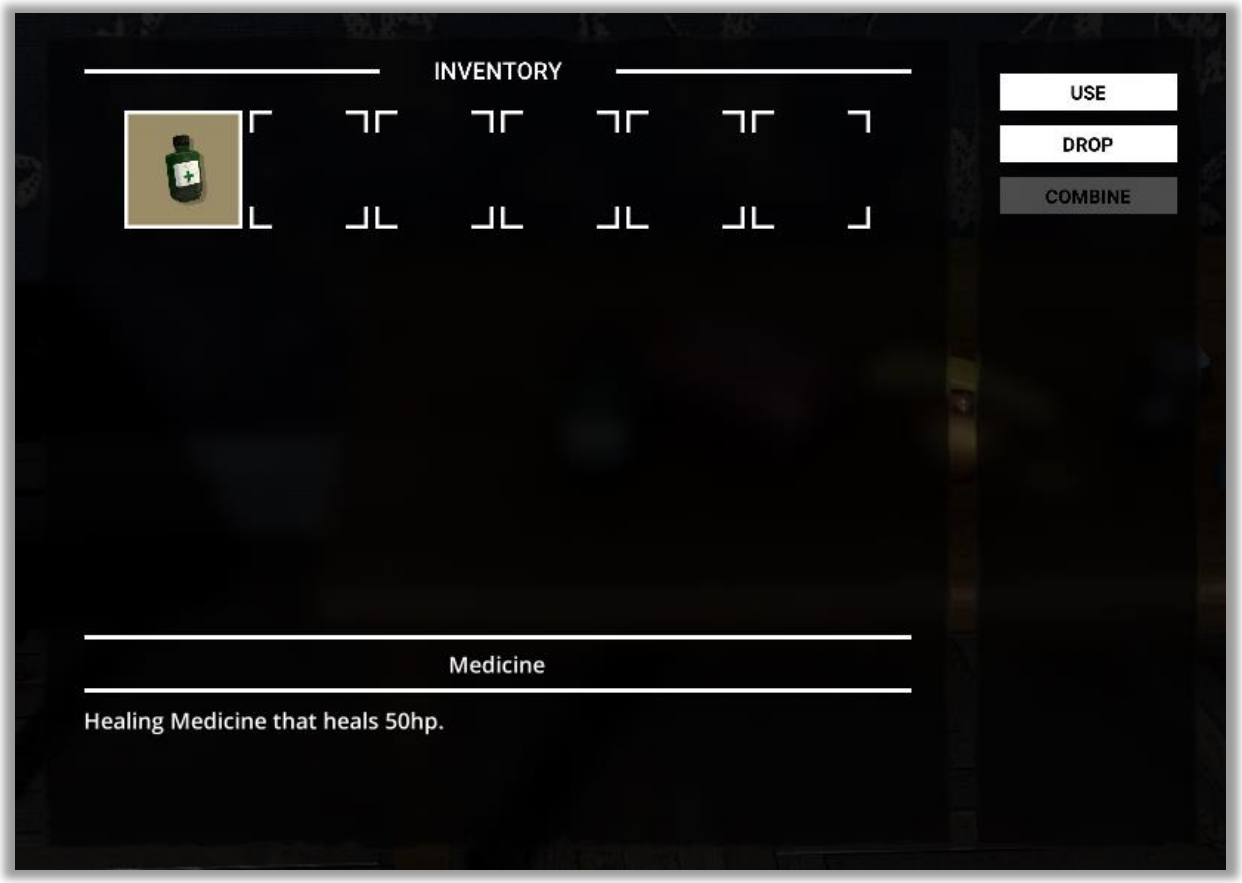
2. IF YOU DOESN'T HAVE INVENTORY DATABASE ASSET YOU CAN CREATE IT IN **TOOLS -> HFPS KIT -> SCRIPTABLES -> CREATE INVENTORY DATABASE**

3. CLICK TO **GAMEMANAGER** OBJECT AND IN INVENTORY SCRIPT SET **INVENTORY DATA** WITH **INVENTORY DATABASE ASSET**
4. NEXT YOU MUST WRITE TITE, LITTLE DESCRIPTION OF YOUR ITEM AND SET ITEM ICON
5. AFTER THAT YOU CAN SET SOME PROPERTIES OF YOUR ITEM IN MY CASE I SET **ITEM TYPE TO HEAL,** CHANGED SOME **ITEM TOGGLES** AND CHANGED **HEAL AMOUNT** IN **ITEM SETTINGS**



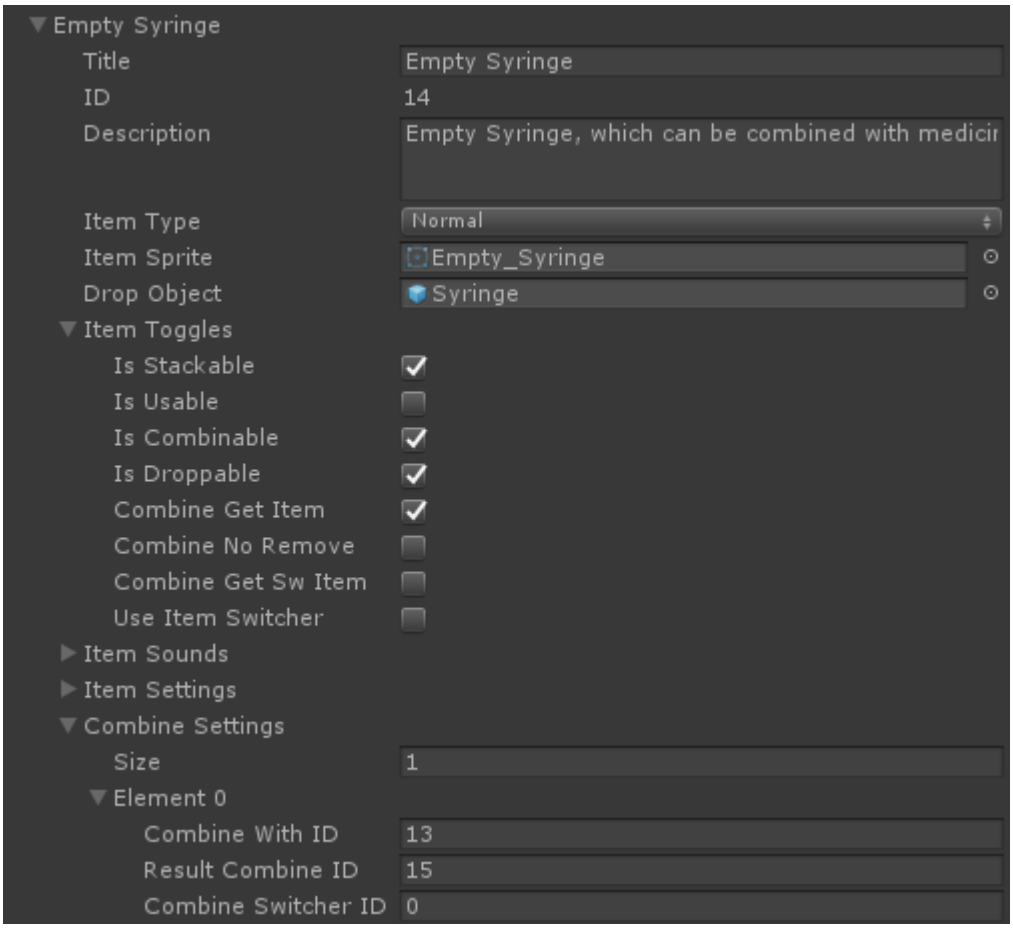


- ITEM ID IS DISPLAYED IN **INVENTORY DATABASE ASSET** SO YOU CAN EASY DETERMINE WHICH ID HAVE YOUR ITEM



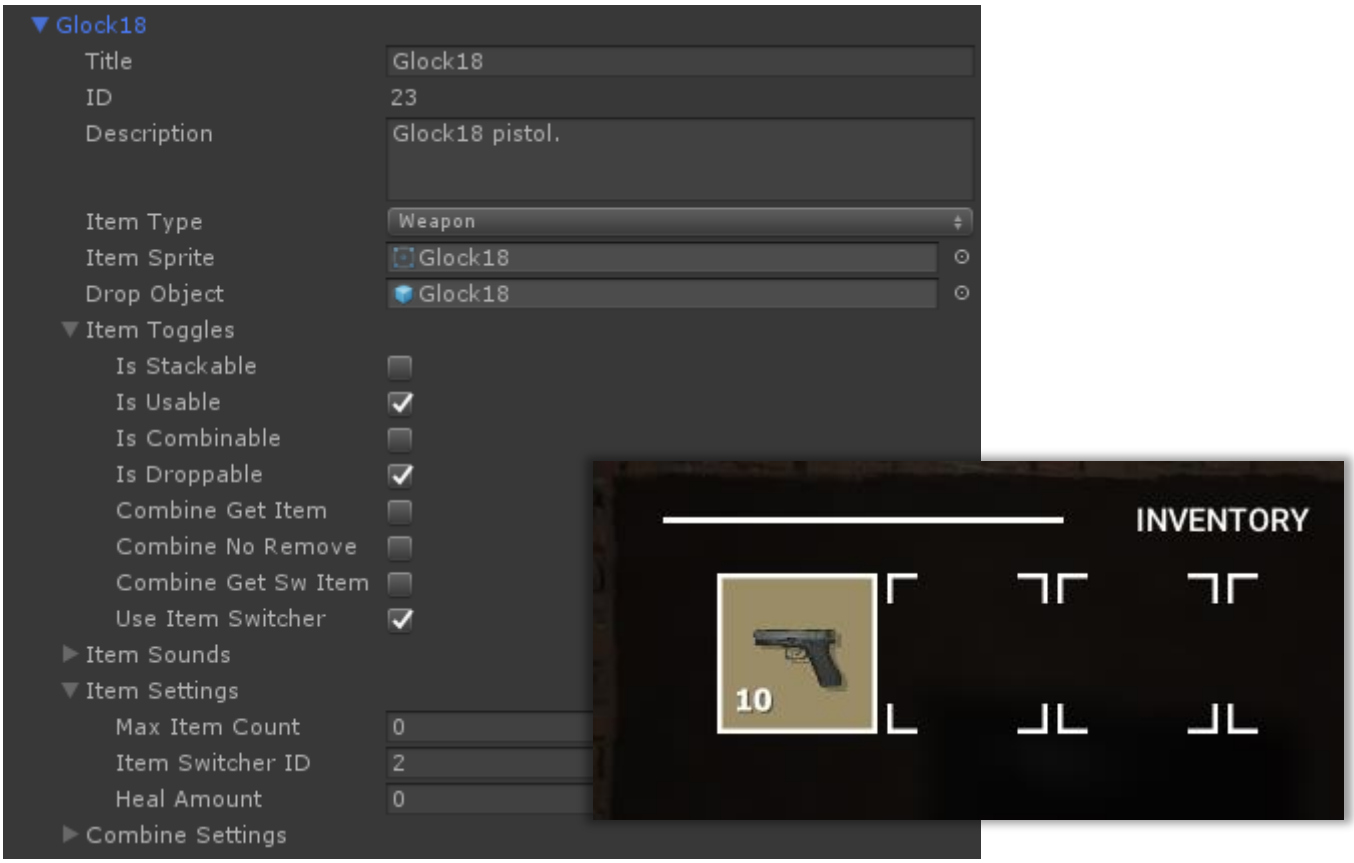
COMBINABLE ITEM

- THE ONLY THING WHAT YOU NEED TO DO IS SET SOME PROPERTIES IN INVENTORY DATABASE ASSET TO MAKE ITEM COMBINABLE
- ONE ITEM CAN BE COMBINABLE WITH MULTIPLE ITEMS TO GET DIFFERENT ITEMS

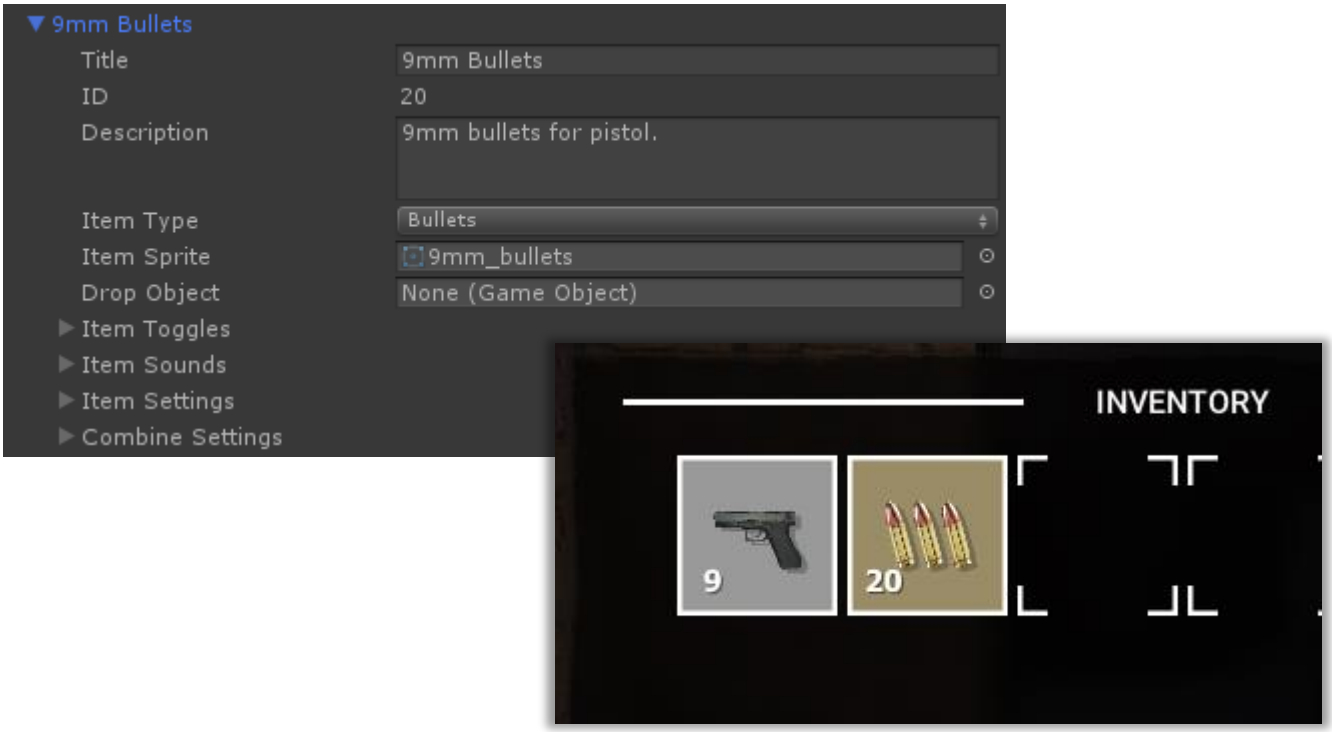


INVENTORY WEAPONS AND BULLETS

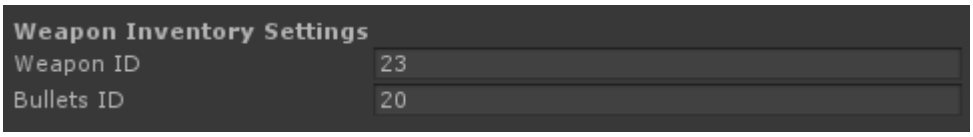
- IF YOU WANT TO ADD NEW WEAPON YOU MUST ADD WEAPON ITEM TO INVENTORY DATABASE, **SET ITEM TYPE** TO **WEAPON** AND SET **ITEM SWITCHER ID**



- AFTER THAT YOU NEED TO ADD BULLETS ITEM AND SET ITEM TYPE TO BULLETS

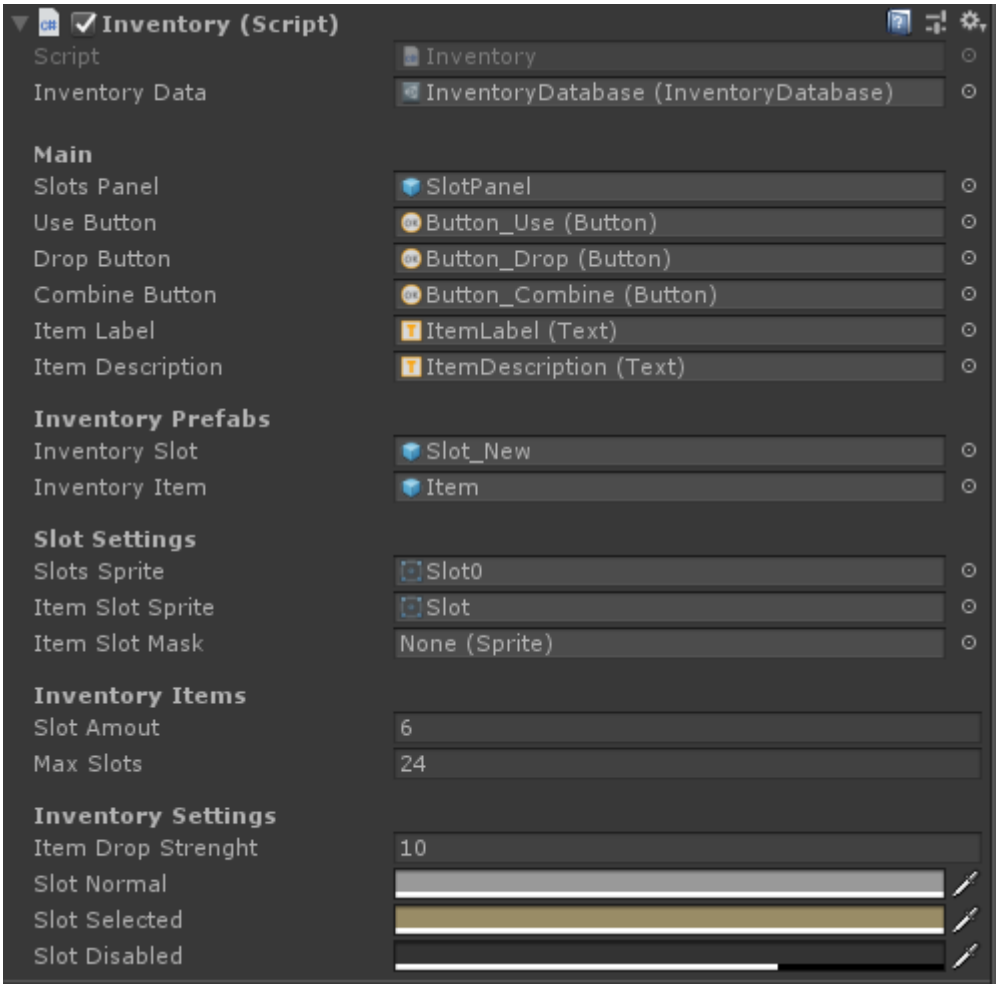


- LAST STEP IS GO TO **WEAPON CONTROLLER** AND SET **INVENTORY SETTINGS**



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

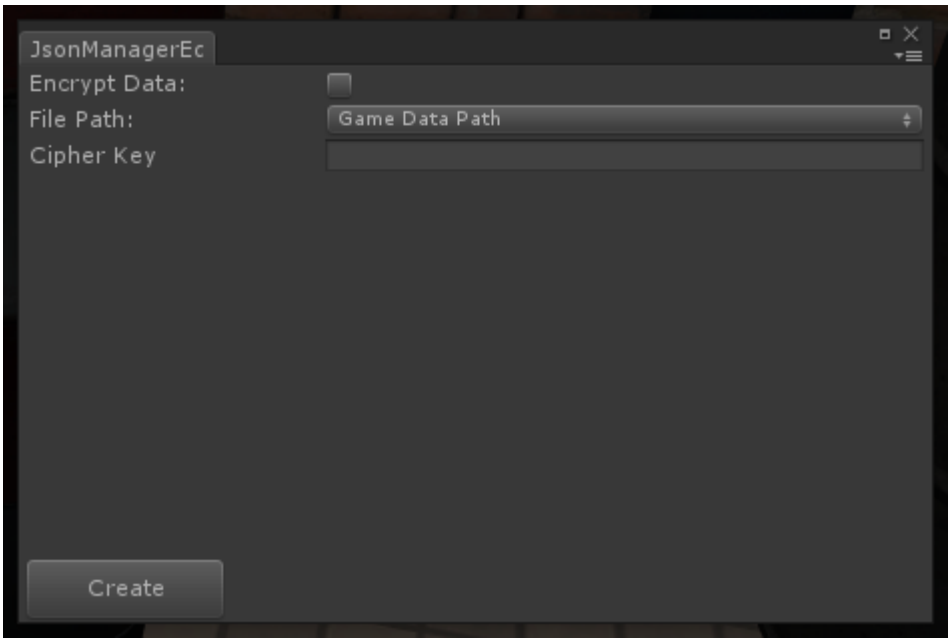


- YOU CAN STYLIZE INVENTORY HOW YOU WANT

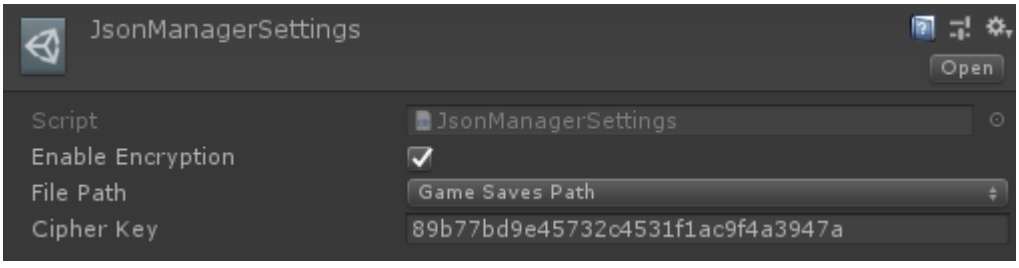
SAVE/LOAD MANAGER

SETTING UP SAVE/LOAD MANAGER

- FIRST OF ALL YOU NEED TO CREATE JSON MANAGER ASSET IN **TOOLS -> HFPS KIT -> SCRIPTABLES -> CREATE JSONMANAGER SCRIPTABLE**



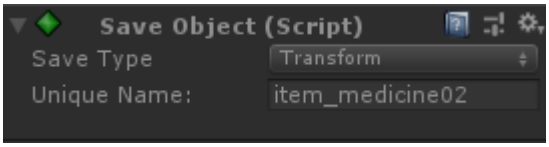
- THERE YOU CAN SET IF YOU WANT ENCRYPT DATA AND SET DEFAULT FILEPATH OF SAVED GAME
- AFTER CLICKING CREATE BUTTON THE JSON MANAGER SETTINGS ASSET WILL BE CREATED, CIPHER KEY FOR SECURITY REASON WILL BE ENCRYPTED BY MD5



- YOU NEED TO SET JSON MANAGER SETTINGS FILED IN **SAVE GAME HANDLER SCRIPT** WITH CREATED **JSON MANAGER SETTINGS ASSET**

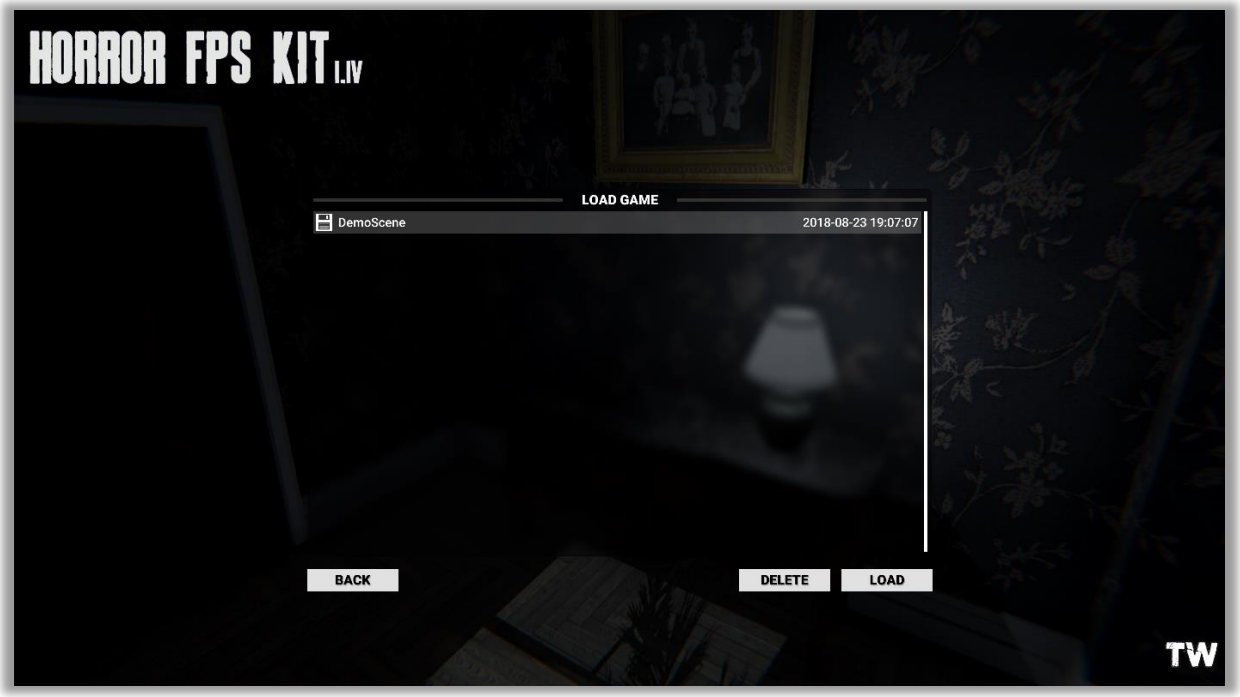
ADDING SAVEABLE OBJECTS

- 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 (PLAYER DATA, INVENTORY, POSITIONS, ROTATIONS, SCRIPTS...)

DISKETTE ICON INDICATES WHEN THE GAME IS SAVED



- DEFAULT SAVE GAME LOCATION IS **\YOURGAME_DATA\DATA\SAVEDGAME**
- SAVED GAME DATA IS ENCRYPTED, SO HACKERS HAVE NO CHANCE TO EDIT SAVED GAME DATA

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:

```
JsonManager.UpdateJsonArray("key", object value));
```

- WITH THIS SENTENCE YOU CAN SAVE EVERYTHING
- JSON IS SCALABE SO VALUE CAN BE DICTIONARY WITH STRING - OBJECT

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

```
int number = JsonManager.Json()["your key"].ToObject<int>();
```

- YOU CAN CONTINUE LIKE THIS

```
JsonManager.Json()["your key"]["your key 2"]...[]
```

SAVING CUSTOM SCRIPT DATA

1. ADD **SaveHelper.cs** UNDER YOUR CUSTOM SCRIPT
2. OPEN YOUR CUSTOM SCRIPT IN SCRIPT EDITOR
3. ADD THIS SENTENCES WITH YOUR OWN VALUES TO YOUR SCRIPT

```
public void OnSave()
{
    if (GetComponent<SaveHelper>())
    {
        GetComponent<SaveHelper>().SetArray(new Dictionary<string, object>(){
            { "isSaved", isPlayed }
        });
    }
}

public void OnLoad(Newtonsoft.Json.Linq.JObject token)
{
    isPlayed = (bool)token["isSaved"];
}
```

4. ADD **SaveObject.cs** SCRIPT UNDER **SaveHelper.cs** SCRIPT AND CHANGE SAVE TYPE TO **SAVE HELPER**

- SAVE HELPER SENDS CALL FUNCTION **OnSave**() TO YOUR SCRIPT TO INVOKE IT AND GET PRE SET VALUES
- FOR LOAD **SAVE HELPER** SENDS CALL FUNCTION **OnLoad(JToken)** WITH JTOKEN SCRIPT TO SET SAVED VALUES
- YOU CAN EASY SAVE AND LOAD VALUES LIKE THIS:

SAVE:

```
public void OnSave()
{
      GetComponent<SaveHelper>().SetArray(
            new Dictionary<string, object> {
            {"position", transform.localPosition}
      });
}
```

LOAD:

```
public void OnLoad(Newtonsoft.Json.Linq.JToken token)
{
      transform.localPosition = token["position"].ToObject<Vector3>();
}
```

SAVING USING ATTRIBUTE

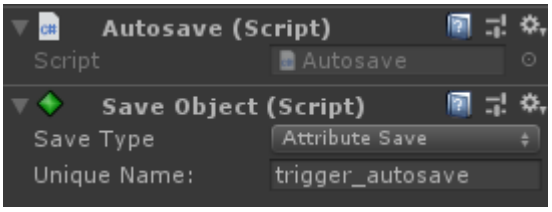
FOR EXAMPLE YOU CAN OPEN **Autosave.cs** SCRIPT

ONLY THING WHAT YOU NEED TO DO IS MARK SCRIPT FIELD AS A **[SaveableField]**

```
[SaveableField, HideInInspector]
public bool isPlayed;
```

NEXT YOU WILL NEED TO ADD **SaveObject.cs** SCRIPT UNDER SCRIPT WHICH WILL BE SAVED, AND YOU WILL NEED TO CHANGE **SAVE TYPE** TO A **ATTRIBUTE SAVE**

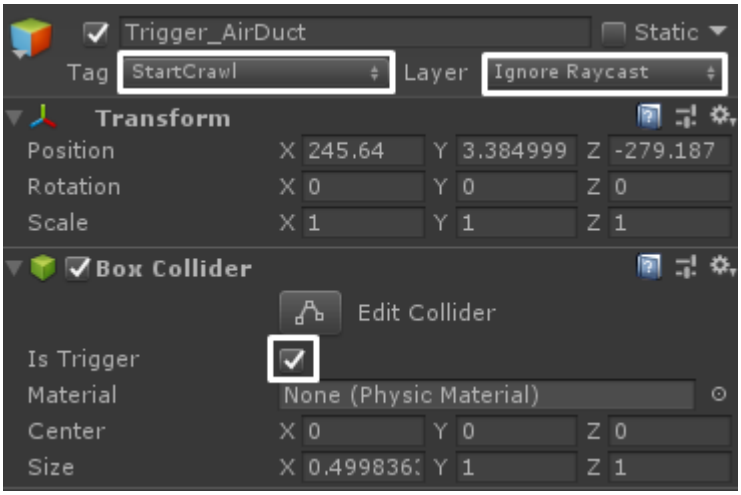
FIELD WHICH WILL BE SAVED MUST BE PUBLIC!



SaveObject.cs SCRIPT WILL AUTOMATICALLY FIND ALL FILEDS MARKED AS **[SaveableField]** AND SAVES IT

PLAYER CRAWL

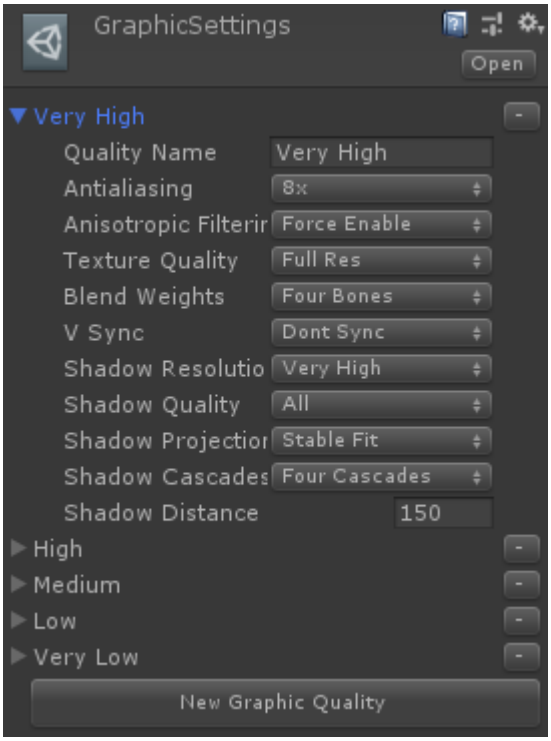
PLAYER CRAWL STATE CAN BE ONLY TRIGERRED BY A TRIGGER WITH **StartCrawl** TAG



ADDING NEW GRAPHIC SETTINGS

TO ADD NEW GRAPHIC SETTINGS YOU NEED TO EDIT **GraphicSettings** SCRIPTABLE WHICH IS LOCATED IN **SCRIPTABLES** FOLDER

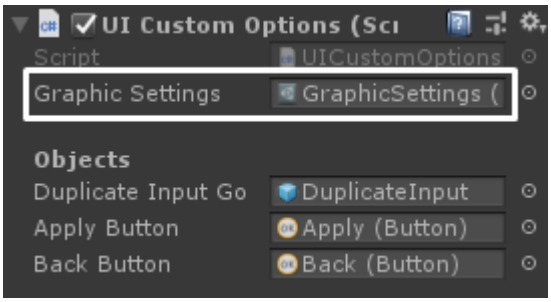
IF YOU DOES NOT HAVE THIS SCRIPTABLE YOU CAN CREATE IT BY SELECTING **TOOLS - > HFPS KIT -> SCRIPTABLES -> CREATE GRAPHIC SCRIPTABLE**



YOU CAN CHANGE MAIN GRAPHIC SETTINGS

CURRENT UNITY GRAPHIC QUALITY WILL BE OVERWRITTEN BY THIS GRAPHIC SCRIPTABLE SETTINGS

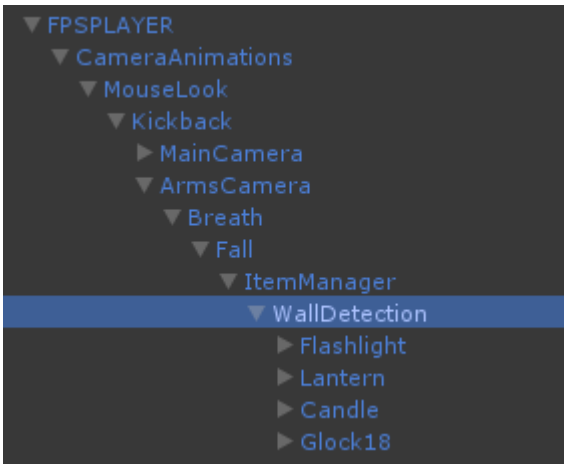
AFTER CREATING GRAPHIC SCRIPTABLE YOU MUST SELECT IT IN **UICustomOptions.cs** (REPEAT THIS STEP IN GAME SCENE)



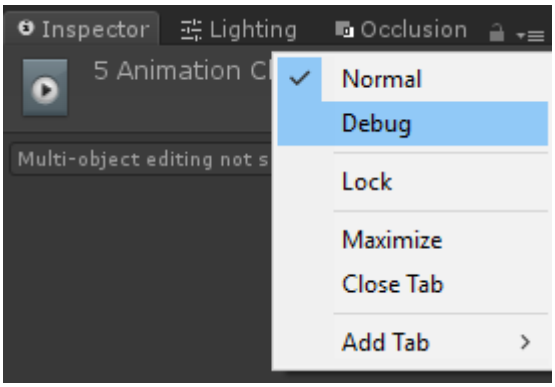
CHANGED GRAPHIC SETTINGS IN RUNTIME WILL BE SAVED TO A CONFIG FILE

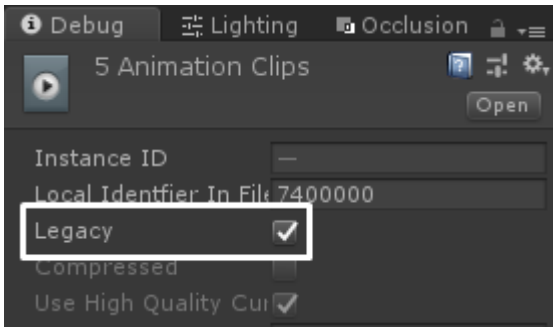
ADDING NEW WEAPONS/ITEMS

1. LOCATE **WallDetection** OBJECT INSIDE **FPSPLAYER** WHICH CONTAINS ALL WEAPON/ITEM OBJECTS

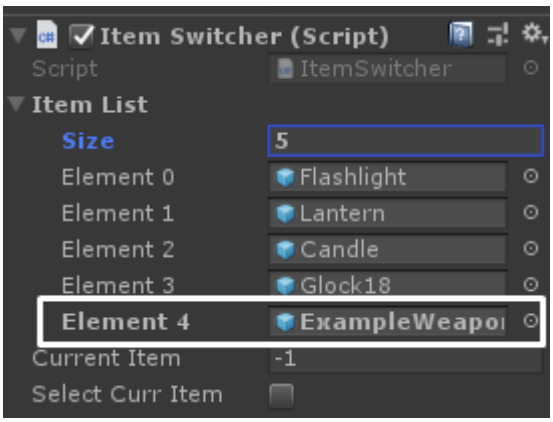


2. DUPLICATE ONE OF THESE ITEMS AND REPLACE DISABLED OBJECT INSIDE DUPLICATED OBJECT WITH YOUR OWN WEAPON
3. MARK YOUR WEAPON ANIMATIONS AS A LEGACY ANIMATIONS AND ADD IT TO YOUR WEAPON ROOT OBJECT





4. ADD YOUR NEW WEAPON TO A **ITEMSWITCHER** SCRIPT



IF YOU WANT TO CREATE YOUR OWN WEAPON/ITEM SCRIPT YOU MUST CREATE MAIN FUNCTIONS TO **SELECT, DESELECT, LOADERSELECT** YOUR ITEM

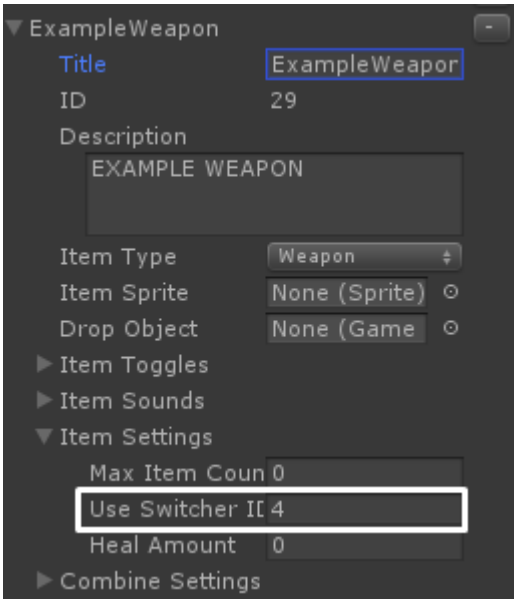
YOU CAN OPEN FOR EXAMPLE FLASHLIGHT SCRIPT AND FOLLOW HOW FLASHLIGHT WILL BE **SELECTED** OR **DESELECTED**..

```
public void Select()
{
    StartCoroutine(UIFader.FadeIn(2));
    FlashlightGO.SetActive(true);
    AnimationComp.Play(DrawAnim);
    isSelected = true;
}

public void Deselect()
{
    if (FlashlightGO.activeSelf && !isReloading)
    {
        StartCoroutine(UIFader.FadeOut(2));
        StartCoroutine(DeselectCorountine());
    }
}
```

```
public void LoaderSetItemEnabled()
{
    FlashlightGO.SetActive(true);
    AnimationComp.Play(IdleAnim);
    isSelected = true;
    isOn = true;
}
```

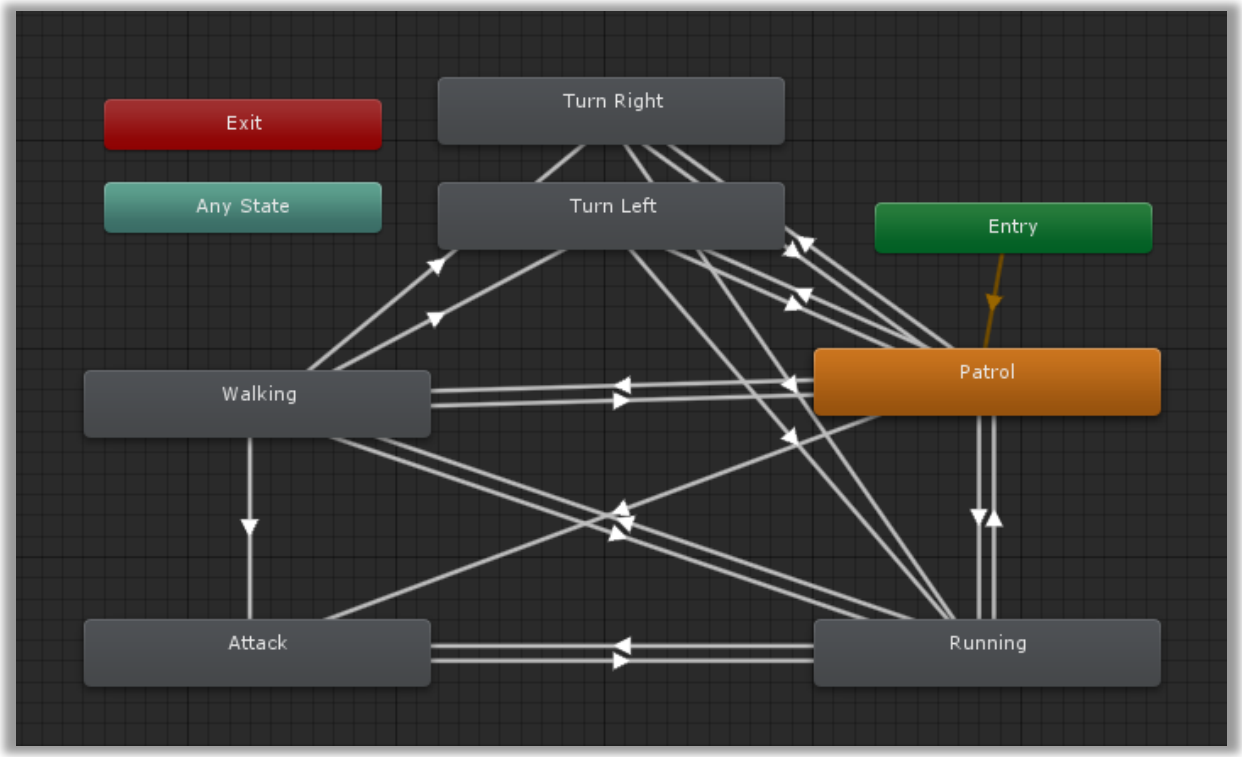
5. AFTER CREATING YOUR OWN WEAPON/ITEM DOES NOT FORGET TO ADD YOUR ITEM TO A INVENTORY DATABASE AND SET **USE SWITCHER ID**

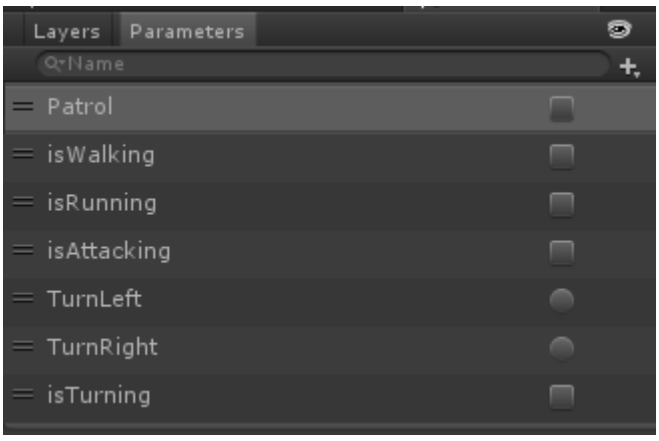


AI SYSTEM

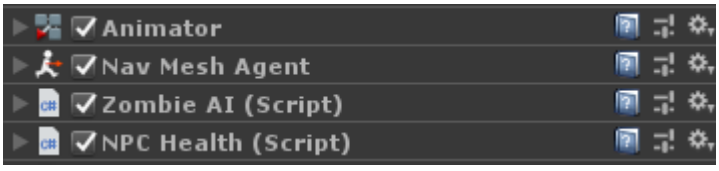
ADDING NEW ZOMBIE

1. DUPLICATE OR CREATE SAME ZOMBIE ANIMATOR AS HAVE MY ZOMBIE

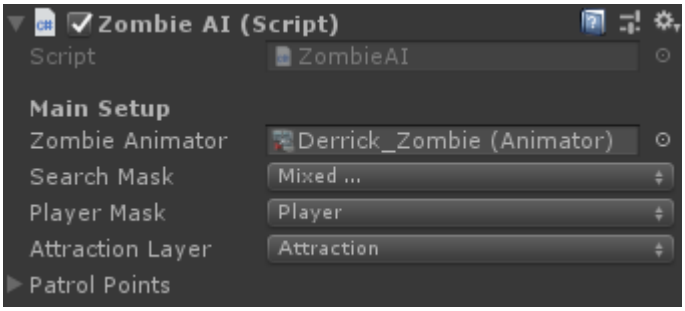




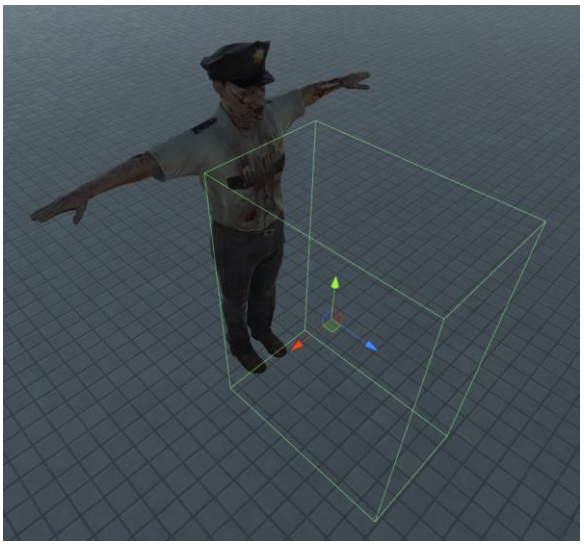
2. FIRST OF ALL CONVERT YOUR ZOMBIE TO A RAGDOLL **GAMEOBJECT -> 3D OBJECT -> RAGDOLL**
3. ADD NEDDED SCRIPTS TO YOUR ZOMBIE

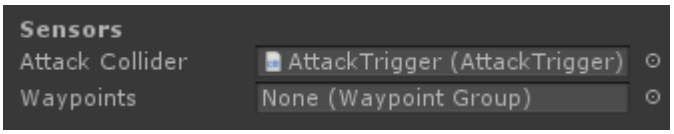


4. YOUR ZOMBIE MUST HAVE **ZOMBIE** LAYER AND ZOMBIE HIPS MUST HAVE **BODYPART** LAYER
5. SET **HIPS TRANSFORM** IN **ZOMBIE HEALTH** SCRIPT AND SET ZOMBIE **HEALTH** POINTS
6. IN **ZOMBIE AI** SCRIPT SET ZOMBIE **ANIMATOR** AND SET **MASKS** (YOU CAN LOOK AT MY ZOMBIE PREFAB HOW I SETTED UP ZOMBIE AI)

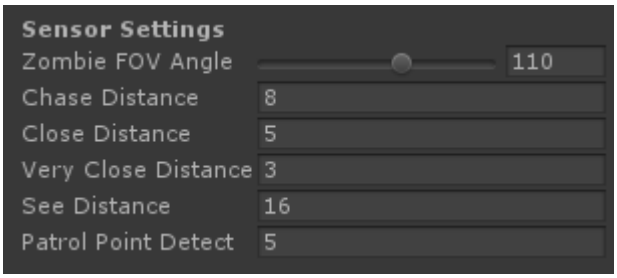


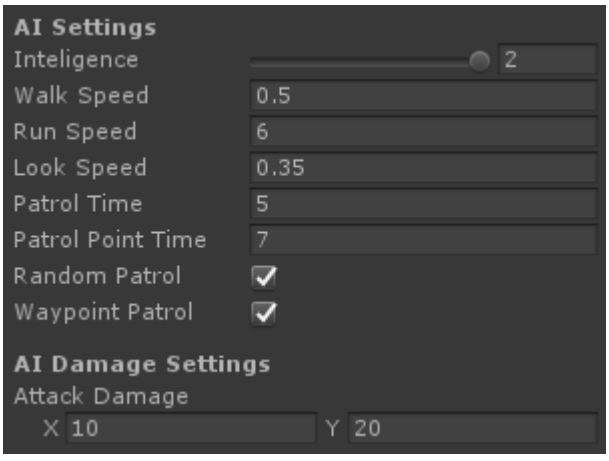
7. CREATE ZOMBIE **ATTACK TRIGGER,** ADD **ATTACK TRIGGER SCRIPT** AND SELECT IT UNDER SENSORS SECTION



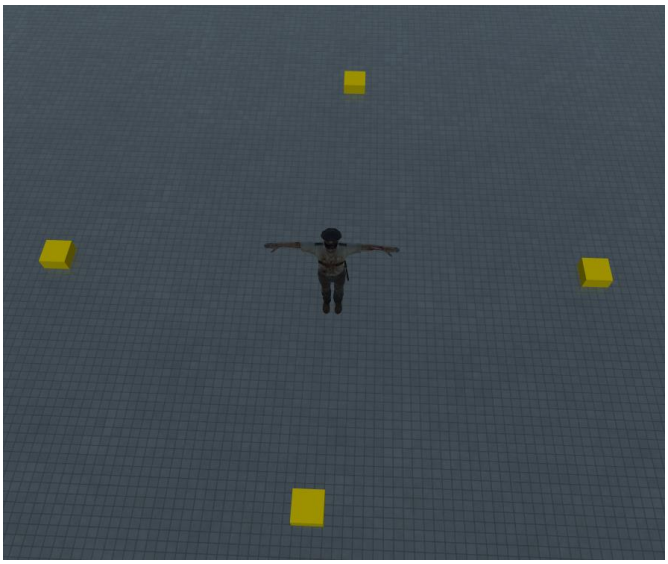


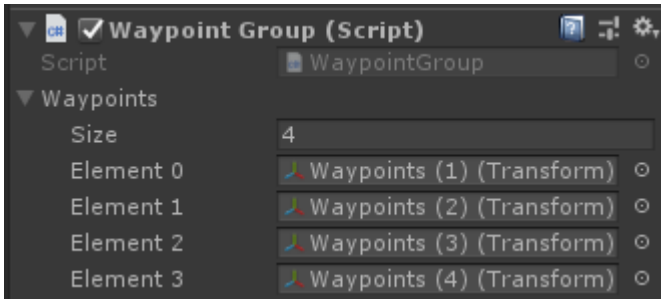
8. SET OTHER ZOMBIE AI SETTINGS



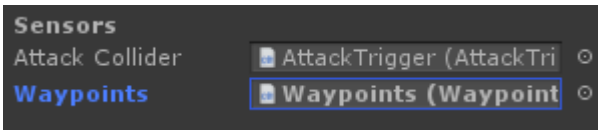


9. CREATE EMPTY GAMEOBJECT AND ADD **WAYPOINT GROUP SCRIPT**
10. BY ADDING EMPTY GAMEOBJECTS TO A OBJECT WHERE IS WAYPOINT GROUP SCRIPT YOU SET NEW WAYPOINT. (WAYPOINTS WILL BE DEFINED AUTOMATICALLY)





11. DON'T FORGOT TO SELECT **WAYPOINTS FILED** IN **ZOMBIE AI SCRIPT**



- **INTELLIGENCE** SLIDER SETS MAIN ZOMBIE INTELLIGENCE SETTINGS AS IS (ATTRACTED STATE, GO TO PATROL POINT STATE, LOOK STATE)

❖ **INTELLIGENCE 1** = ZOMBIE CAN BE ATTRACTED AND CAN GO TO A PATROL POINT
❖ **INTELLIGENCE 2** = ZOMBIE WILL TURN TO A DIRECTION WHERE YOU FIRED

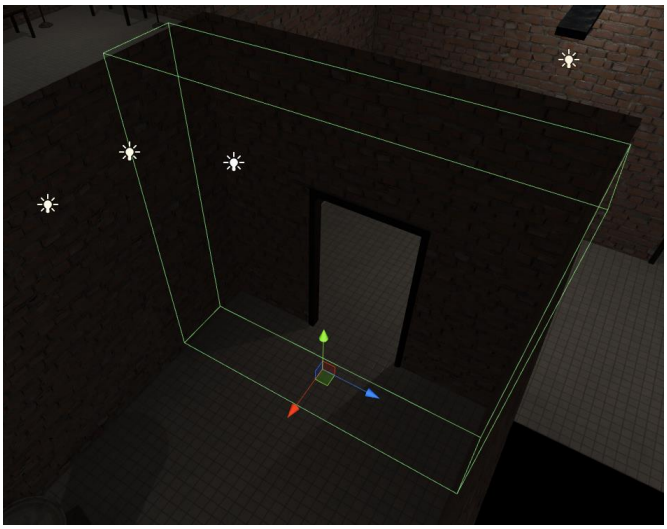
- PATROL POINT IS POINT WHERE ZOMBIE GO IF DISTANCE BETWEEN **LAST SEEN POSITION** AND **POTROL POINT POSITION** IS IN RANGE OF **PATROL POINT DETECT**

JUMPSCARES

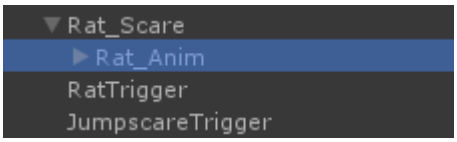
TRIGGER ANIMATION

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

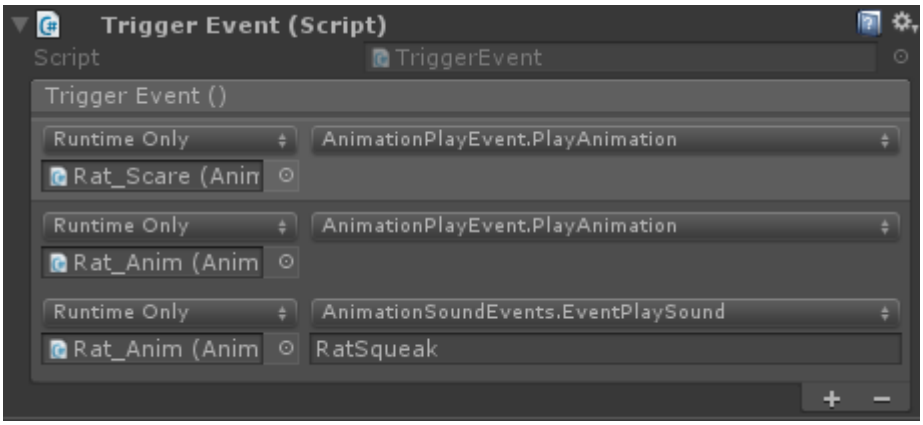
1. FIRST YOU MUST CREATE TRIGGER



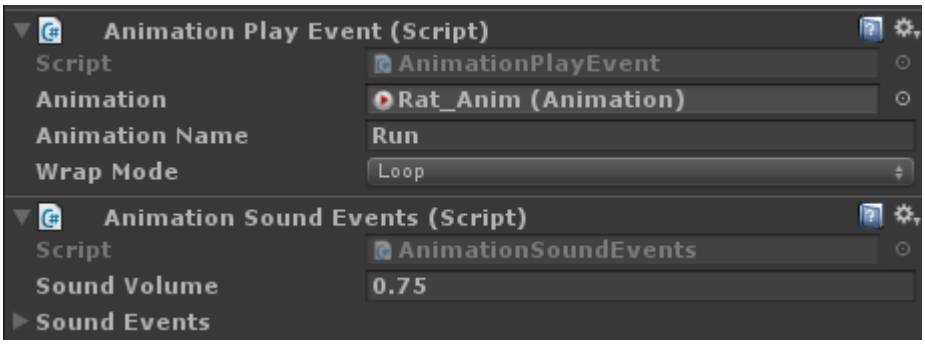
2. THEN CREATE EMPTY GAMEOBJECT AND MOVE CREATURE TO IT



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

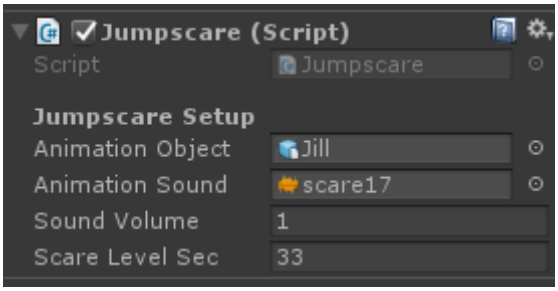


4. CREATE CREATURE MOVE ANIMATION
5. THEN ADD EVENT LISTENERS TO SET WHAT HAPPEND IF YOU GO TO TRIGGER IN MY CASE I ADDED ANIMATION AND SOUND PLAY EVENT

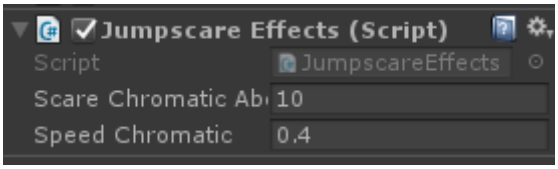


JUMPSCARE ANIMATION

- JUMPSCARE 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
- **JUMPSCARE EFFECTS** CONTROL CAMERA SHAKE, SCARED BREATH AND CHROMATIC ABERATION

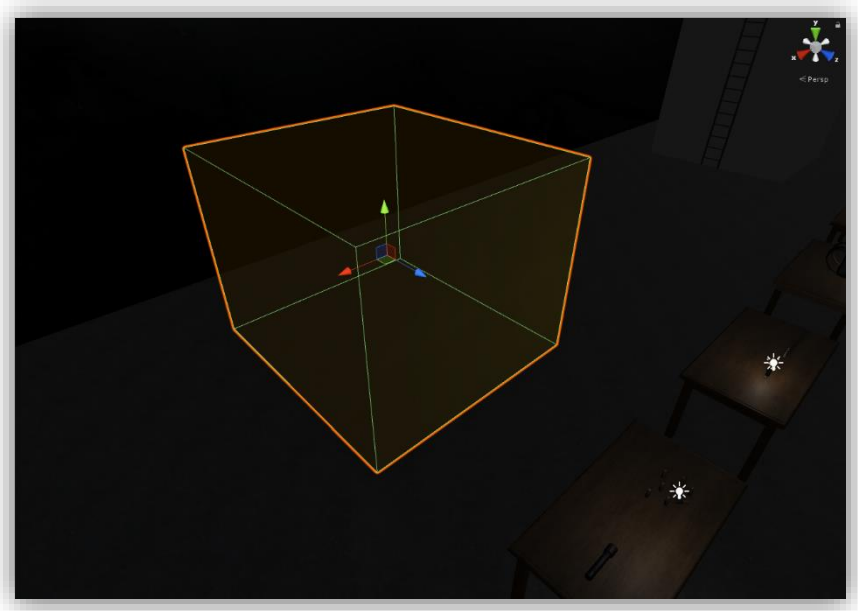


- IF YOU HAVE PROBLEMS WITH CREATING JUMPSCARE ANIMATION YOU CAN GO TO MY YOUTUBE CHANNEL AND WATCH MY JUMPSCARE TUTORIAL: **JUMPSCARE TUTORIAL**

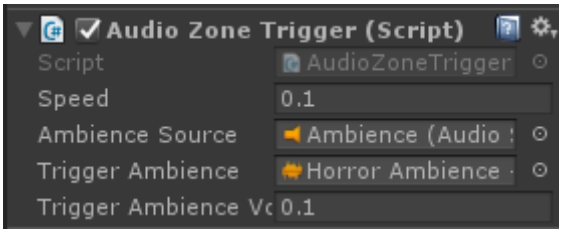
AUDIO ZONE TRIGGER

- THIS IS GOOD FOR CHANGING BACKGROUND AUDIO IF YOU GOING TO A OTHER ROOM

1. FIRST CREATE TRIGGER



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



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

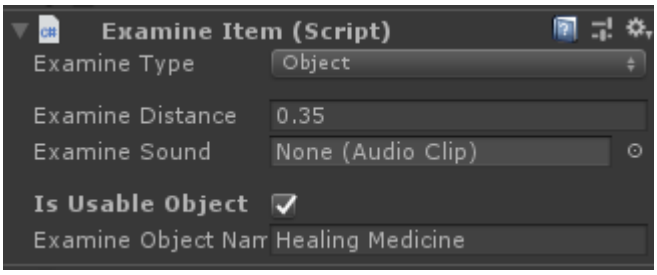
IF YOU WANT TO CHANGE STARTING AMBIENCE GO TO **FPSPLAYER -> SOUND EFFECTS -> AMBIENCE** AND DRAG YOUR AMBIENCE SOUND TO A AUDIO SOURCE

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

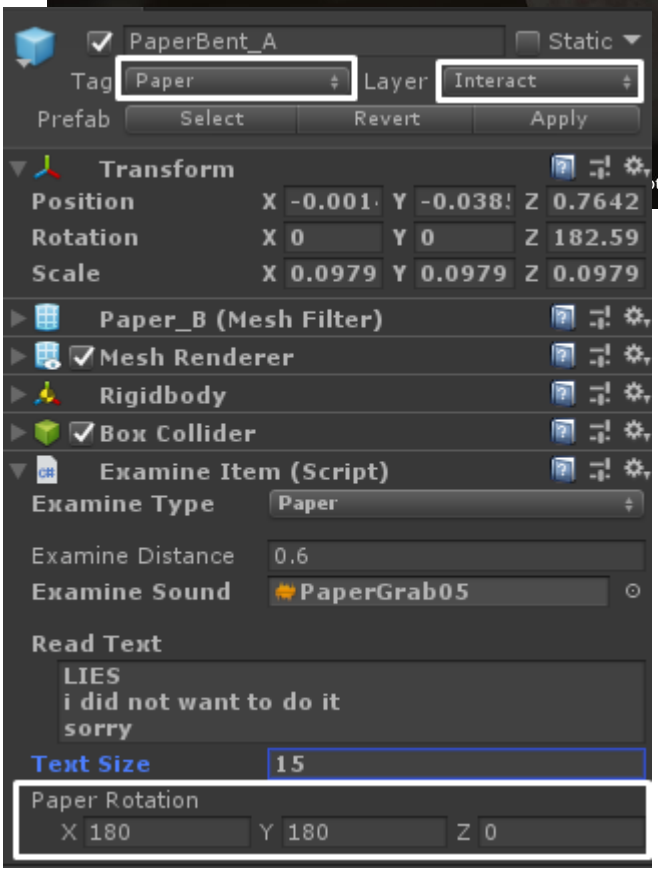


ADDING NEW PAPERS

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

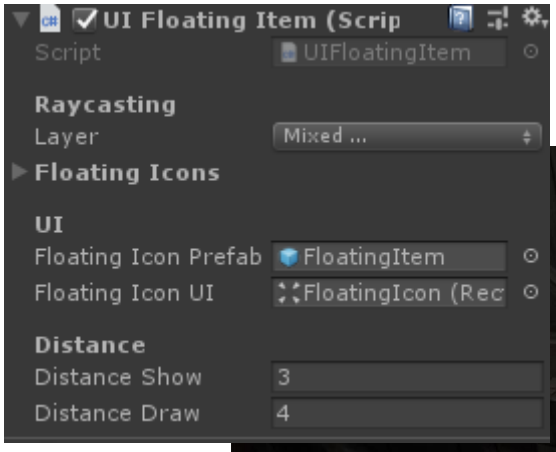
1. JUST DRAG AND DROP **ExamineItem.cs** SCRIPT TO PAPER
2. THEN CHANGE PAPER TAG TO "**Paper**" and LAYER TO "**Interact**"
3. MAIN PART IS SET **PAPER ROTATION** TO CORRECT ROTATION WHEN YOU EXAMINE PAPER
4. THEN YOU CAN CHANGE PAPER READ TEXT AND DISTANCE GRAB

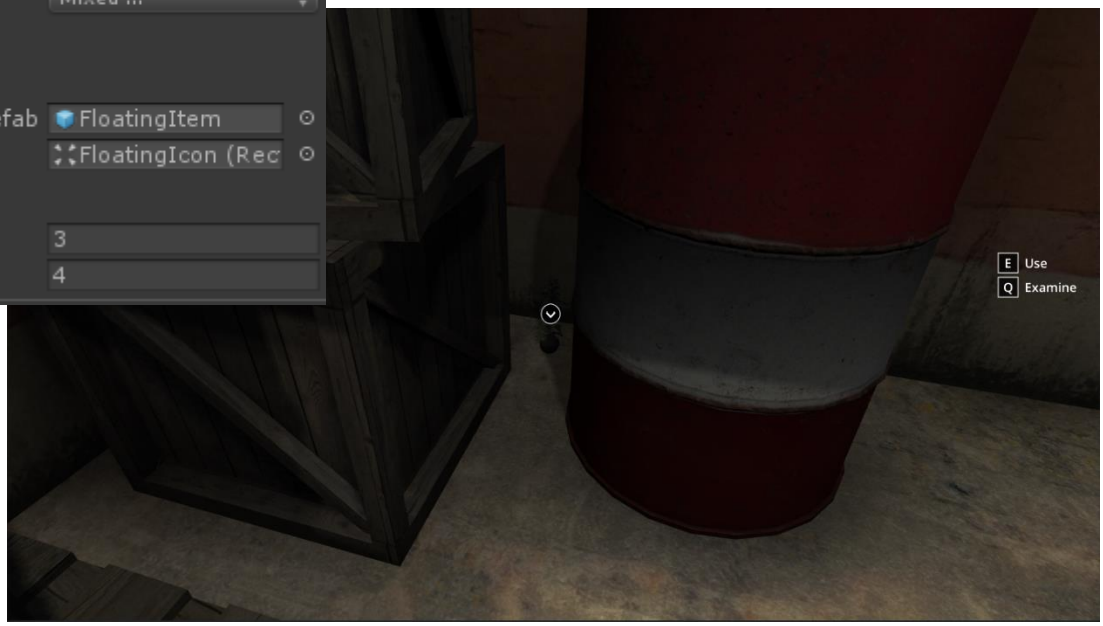


FLOATING ICON

- THIS SCRIPT IS LOCATED IN GAMEMANAGER

TO ADD NEW ICON JUST SELECT GAMEOBJECT WHICH YOU WANT TO ADD INTO FLOATING ICONS AND SELECT **TOOLS - > HFPS KIT -> ADD FLOATINGICON**





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

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

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

THANKS TO A **Francesco Vecchio** WHO PROVIDED ME SOME BACKGROUND MUSIC.