HORROR SYSTEM

Hello!

Thank you for purchasing our project. We hope that this asset will help you in creating your projects. And we would be pleased if you left a review about the project on the Asset Store.

All the best! :)

Subscribe to our social networks:

https://twitter.com/vexkan

https://www.instagram.com/vexkan/

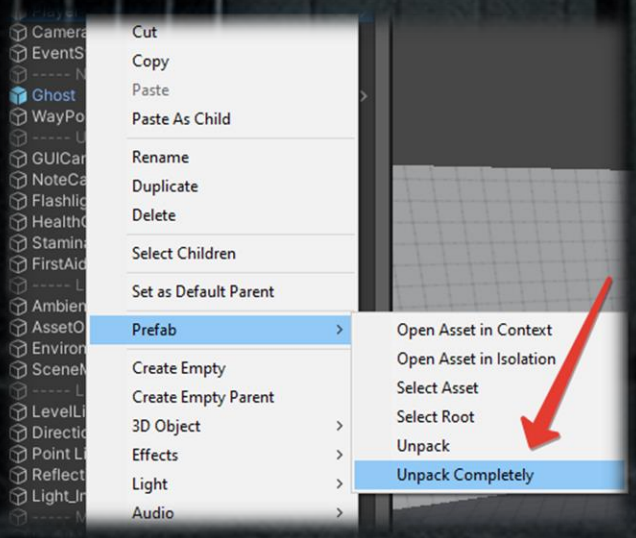
*** This guide tells you how to get started with an asset quickly, but if you have any additional questions or have any difficulties, please contact us by mail: vexkan@gmail.com**

Control:

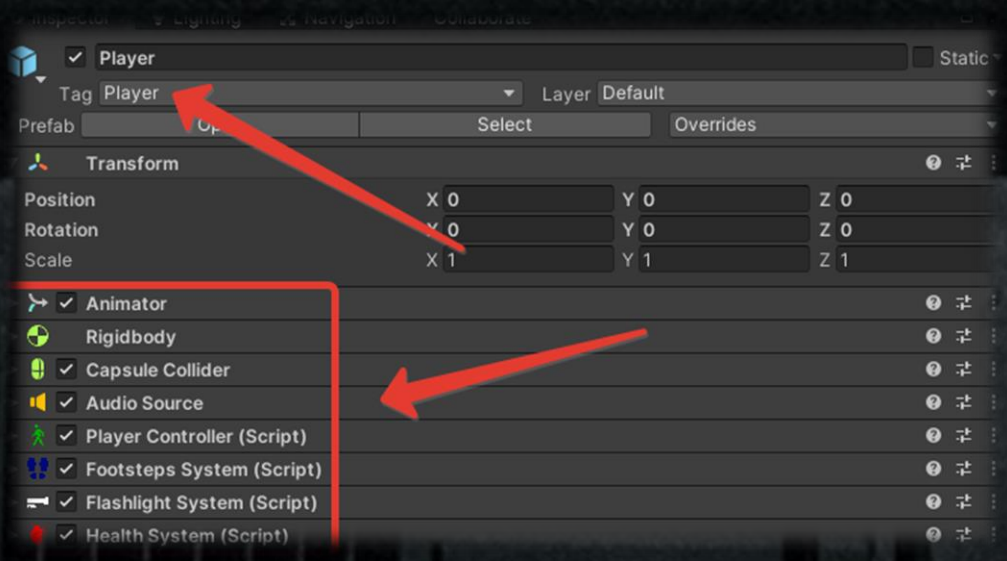
- **E – interact with object;**
- **F – turn on the flashlight;**
- **Y – treat;**
- **R – reload battery;**
- **T – disable/enable Mesh Render.**
- **1 - Machete**

CHARACTER

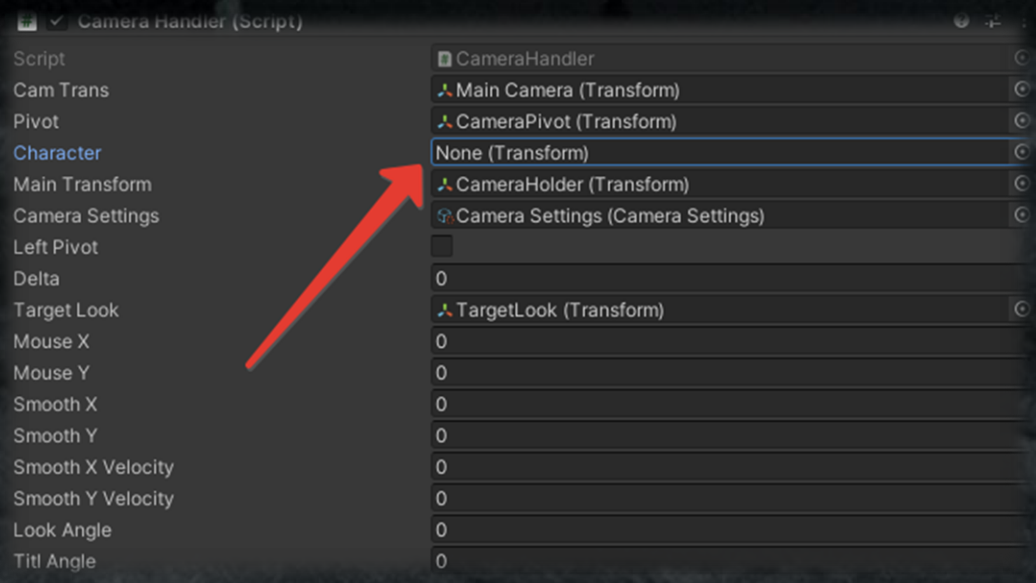
Place your character on the scene at zero coordinates without deleting the test character.
Unpack the test character prefab.



Transfer the flashlight from the test character and place it in the character's brush, as well as the customized components and animator using copy-paste.

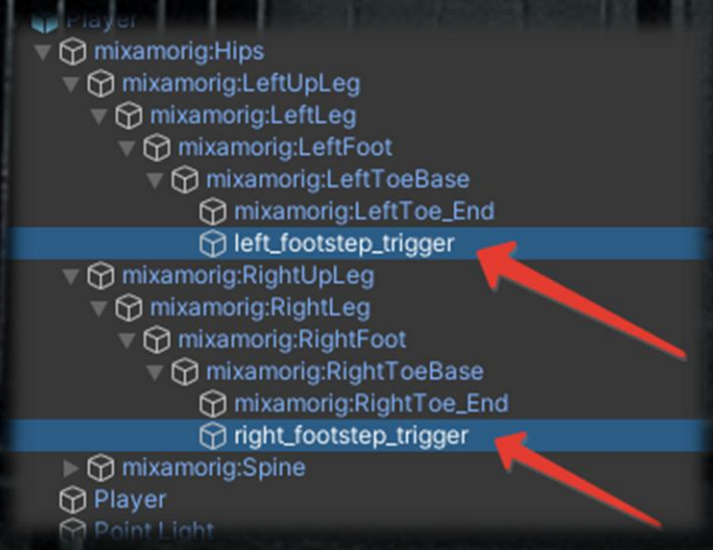


In Camera Holder, replace the character.

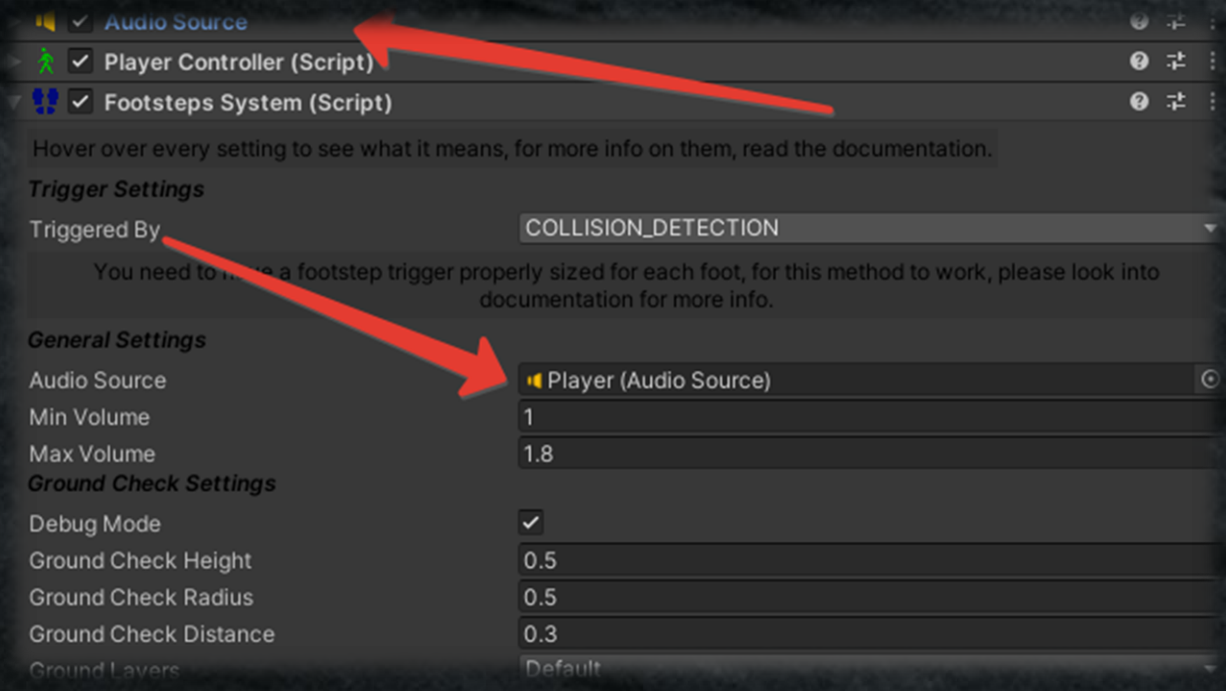


FOOTSTEPS SYSTEM

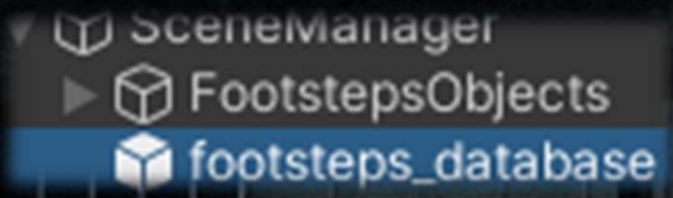
Transfer c from the test character to your left_footstep_trigger and right_footstep_trigger.



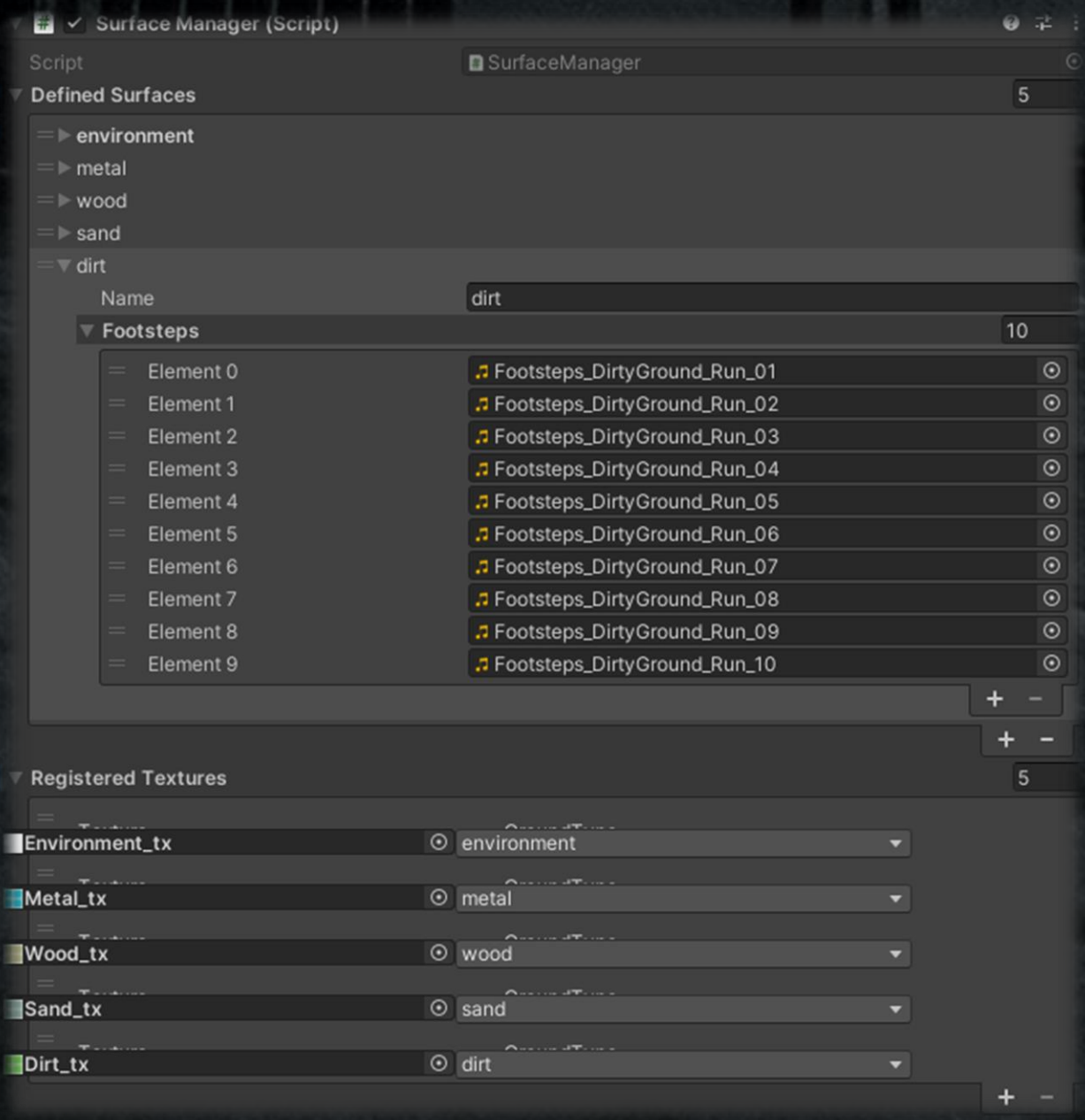
Add the AudioSource component to the character and transfer the character to the Footsteps System script.



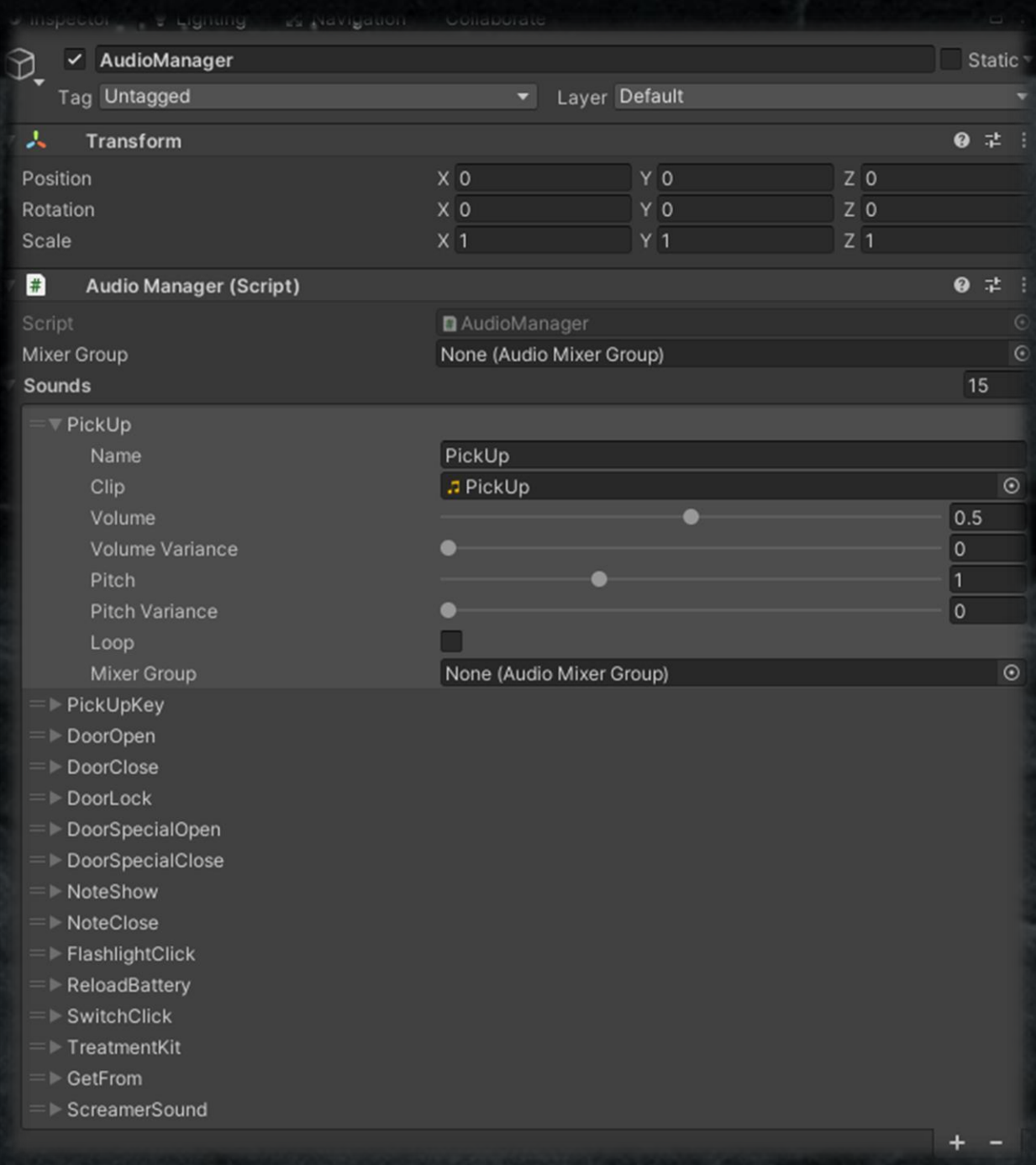
Find footsteps_database on the scene.



There is a Surface Manager script on it that allows you to change the sounds of steps, as well as the texture for a certain surface.



AUDIO MANAGER



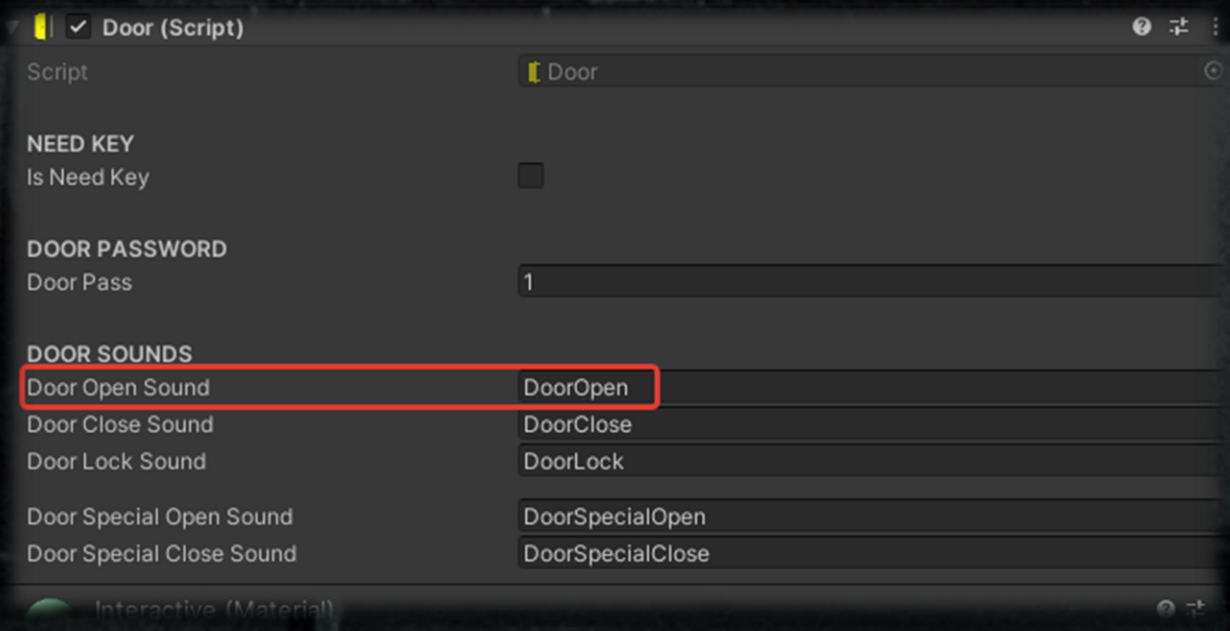
In some scripts, the AudioManager component is called, for example, in the Door script:

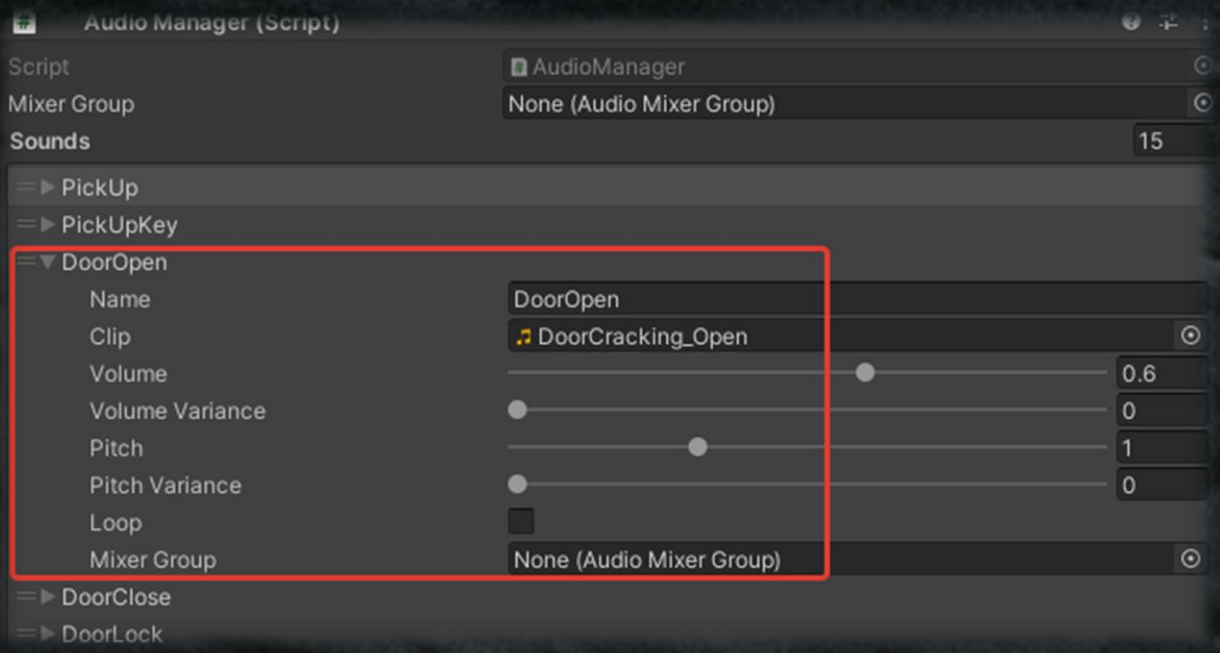
```
    public void OpenSound()
    {
        AudioManager.instance.Play(this.doorOpenSound);
    }
```

In order for this system to work, you need to connect the library "using Interaction System;".

```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Door : Item
{
    [Header("NEED KEY")] //Do I need a key?
    [Space(10)]
    [SerializeField] private bool isNeedKey;

    [Header("DOOR PASSWORD")] //The password for the door (mu
    [Space(10)]
    [SerializeField] private string doorPass;

    [Header("DOOR SOUNDS")]
    [Space(10)]
    [SerializeField] private string doorOpenSound;
    [SerializeField] private string doorCloseSound;
    [SerializeField] private string doorLockSound;

    [Space(10)]
    [SerializeField] private string doorSpecialOpenSound;
    [SerializeField] private string doorSpecialCloseSound;
```

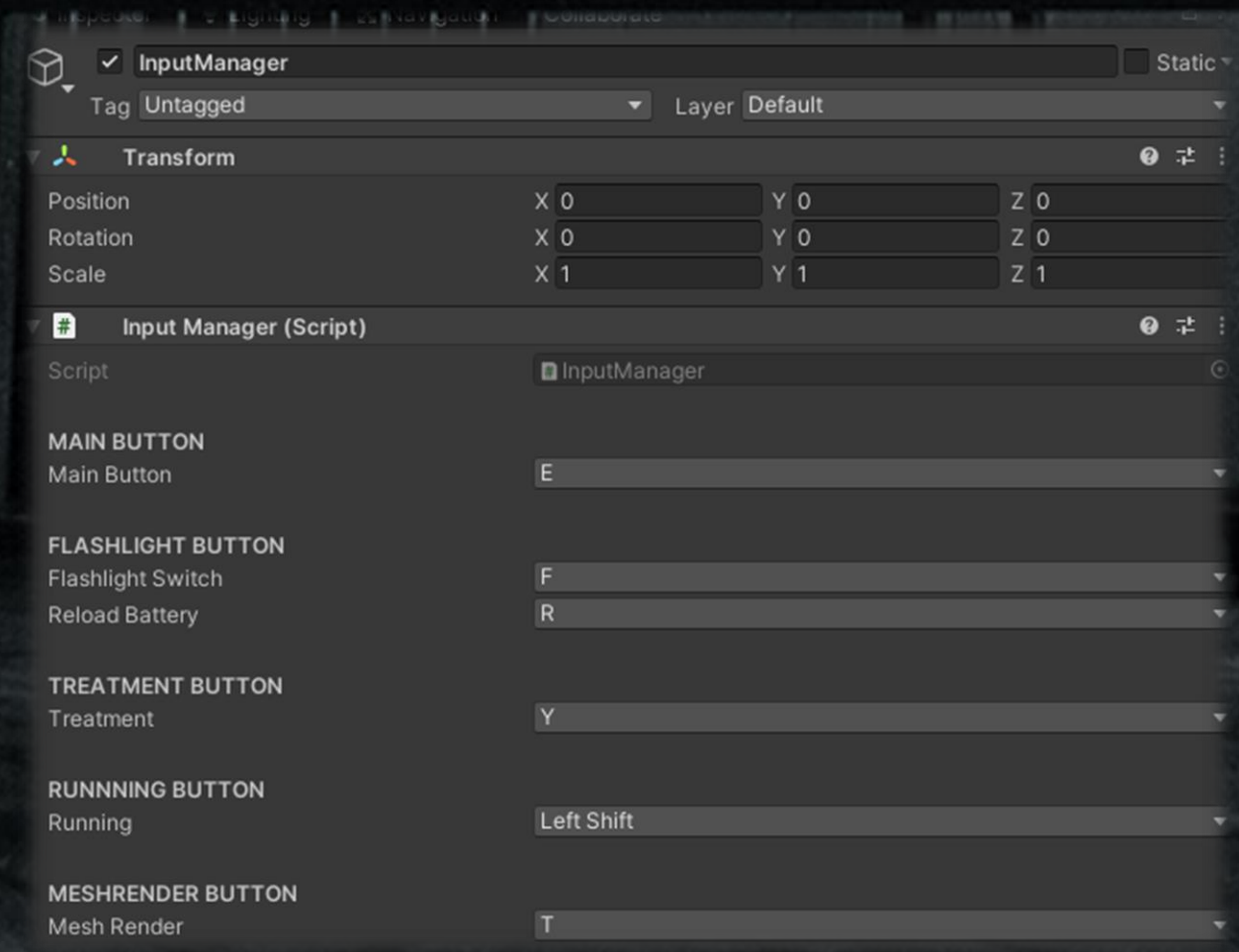
After that, in the Door script, we can add the name of the desired sound effect, which we added to Audio Manager.





INPUT MANAGER

Works by analogy with Audio Manager:

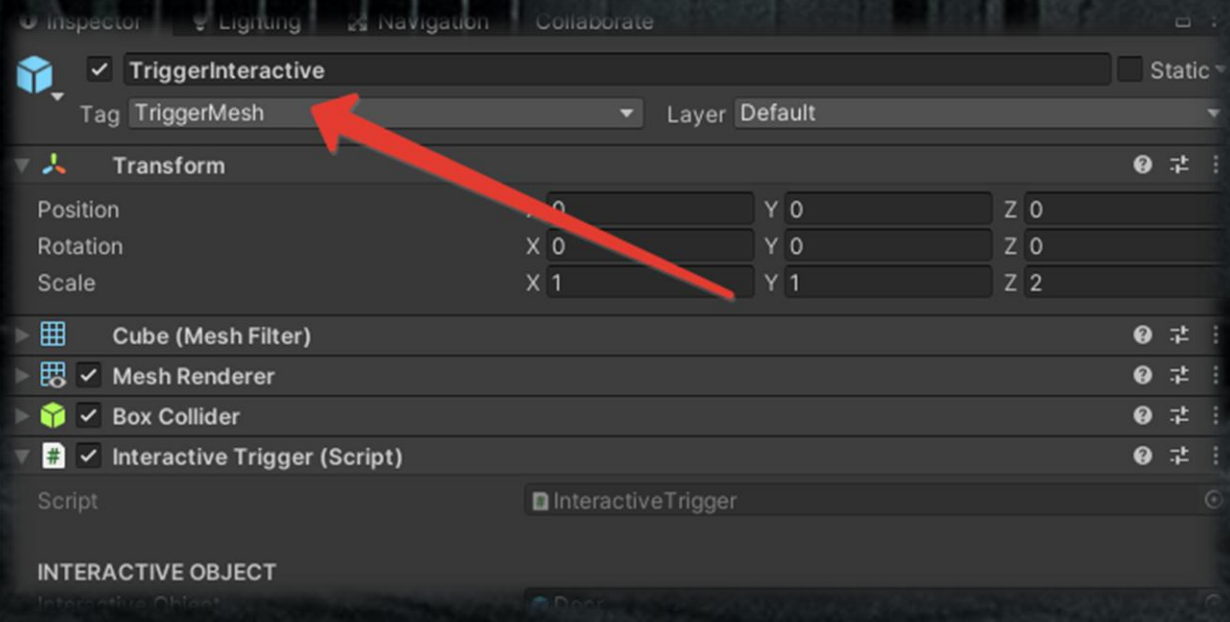


```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class InteractiveTrigger : MonoBehaviour
{
    [Header("INTERACTIVE OBJECT")]
    [Space(10)]
    [SerializeField] public GameObject interactiveObject;
    [SerializeField] public GameObject interact;

    public bool allowInteraction;


    private void Start()
    {
        allowInteraction = false;
        interact.SetActive(false);
    }

    private void Update()
    {
        if (Input.GetKeyDown(InputManager.instance.mainButton) && allowInteraction)
        {
            interactiveObject.GetComponent<Item>().ActivateObject();
        }

        if (interactiveObject == null)
        {
            Destroy(gameObject);
        }
```

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

namespace InteractionSystem
{
    public class InputManager : MonoBehaviour
    {
        [Header("MAIN BUTTON")]
        [Space(10)]
        public KeyCode mainButton;
```

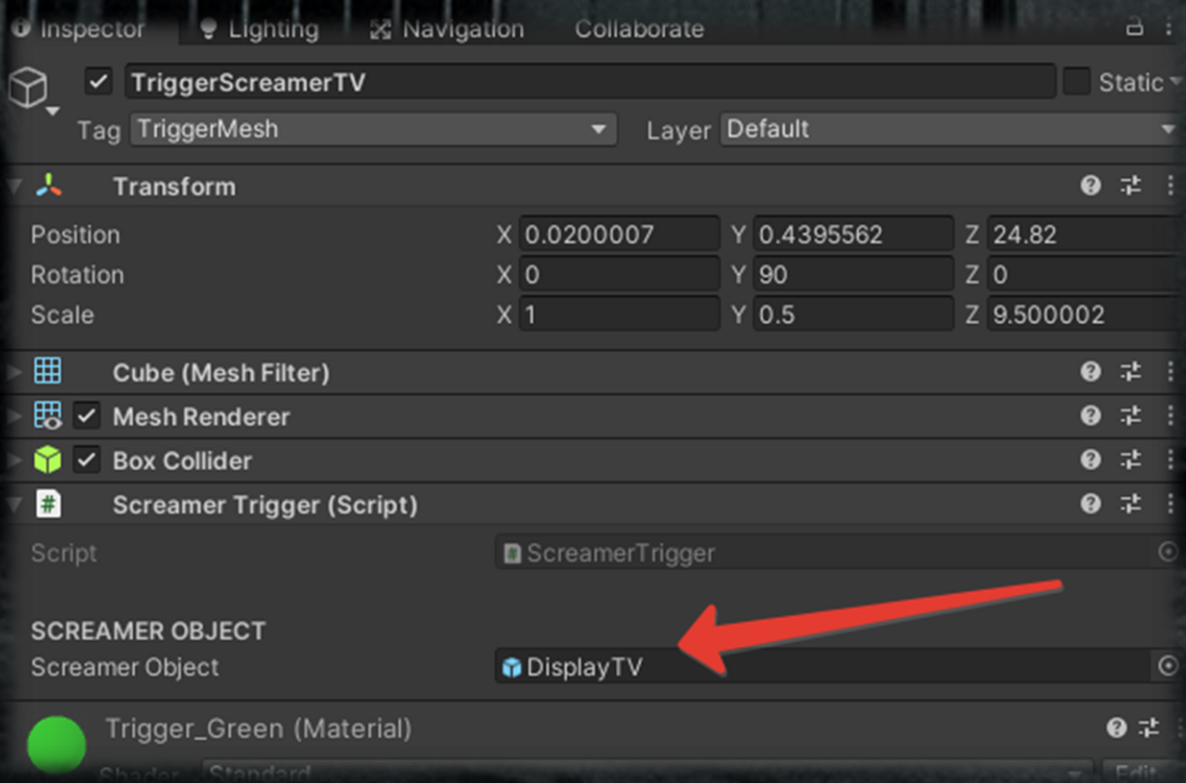
DISABLE MESH RENDER

The Disable Mesh script is used to disable and Mesh Renderer for triggers. The script performs a purely test function for convenient work with the project. The "TriggerMesh" tag is set on triggers. When the scene starts, the script searches for all objects with this tag, and when the T button is pressed, it disables/enables Mesh Renderer.

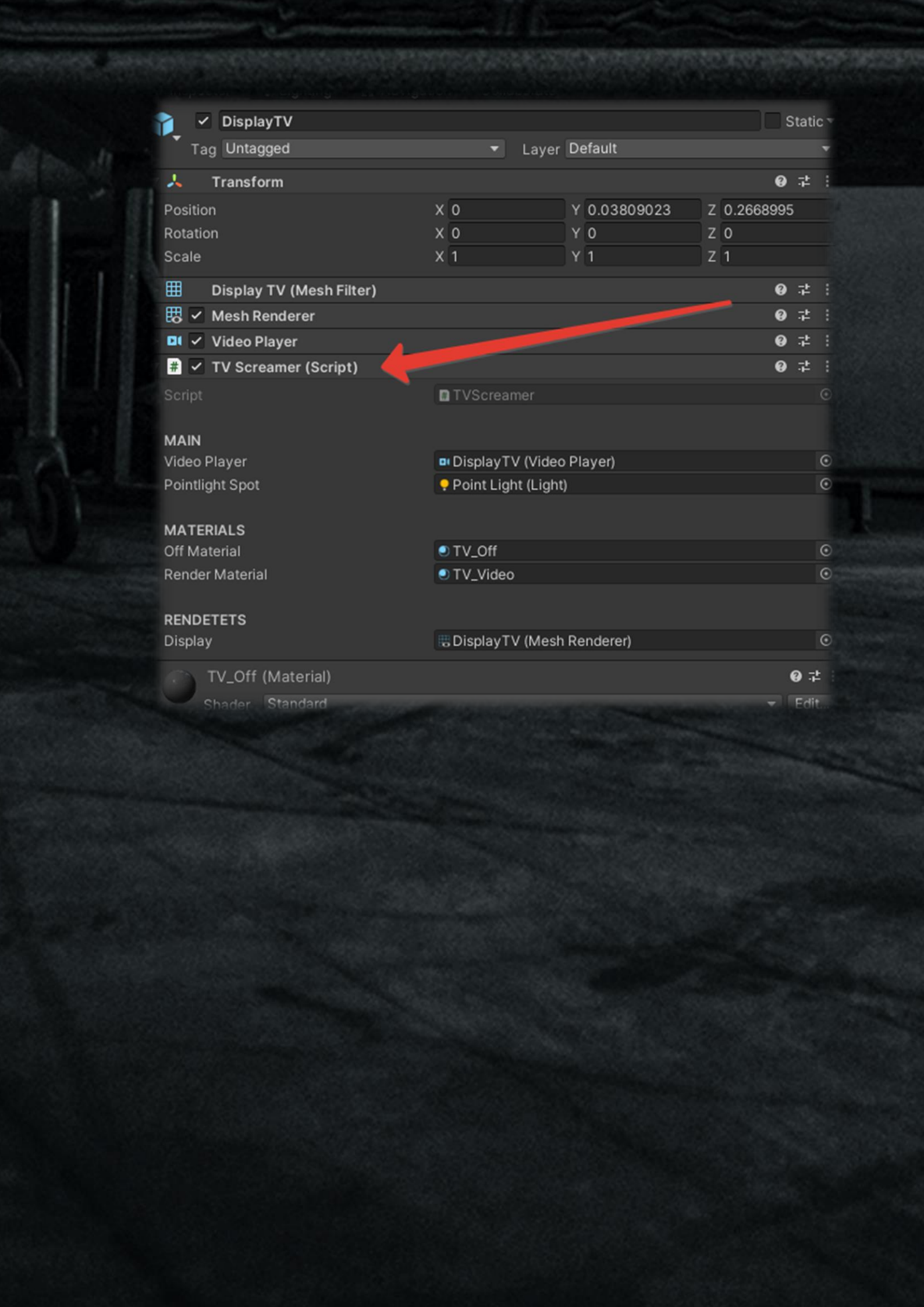


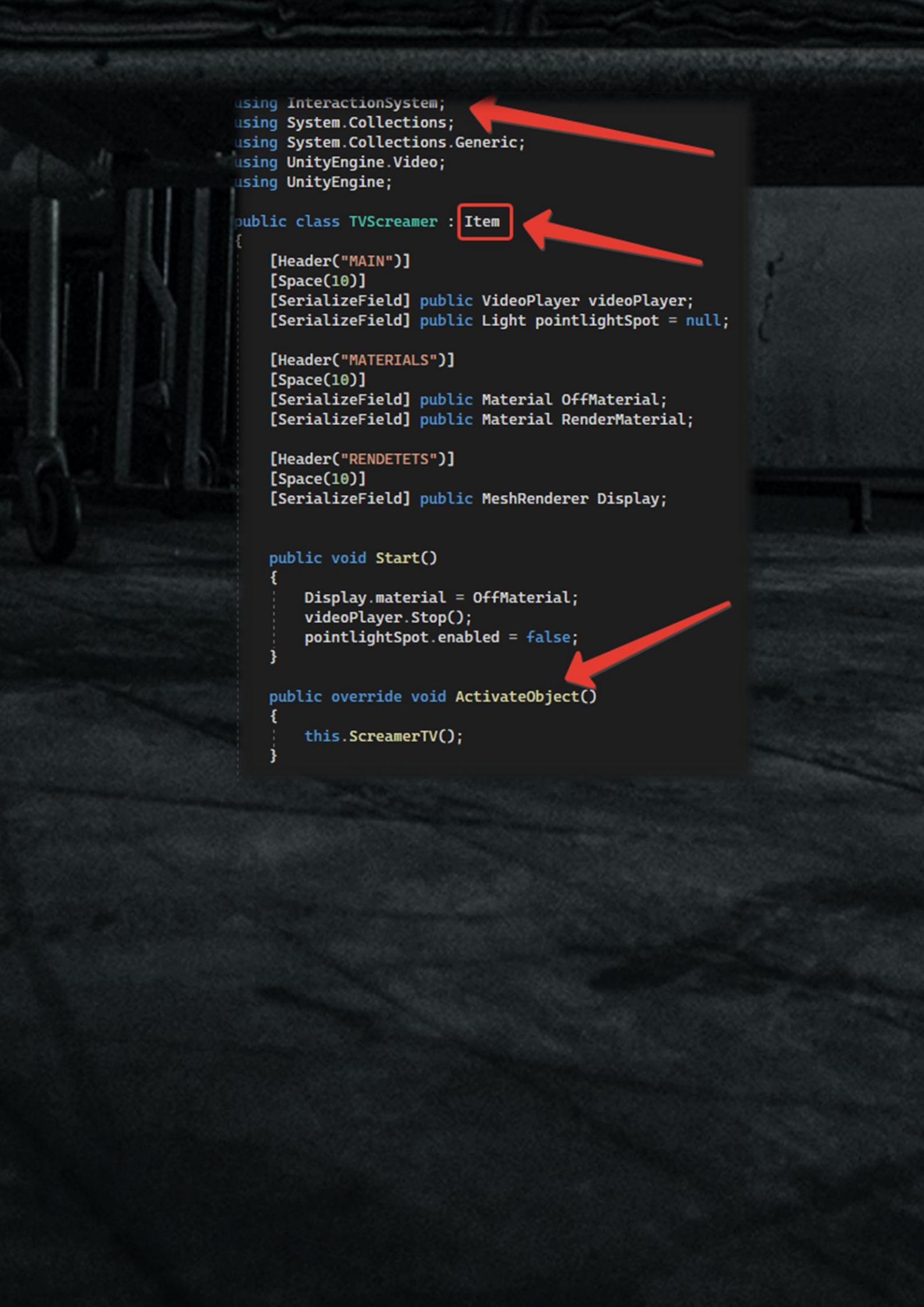
SCREAMERS

In order for the streamers to work, we must enter the trigger that will trigger an event for the object.




```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ScreamerTrigger : MonoBehaviour
{
    [Header("SCREAMER OBJECT")]
    [Space(10)]
    [SerializeField] public GameObject screamerObject;


    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            screamerObject.GetComponent<Item>().ActivateObject();
        }
    }
```

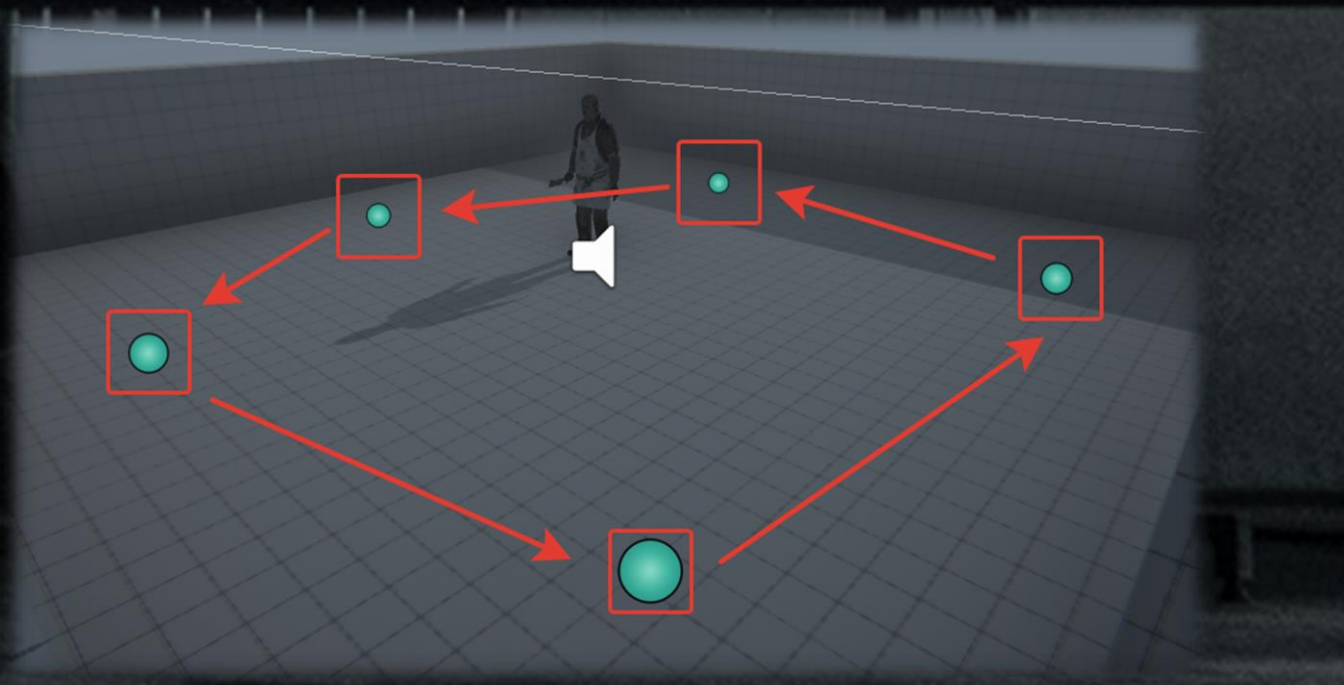

DisplayTV
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0.03809023 Z 0.2668995
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Display TV (Mesh Filter)
Mesh Renderer
Video Player
TV Screamer (Script)
Script TVScreamer
MAIN
Video Player DisplayTV (Video Player)
Pointlight Spot Point Light (Light)
MATERIALS
Off Material TV_Off
Render Material TV_Video
RENDETETS
Display DisplayTV (Mesh Renderer)
TV_Off (Material)
Shader Standard
Edit

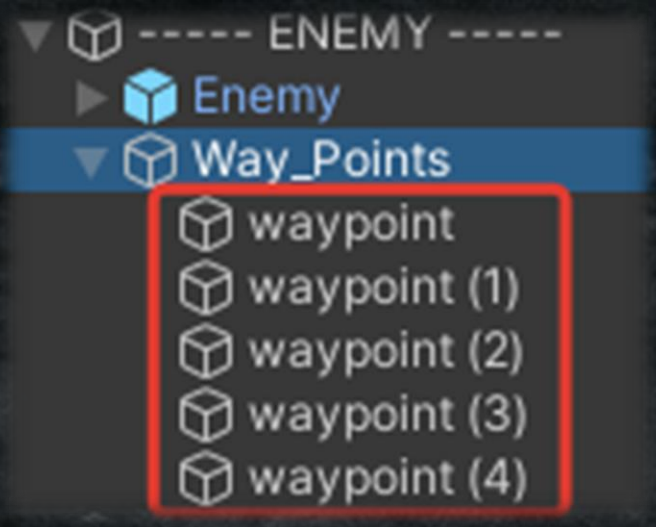


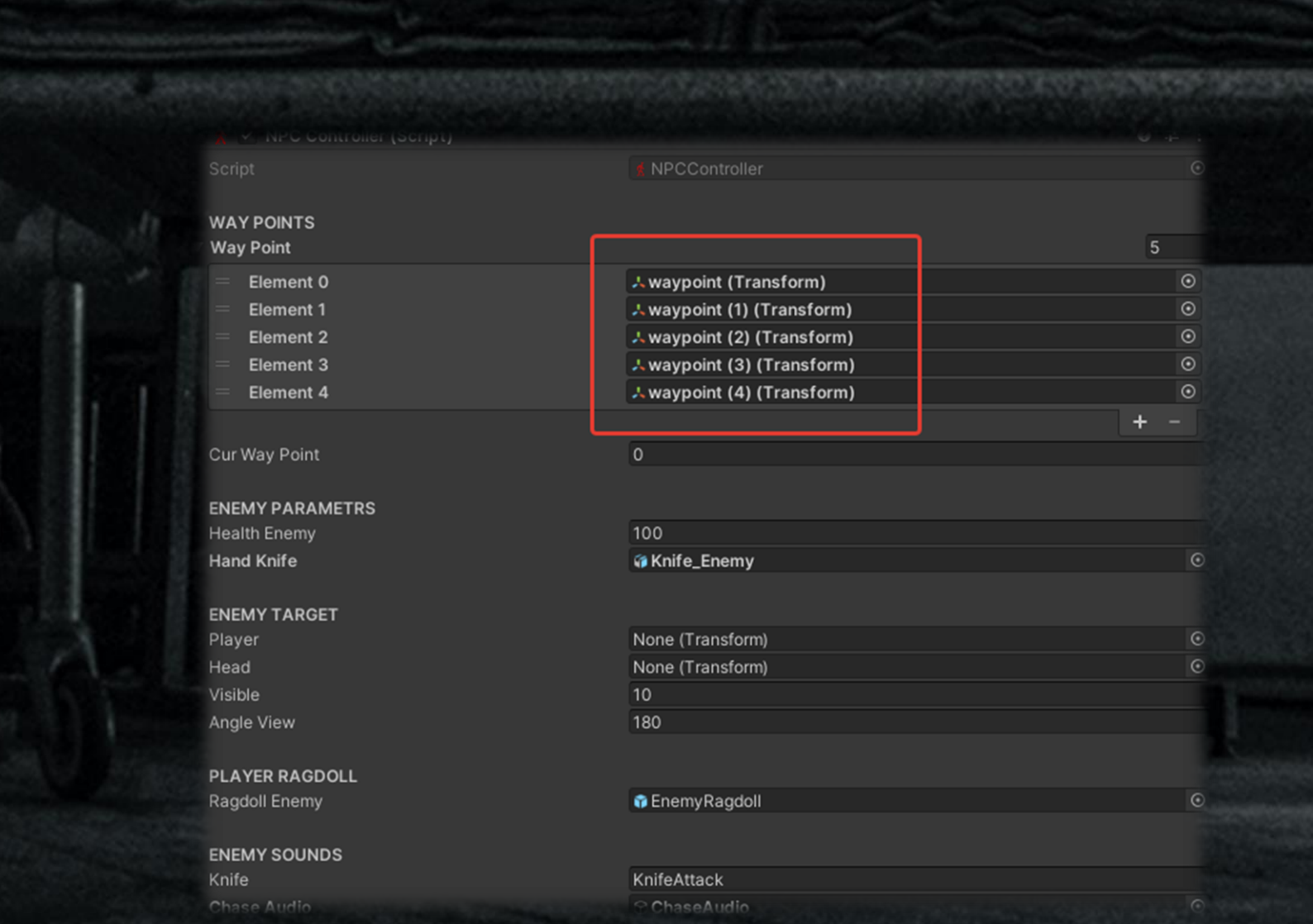
```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.Video;
using UnityEngine;

public class TVScreamer : Item
{
    [Header("MAIN")]
    [Space(10)]
    [SerializeField] public VideoPlayer videoPlayer;
    [SerializeField] public Light pointlightSpot = null;

    [Header("MATERIALS")]
    [Space(10)]
    [SerializeField] public Material OffMaterial;
    [SerializeField] public Material RenderMaterial;

    [Header("RENDETETS")]
    [Space(10)]
    [SerializeField] public MeshRenderer Display;


    public void Start()
    {
        Display.material = OffMaterial;
        videoPlayer.Stop();
        pointlightSpot.enabled = false;
    }

    public override void ActivateObject()
    {
        this.ScreamerTV();
    }
```

ENEMY

While the NPC does not see the player, he patrols from one point to another:



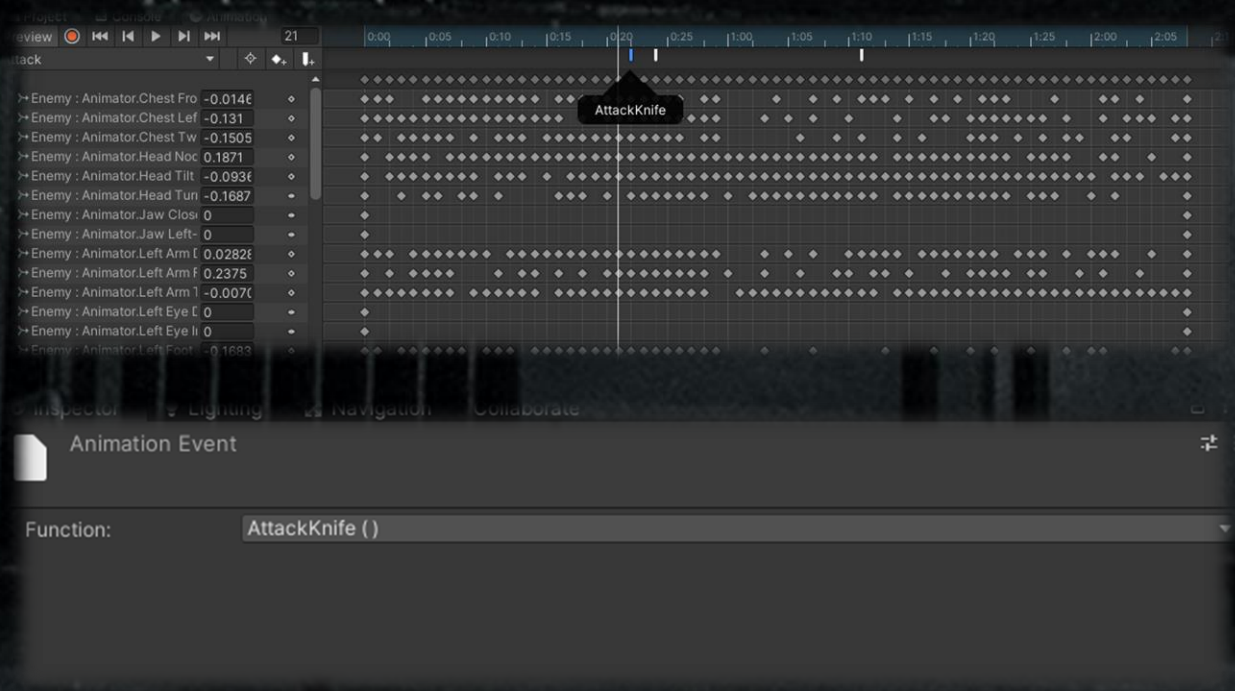




There can be an unlimited number of WayPoints. If there is one WayPoint, then the enemy will stand still.

As soon as we get into the NPC's line of sight, she starts following us.

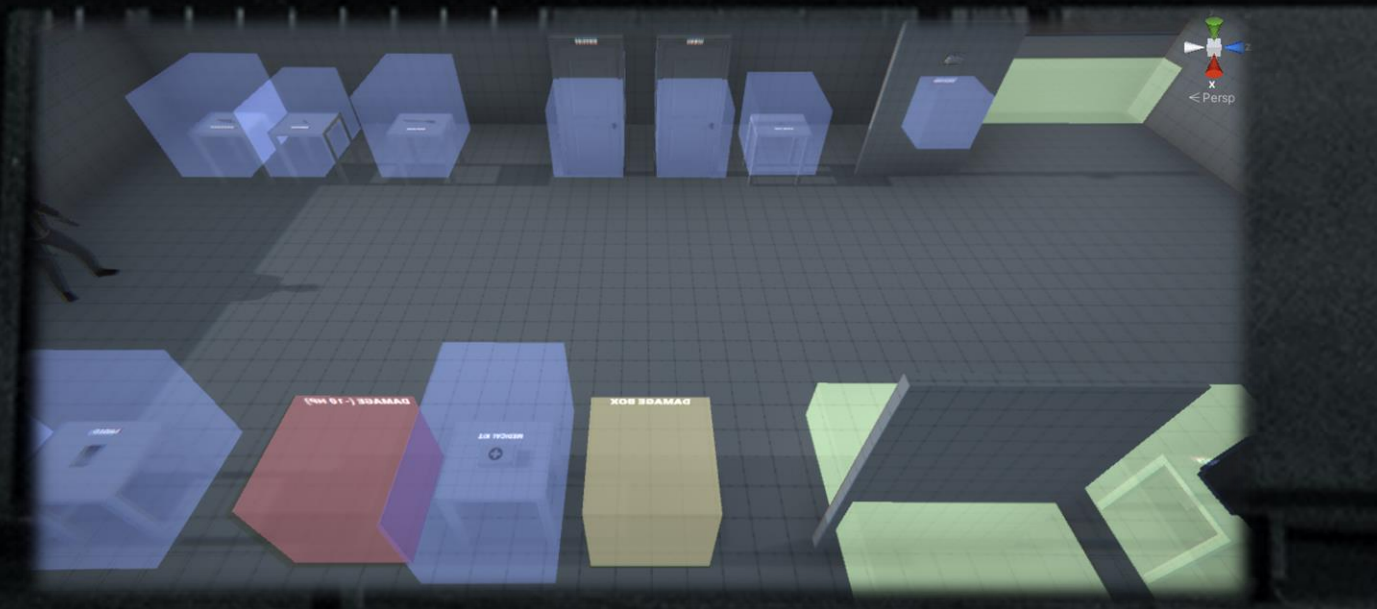
There are several evens on the attack animation that are responsible for the sound effect, enabling and disabling the trigger.



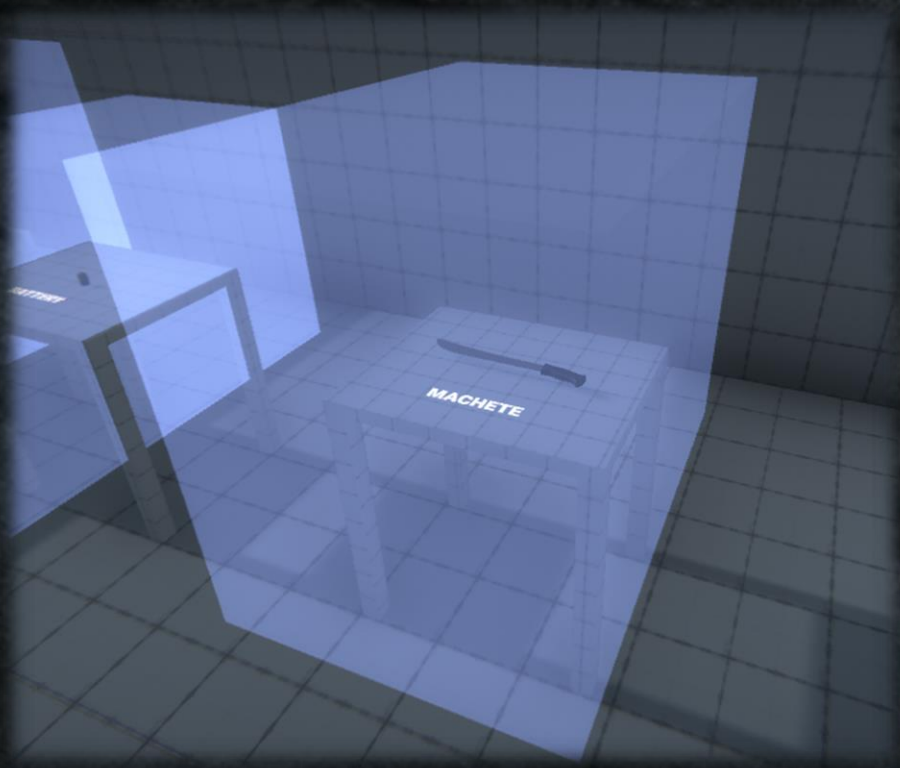

```
18
19      void AttackKnife()
20      {
21          AudioManager.instance.Play(knife);
22      }
23
24      public void TriggerEnable()
25      {
26          handKnife.GetComponent<Collider>().enabled = true;
27      }
28
29      public void TriggerDisable()
30      {
31          handKnife.GetComponent<Collider>().enabled = false;
32      }
33  }
```

INTERACTIVE

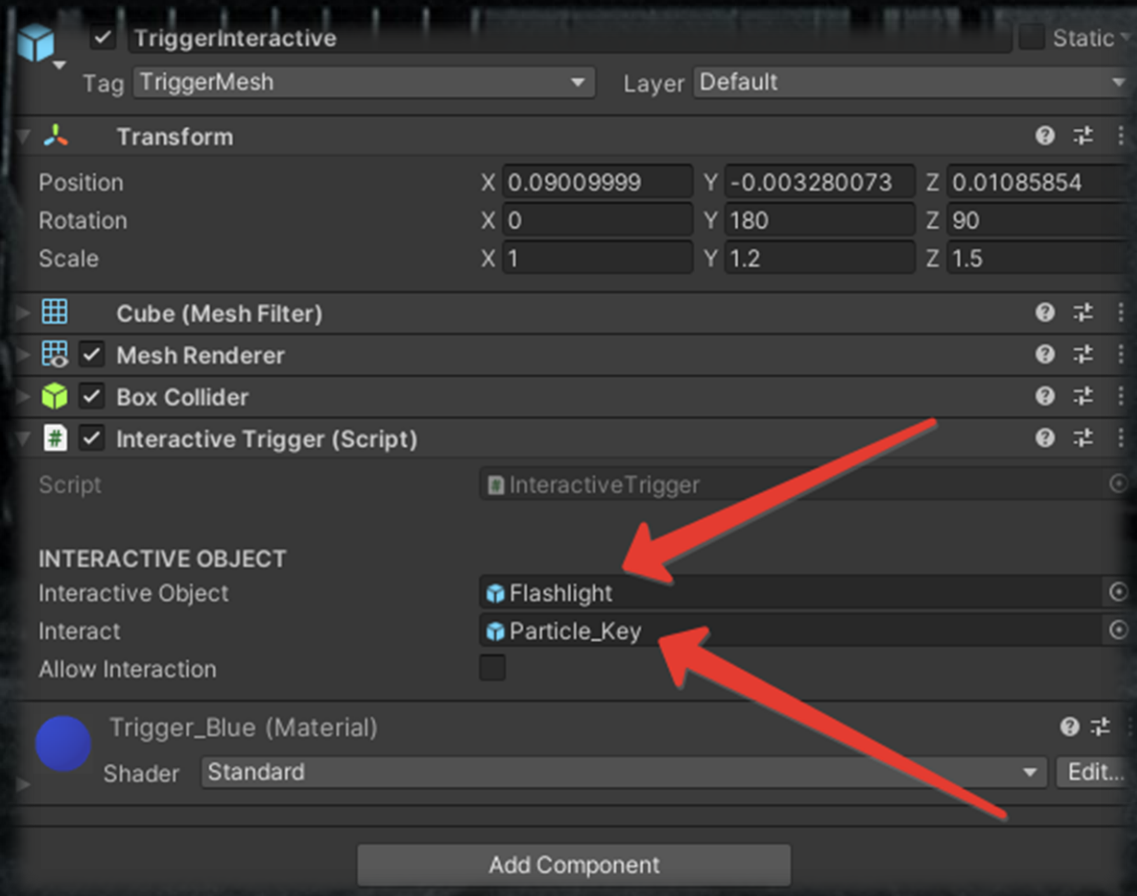
There are several objects on the stage that you can interact with.



An interactive object consists of two elements: the object itself and a trigger in which we can interact with the object.



The Interactive Trigger script is located on the trigger, which specifies the interaction object and the interactivity icon that appears when the player is in the trigger.



The script works like this: when we are in the trigger, we press the button and the object starts an action.

```
private void Update()
{
    if (Input.GetKeyDown(InputManager.instance.mainButton) && allowInteraction)
    {
        interactiveObject.GetComponent<Item>().ActivateObject();
    }

    if (interactiveObject == null)
    {
        Destroy(gameObject);
    }
}
```

All scripts for interactive objects presented in this asset have the ActivateObject() method.

Also, these scripts must have the library "using Interaction System;" connected.

And also have an Item class.

Battery

```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Battery : Item
{
    [Header("BATTERIES COUNT")]
    [Space(10)]
    [SerializeField] private int count;


    public override void ActivateObject()
    {
        FlashlightSystem.instance.CollectBattery(this.count);
        this.DestroyObject(0);
    }
```

Key

```
using InteractionSystem;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Key : Item
{
    [Header("KEY PASSWORD")] //Key password (must match t
    [Space(10)]
    [SerializeField] private string keyPass;

    [Header("KEY SOUND")]
    [Space(10)]
    [SerializeField] private string pickUpKey;

    public override void ActivateObject()
    {
        this.AddKey();
    }
```

And others

DISABLE MANAGER

When selecting a note in the Note script, we refer to the Disable Manager script in which we disable the animator player controller, player controller and camera.

Note

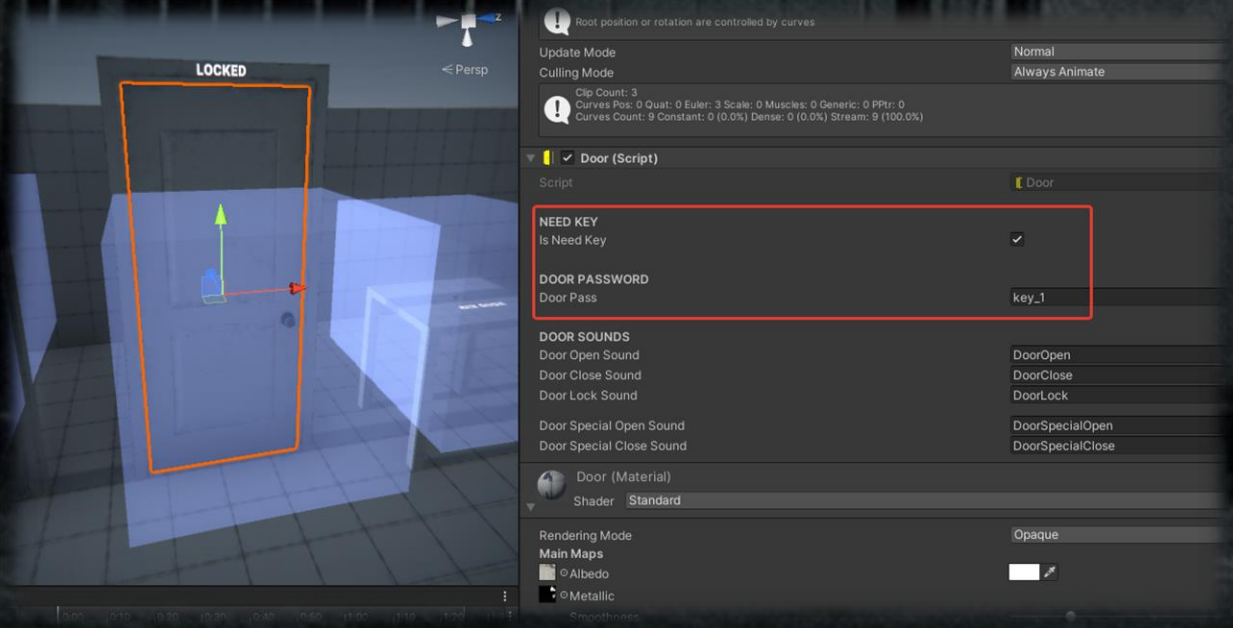
```
public void ShowCloseNote()
{
    if (this.isOpenNote)
    {
        this.isOpenNote = false;
        noteImage.enabled = false;
        AudioManager.instance.Play(noteCloseSound);
        DisableManager.instance.DisablePlayer(false);
    }
    else
    {
        this.isOpenNote = true;
        noteImage.enabled = true;
        AudioManager.instance.Play(noteShowSound);
        DisableManager.instance.DisablePlayer(true);
    }
}
}
```

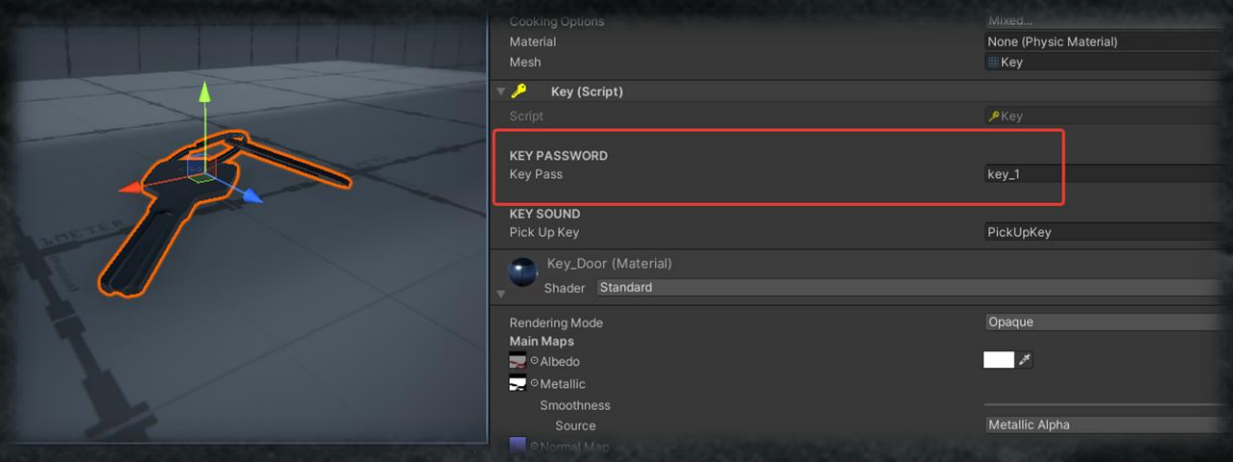
Disable Manager

```
public void DisablePlayer(bool disable)
{
    if (disable)
    {
        player.enabled = false;
        animatorPlayer.enabled = false;
        cameraPlayer.enabled = false;
    }

    else
    {
        player.enabled = true;
        animatorPlayer.enabled = true;
        cameraPlayer.enabled = true;
    }
}
```

KEY AND DOOR

One of the doors on the stage is closed. It can be opened only after selecting the key. in the Door script, we specify the password for this door and in the Key script, the exact same password.





After selecting the key, the door automatically becomes open.