Report – Guardian Of The Shadow

Diogo

DEPARTAMENTO **CIÊNCIA E TECNOLOGIA**

Contents

VISUALS

The imported terrain and its objects already have colliders pre-set to prevent the player from passing through them.

Normal vision:

For the environment, I used the "Flooded Grounds" asset from the Unity Store. This asset comes with a default daytime lighting setup. I modified the post-processing settings to convey a nighttime atmosphere.

I added the following effects:

- **Ambient Occlusion:** Enhances the depth and realism of the scene by simulating how light radiates in real life, creating subtle shadows in crevices and around objects.

- **Bloom:** Creates a glowing effect around bright areas, giving the impression of intense light sources and adding a more cinematic feel.

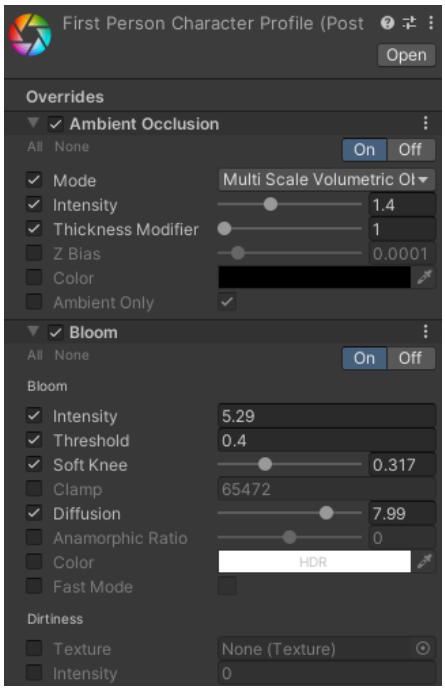


Figura 1-AO and Bloom (Normal)

- **Color Grading:** Adjusts the color tones, contrast, and saturation to achieve a specific mood or aesthetic, enhancing the overall visual atmosphere.

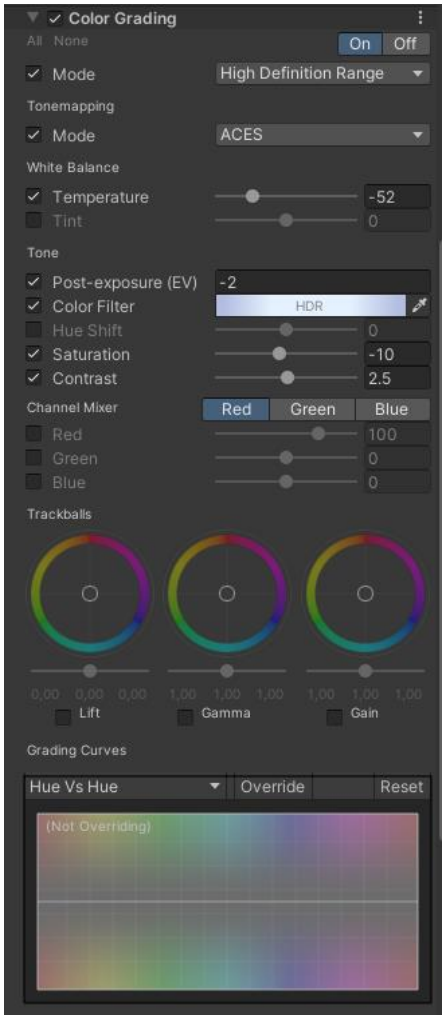


Figura 2-Color Grading (Normal)

- **Grain:** Adds a film grain effect to the image, simulating the texture of traditional film and giving the scene a gritty, realistic look.

- **Vignette:** Darkens the corners of the screen, drawing attention to the center and enhancing the feeling of night.

Figura 3-Grain & Vignette (Normal)

These adjustments help create an immersive and tense nighttime environment.

Figura 4-Normal Vision with Post-Processing

Night Vision:

For the night vision view, I adjusted the same post-processing effects, changing it to give a green ambience.

Additionally, I created custom models in Photoshop specifically for the night vision view and imported them into Unity, placing them in the HUD (on the canvas). I disabled raycasting for these elements to ensure they cannot be clicked.

Also used a script to control the battery duration and the zoom, as well as changing between post processing profiles.

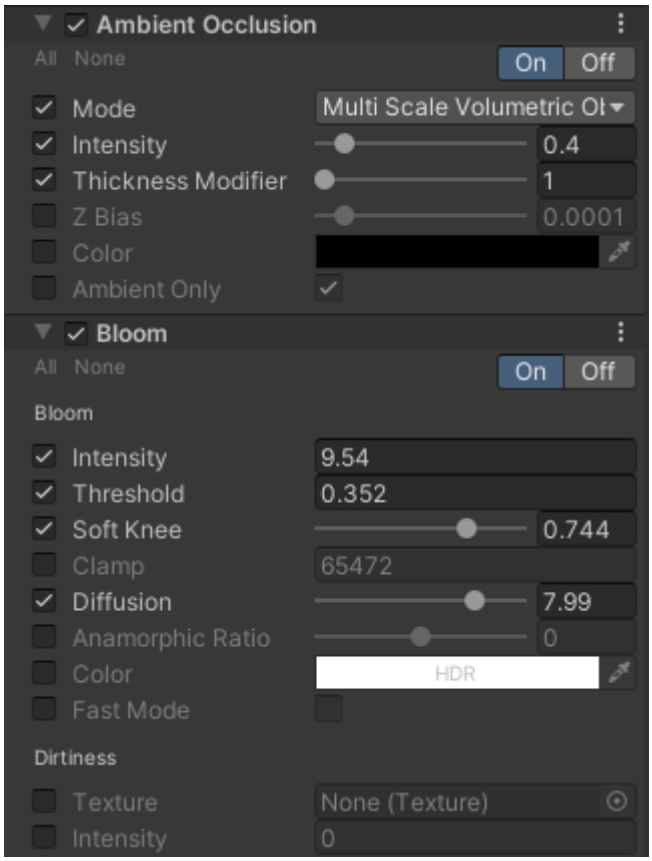


Figura 5-AO and Bloom (Nightvision)

Figura 6- Color Grading (Nightvision)

Figura 7-Grain and Vignette (Nightvision)

Nightvision Overlay

Created the following elements using Photoshop:

Figura 8-Battery Chunks

Figura 9-Battery outer

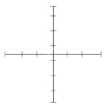


Figura 10- Crosshair

Figura 11- W(ide angle) and T(elephoto)

Figura 12 - Zoom Bar

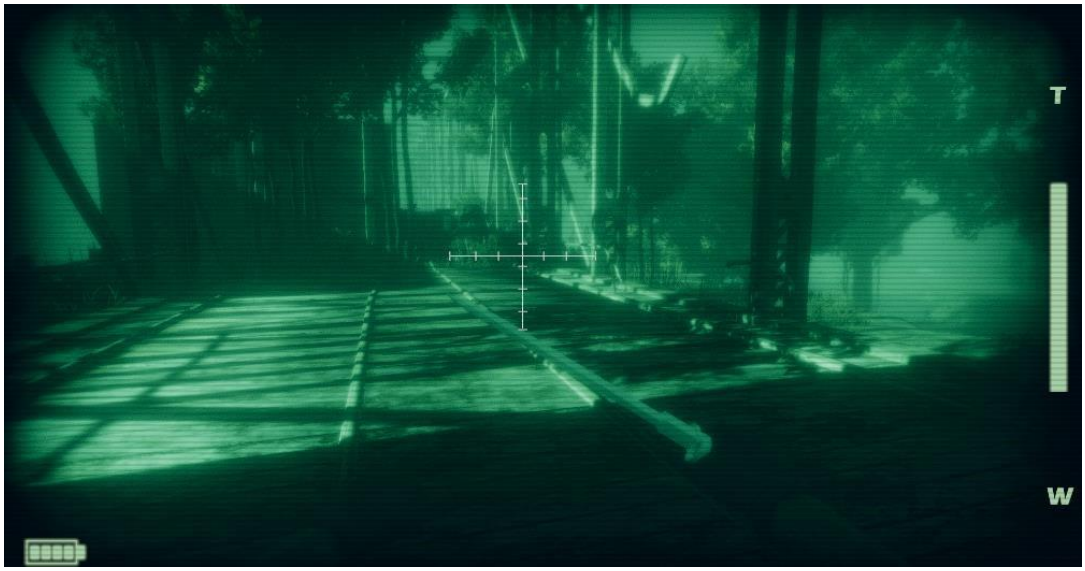


Figura 13- Nightvision

Inventory Vision

For the inventory vision view, I adjusted the same post-processing effects, changing it to give a gray ambience.

I created the weapons and items menu in Photoshop using models from Sketchfab. I also created a script to prevent the character from moving or attacking while the inventory is open (showed in the Script section).

Through another script, I ensure that weapons or items that the player does not possess appear darkened, giving the impression that they are unavailable and preventing them from being selected or

used.

Figura 14- Weapon Inventory

Figura 15- Item inventory

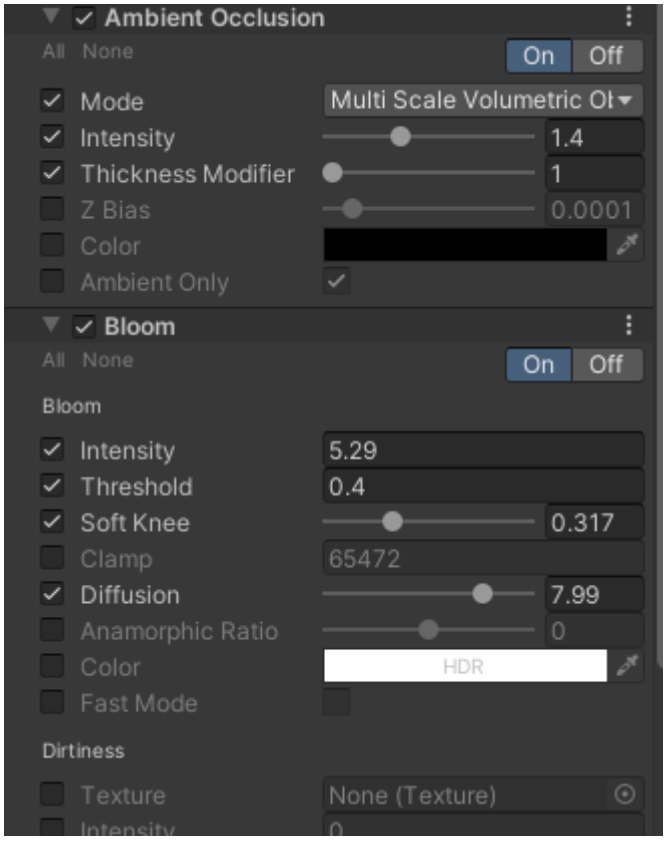


Figura 16-AO and Bloom (Inventory)

Figura 17-Color Grading (Inventory)

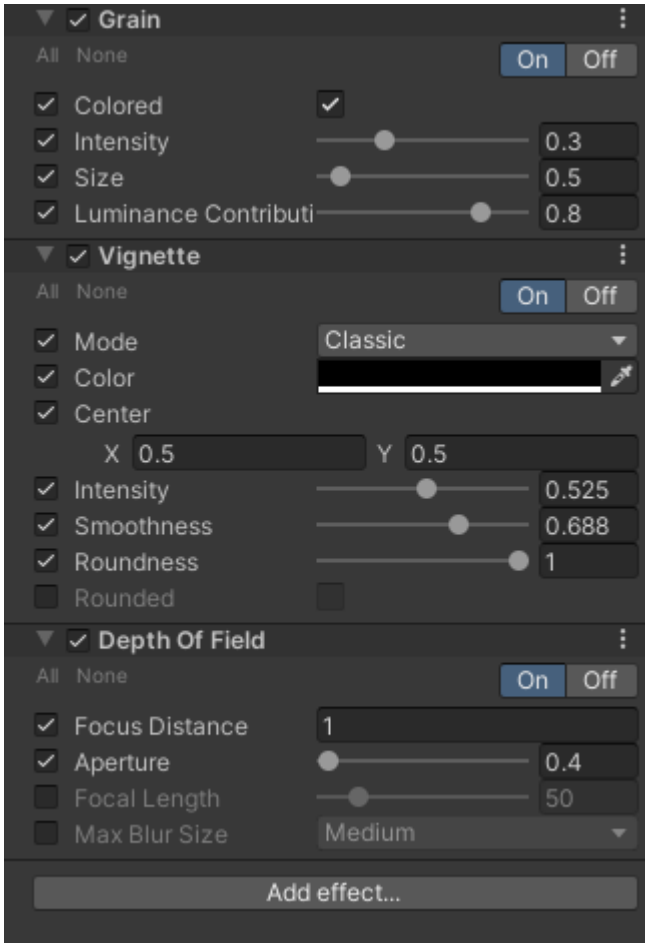


Figura 18-Grain & Vignette (Inventory)

I also made a combination menu and a script that shows If the player has the items needed to combine or not , If the player doesn't have the items needed the use button doesn't appear.

Figura 19-Crafting menu

Flashlight

For the flashlight, I used a spotlight attached to the player. Using the same script as the night vision, the flashlight can be toggled on or off with the "F" key. For the flashlight model, I used a pre-existing cookie from Unity, which helps shape the light beam for a more realistic effect.

The models I used in the HUD are the same as the Night vision, just color changed.

Figura 20- Flashlight

Lighthouse:

For the lighthouse, I used a capsule that emits light in a specified area (with emission enabled in the material). I positioned the capsule slightly outside and at the top of the lighthouse. Using a simple script, the capsule rotates, creating the effect of a rotating lighthouse beam. The lighthouse casts a bluish light, illuminating specific areas both in front of and behind the capsule's position.

Figura 21-Lighthouse

CHARACTERS

Player

Figura 22 - Player model

The player model consists only of the arms, which have animations for all the weapons they hold. The player (First Person Character) has a movement script, a Rigidbody, a Character Controller, a Audio Source and a Audio Listener.

The camera is positioned in the eye-zone of the player .This camera the player holds has a Flare Layer and a Line Renderer attached to it . The flare layer is used to render lens flares, which are bright spots of light that appear when you look at a light source and the Line Renderer is used to draw lines in the game world , in the case of the player It's used to make a line to throw the Molotov .

The First Person Character is an empty GameObject that contains the following objects:

- Flashlight: Provides directional lighting activated by the player.

- Armslight: A light source specifically for illuminating the player's arms, ensuring they are visible.

- Throwpoint: The point from which bottles (or molotovs) are thrown.

- Zombie Light: A light source aimed at illuminating zombies.

- Hide Cube: A mechanism allowing the player to hide from zombies.

- Breathing: Contains a breathing script for when the player is fatigued and a audio source.

These components and objects work together to create a fully functional first-person player character.

Animator

The player animator is the following:

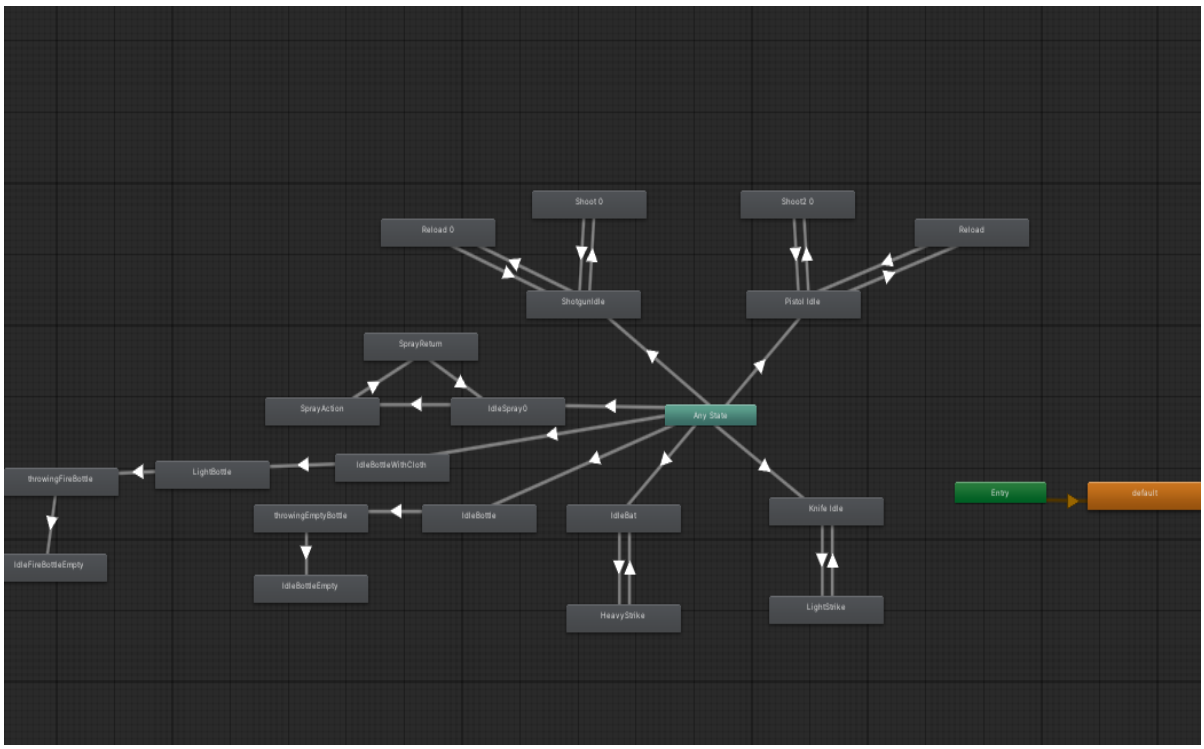
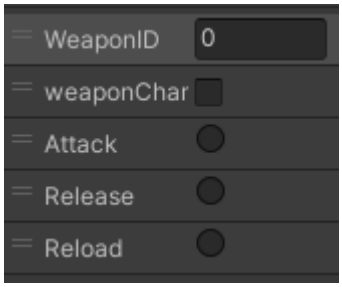


Figura 23-Player animator

It contains the animation for all the guns and player behaviour.



This are the triggers on the animator as well as the enum for the weaponID.

Zombies

There are a bunch of types of zombies in the game. Normal zombie , male and female , thin zombie , cop zombie (male and female) and military zombie.

Figura 24- Female zombie

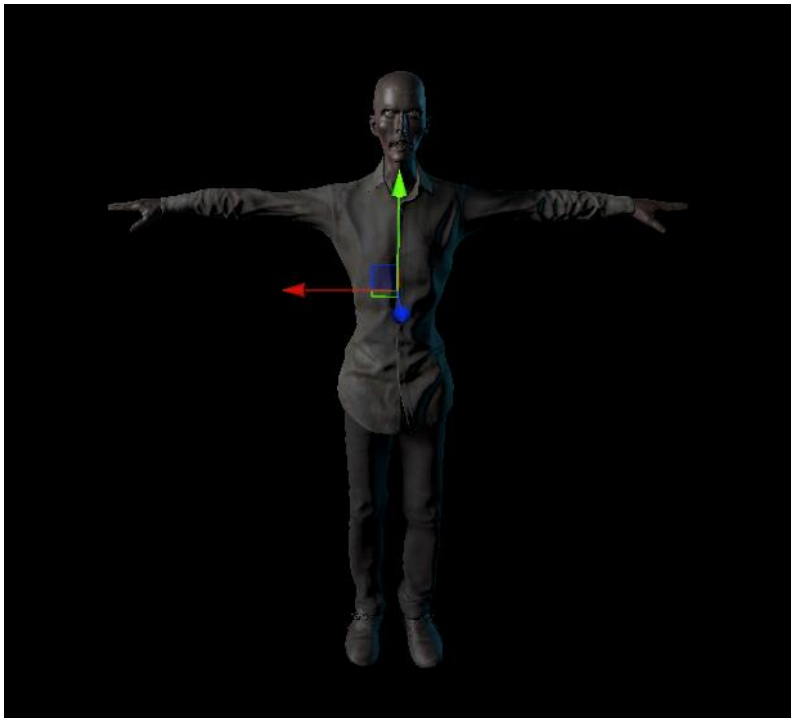


Figura 25-Thin zombie

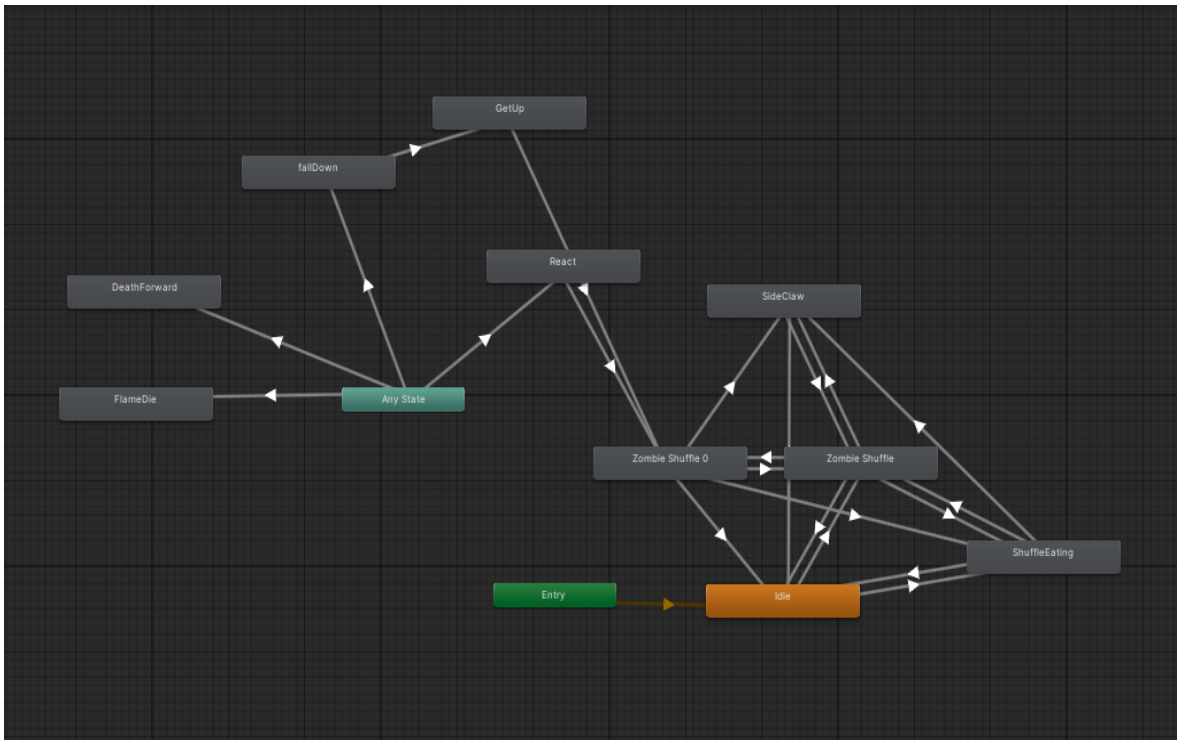
Figura 26-Cop zombie

Figura 27- Military zombie

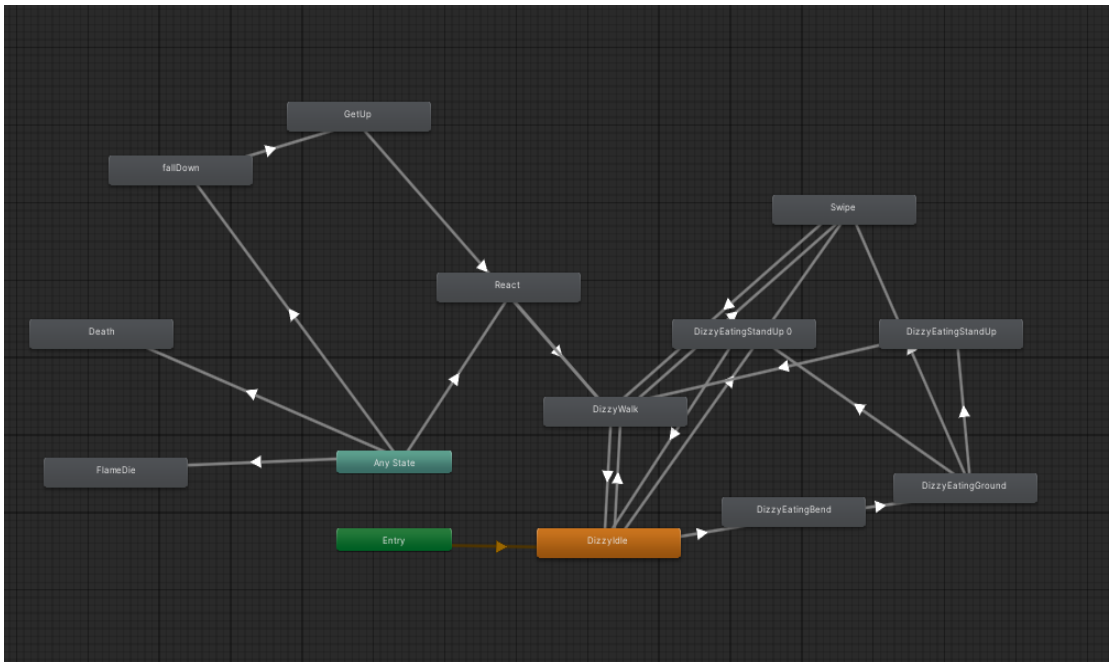
All zombies are controlled through a script. They spawn in designated spawn zones placed throughout the map and appear in these zones via a script. The zombie population and types are stored in an object called "Zombie Load." The number of zombies never exceeds 140 to prevent overloading the computer. Zombies inside houses only appear when the player enters the house, triggered by a collider and a tag comparison (tag.CompareTag).

All zombies have parameters that define their behaviors. Some parameters control their behavior upon spawning, while others determine how they react to the player (dead, `firedie, isDead (bool), axereact) and their actions when attacking the player (attacking). These zombies also have states that determine their animations. They spawn with specific states, or these states can be triggered when they are attacked by the player. When a player enters a zone the zombies become alert.

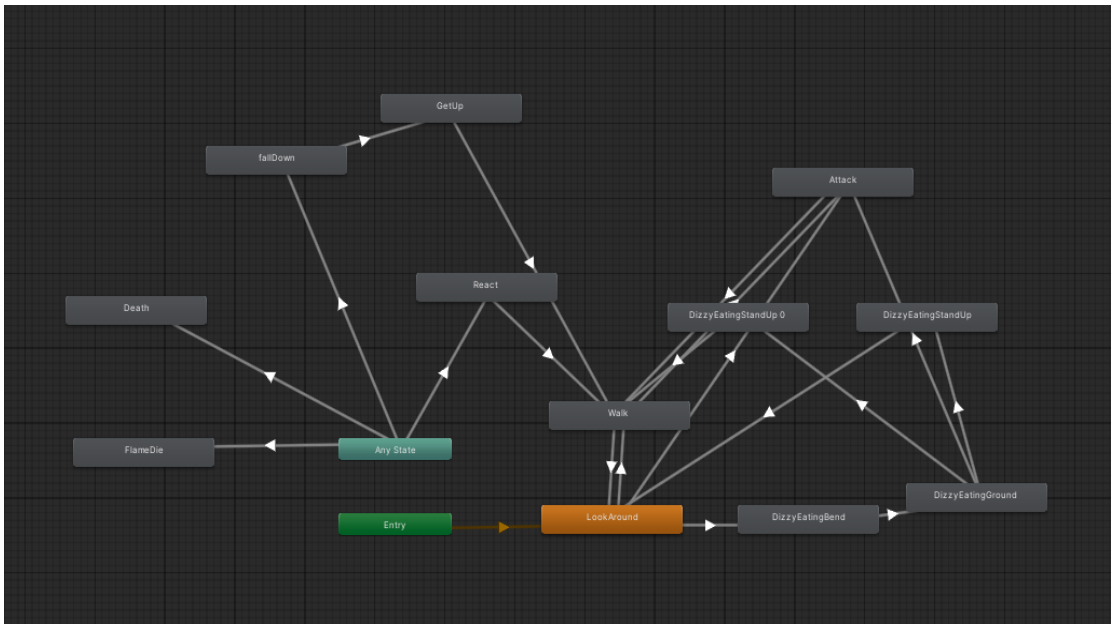
Shuffe State

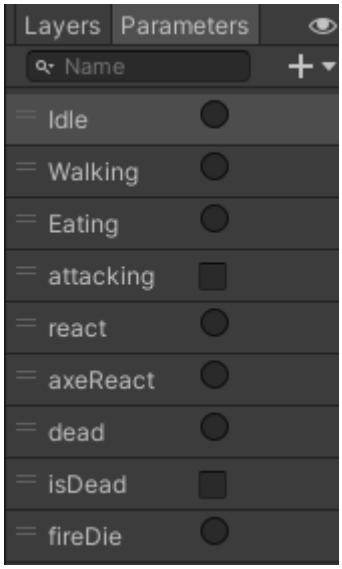


Dizzy State



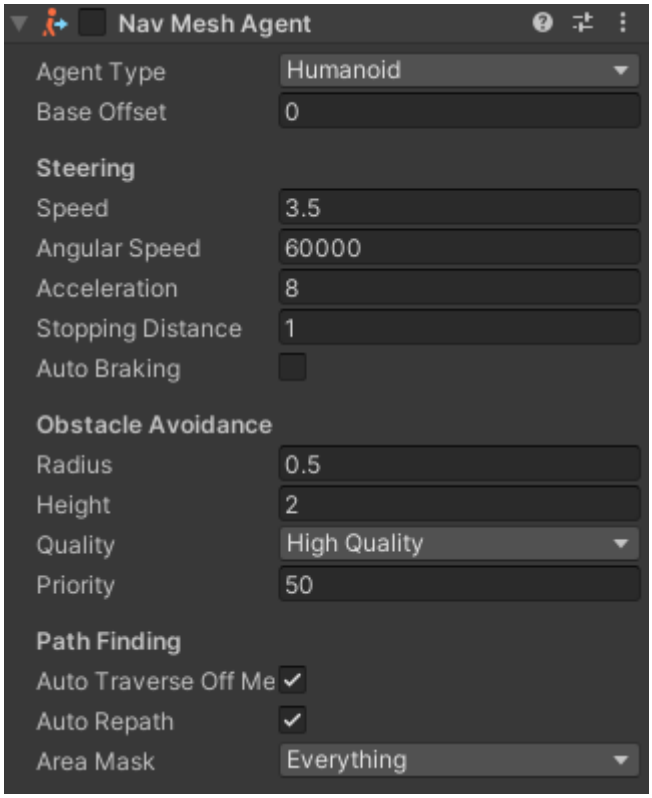
Alert





This parameters are applied to all of the states.

All zombies have a NavMesh Agent of the humanoid type. This component enables realistic and intelligent movement by allowing zombies to calculate paths, move along these paths, and avoid obstacles. It ensures lifelike navigation, optimized for characters that walk on two legs, enhancing behaviors such as chasing the player, patrolling, and retreating.



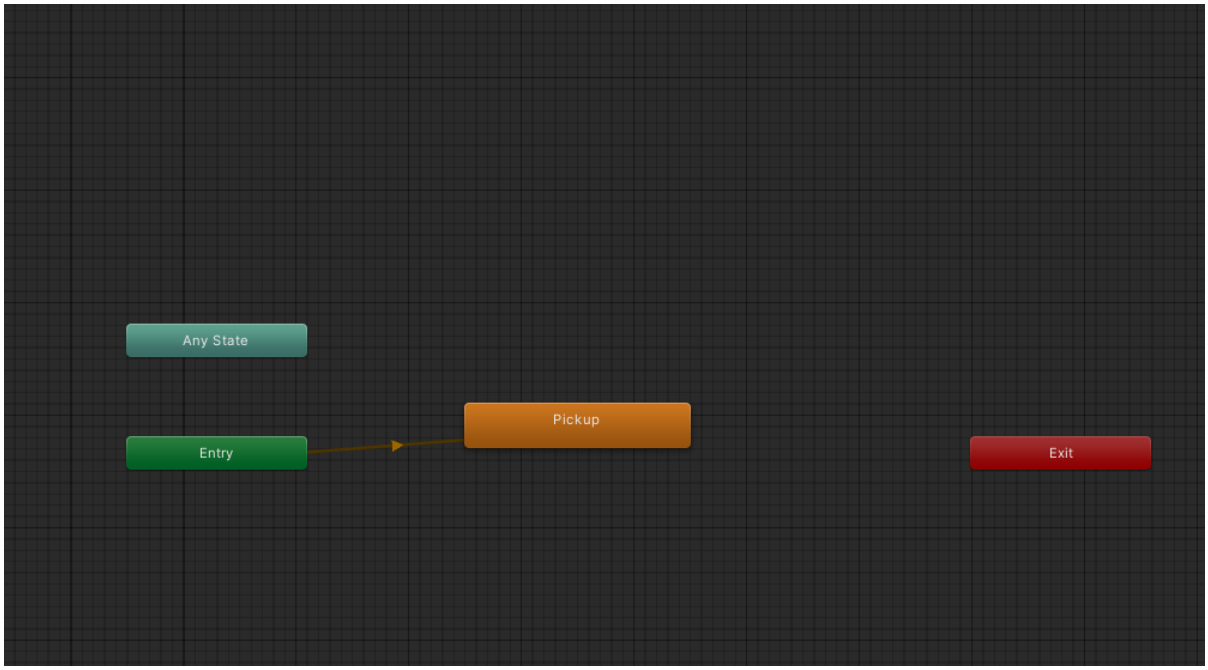
They also have an audio source, each type of zombie has a diferent sound effect.

PICKUPS

Figura 29-Pickups

For the pickups, there is a script that controls them and a script that indicates the type of item. Based on the item type, it is correctly added to the inventory. When the player points at the item, a panel is shown with the message "{ITEM NAME} E TO PICK UP." When placing the items, the item type script is added, and the item type is selected accordingly.

The pickup items have an animation applied to them that makes them glow, highlighting them to make them stand out.



The pickups can be located on the floor or inside cabinets and lockers. The doors of these cabinets and lockers are closed, and the player will need to open the doors to access and pick up the items.

Figura 30- Item in the cabinet

Figura 31-Item in a locker

GUNS

Each weapon has its own model, with adjustments made to position and rotation to fit the player model. Each weapon also has its own animation, whether for attacking or firing, which is integrated into the player's animations as previously mentioned. All the weapons used were obtained from the website Sketchfab.

Figura 32-Cleaver

Figura 33-Bat

Figura 34-Axe

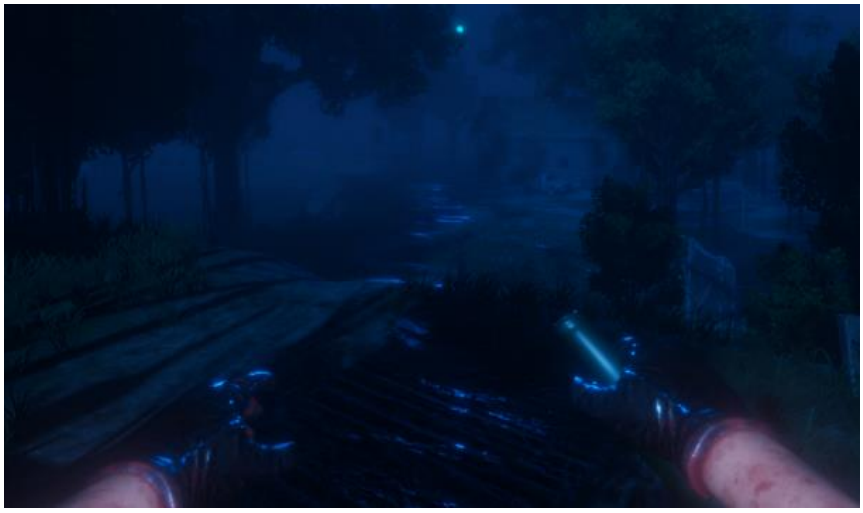


Figura 35- Bottle

Pistol

The pistol , shotgun and spray can have overlays. This overlays were made in Photoshop.

Both pistol and shotgun can be reloaded using the right click , this triggers the reload animation.

Figura 36- Pistol

Shotgun

Both the shotgun and pistol have a muzzle flash effect when fired. This effect is created using Unity's particle system, which renders the muzzle flash and a light effect from imported files. The size of the muzzle flash for the pistol is smaller than that of the shotgun.

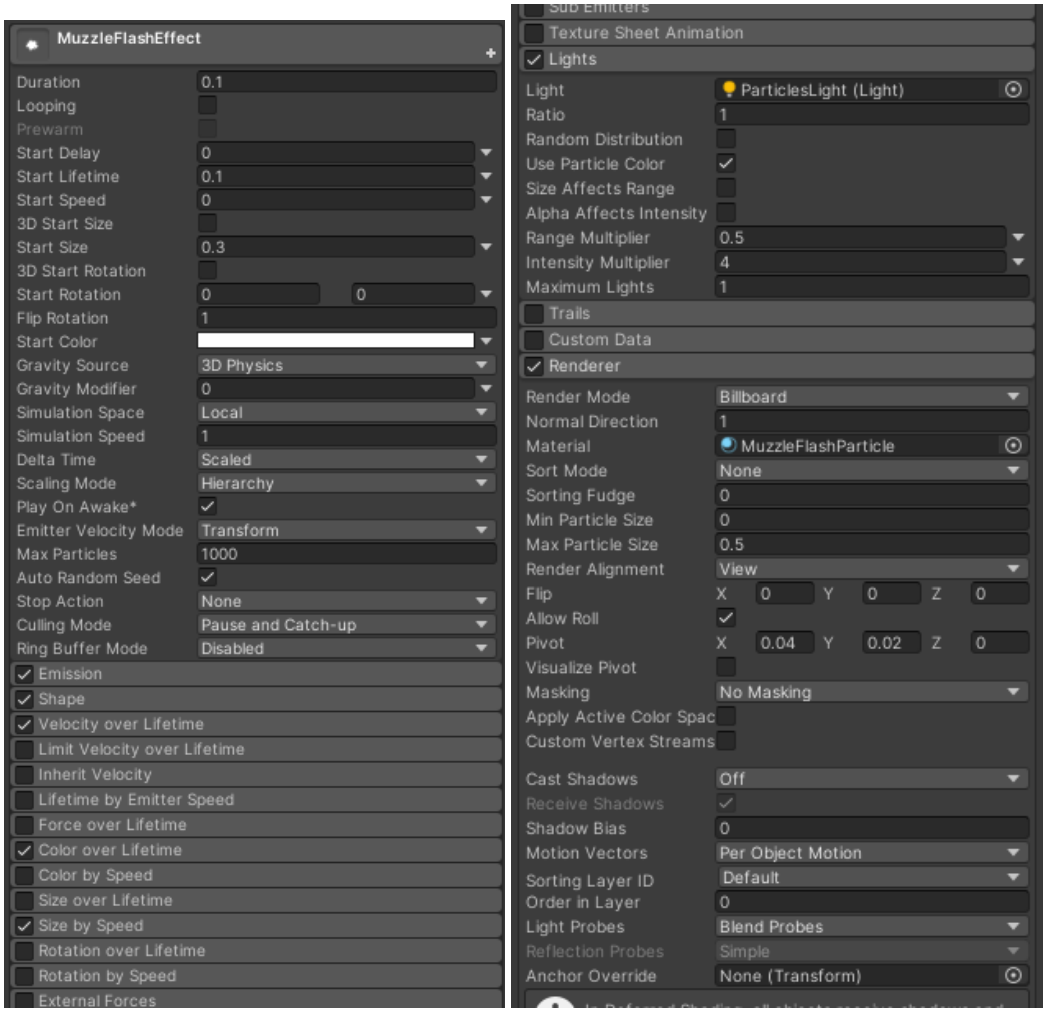


Figura 37-Shotgun

Molotov

This item is made combining a bottle with a lighter and cloth.

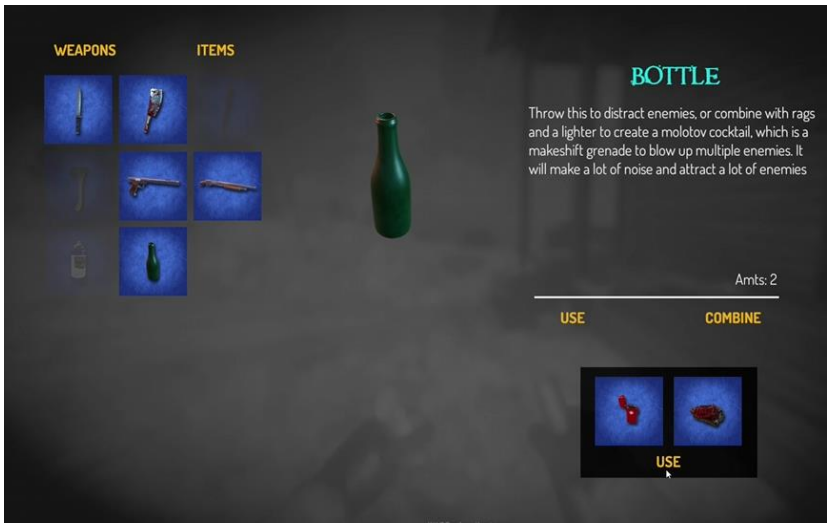
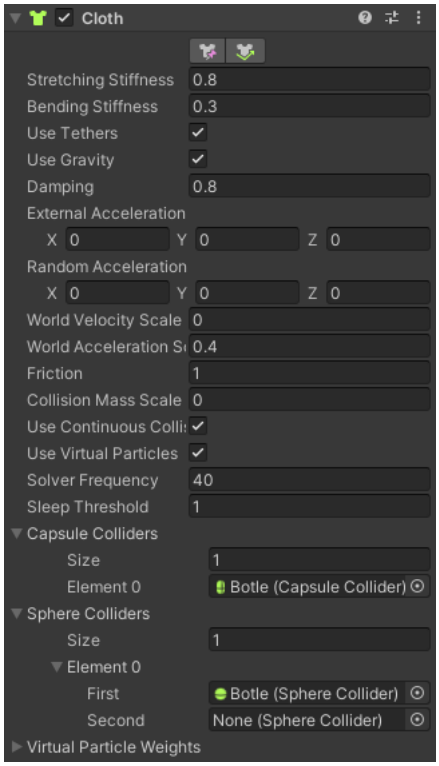


Figura 38- Molotov Combining

The cloth in the bottle has the cloth effect using the cloth unity parameter attached to the cloth object. I connected some points to the bottle so the cloth doesn't get off.



The player can use the Right Click and through the line renderer , a line appears that calculates the distance the Molotov Is thrown.

Figura 39-Molotov explosion

It’s thrown using a script and It explodes using another one.

The explosion is made using the particle system , the “BottleSmash” script controls how a bottle behaves when it collides with another object or zombie. It sets up audio playback

and physics handling, plays a collision sound once, stops the bottle's movement, saves its position, triggers an explosion effect, and removes it from the scene.

The lighter also has It's script , enabling or disabling it , playing the animation. And a particle system that indicates It's ignition.

"Flamethrower"

The Spray Can can be combined with a lighter using the combination menu.

Figura 40- Spray can combining

Using a script that manages animation and triggers the attack action, we control the spray can's behavior. Its durability is handled by another script, which depletes the can visually as it's used, reflecting its duration. When triggering the attack with the spray can, corresponding effects like flames, fire embers, and light are instantiated. These effects are all managed by Unity's Particle System.

Figura 41 -Spray Can attack

DOOR SYSTEM

When pointing at a door, the message "PRESS E TO OPEN DOOR" appears, facilitated by a script that checks the door type and whether it requires a key to open. If no key is needed, the script transforms the door and plays an opening animation. If the player lacks the key or if the key requires power, a message informs them accordingly. The doors have two parameters that condition their animations: whether the door is closed or open. For the double doors each one of the doors has a animation , whether it's the right or the left door.

This door system applies to cabinets , cabin doors and house doors.

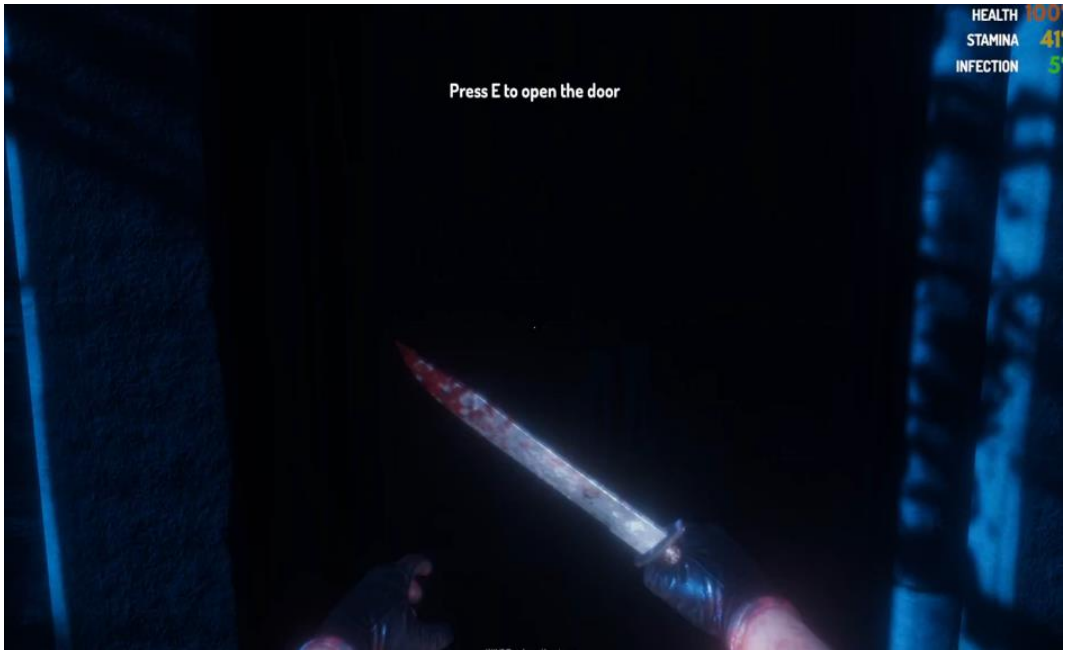


Figura 42- Door Message

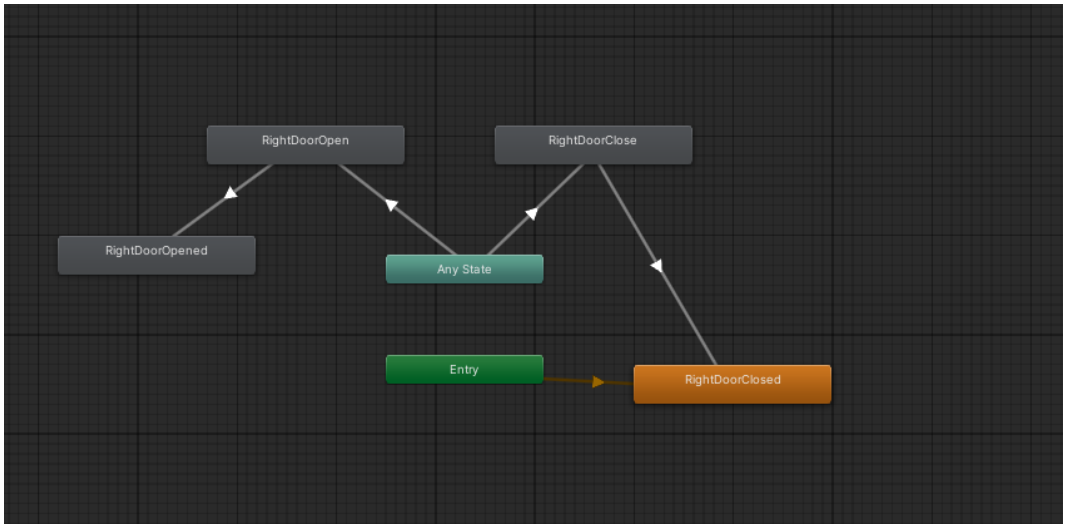


Figura 43-Door animator

GENERATOR

This game features a generator that powers an electric door. To activate the generator, a jerry can is required. Accessing the inventory and using the jerry can allows the generator to start, enabling the use of the door. When the generator It's turned on a sound effect starts to play.

Figura 44- Generator

VACCINES

The purpose of the game It's to pickup a vaccine , using a script , when the player points to the vaccine It appears the following message , the objective It's to pickup the vaccine and make a run to the lighthouse.

SPAWN POINTS & SOUNDS

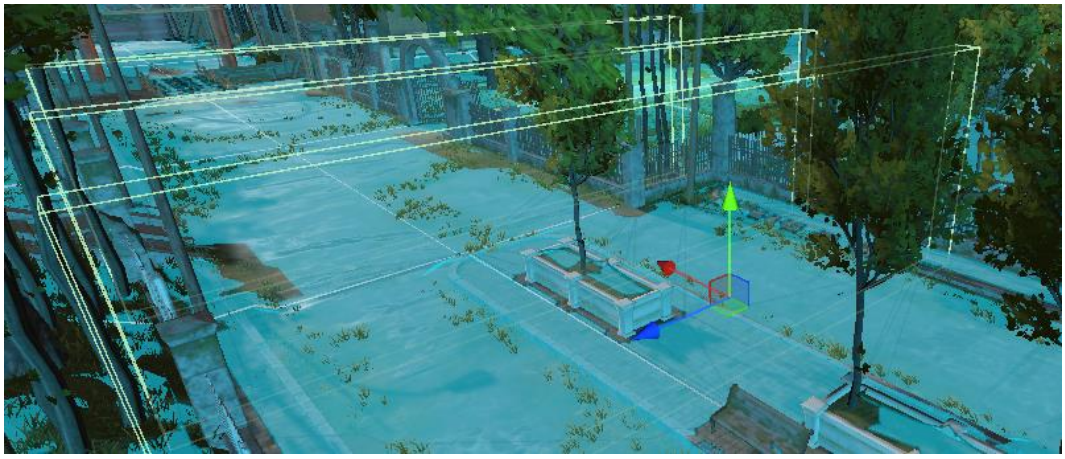


Figura 45-Spawn Points

Using spawn points, colliders, and a script, zombies are spawned in the game. This script includes a timer to respawn zombies periodically. Inside houses, there are also spawn points. When a player enters a house (by a trigger), zombies spawn at that location as well.

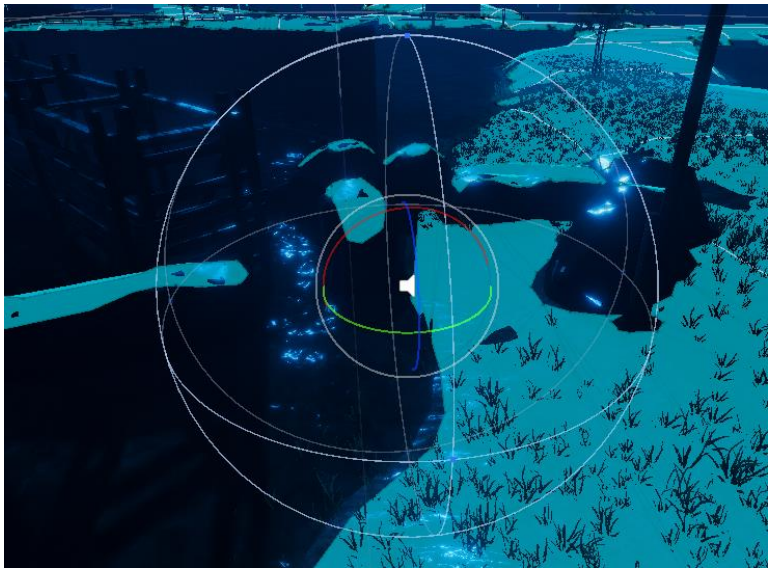


Figura 46- Audio zone

Music zones are elements in the game that play sound effects or music. These zones have a defined range and play their content in a loop during the awake phase. When the player enters the range of a music zone, they can hear the music through their own audio listener.

HUD

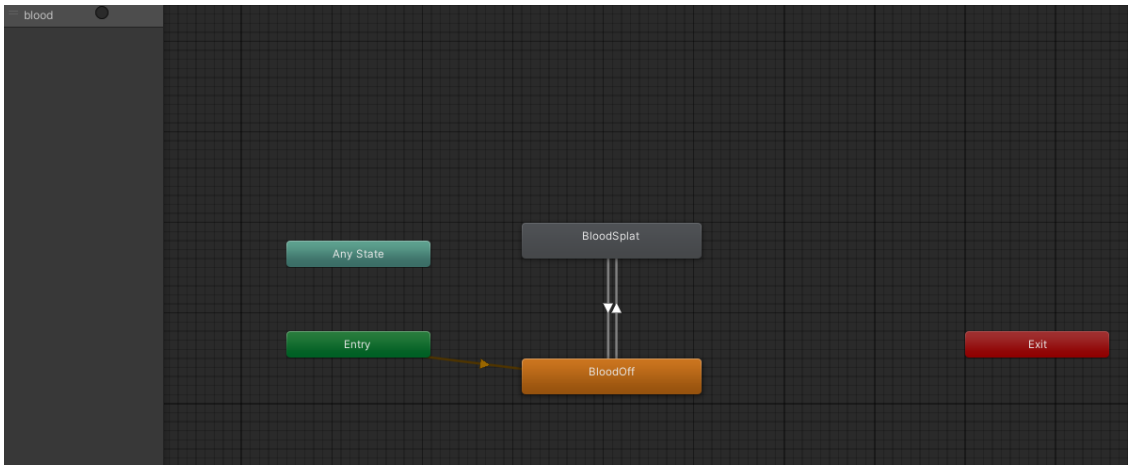
Figura 47-Life HUD

The HUD displays health, stamina, and infection levels, which dynamically change during gameplay. Infection increases slowly at 0.1f * Time.deltaTime when below 50% and faster at 0.4f * Time.deltaTime when above. Pills can reduce infection by 25%. Stamina decreases significantly when running (10 * Time.deltaTime), attacking, or crouching. Health decreases by 3 points per enemy attack, increasing infection by 3. Apples, water, or health packs can restore health and stamina.

When the player is attacked the following blood effect appears (this was made by me in photoshop):

Figura 48-Blood effect

The following effect is triggered when attacked by a zombie, and it has its own animation.



The HUD also contains a dot that Is used as a crosshair.

Figura 49- Crosshair

In the HUD the player can also show or disable the FPS counter using the “V” and “B” keys . This displays the FPS’s in the left top corner.

If the user use the ESC key It transtions to the main menu screen.

AUTO LOD & SHADERS

AutoLOD

As the project occupied too much space due to high-poly assets, I used software to reduce the triangle count of each item. Using auto LOD helped achieve this purpose.

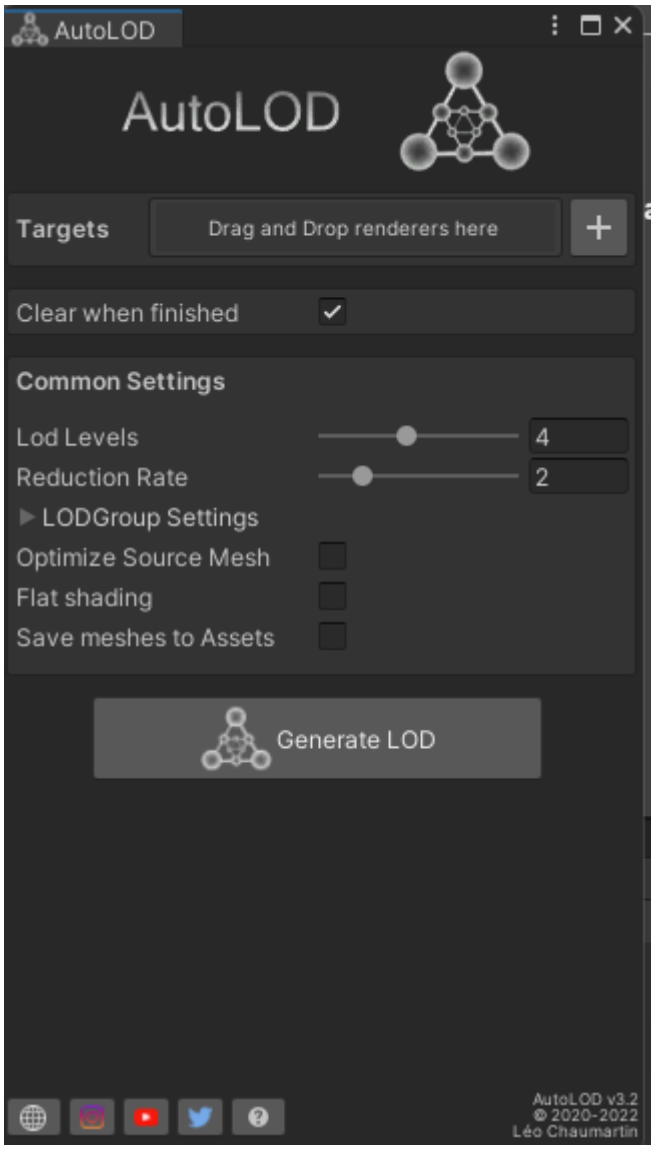


Figura 50-Auto LOD

Shaders

I used shaders from Ciconia Studios to apply a double-sided shader to the bottle's fabric. Before applying the shader, only the top side was illuminated, while the underside remained too dark.

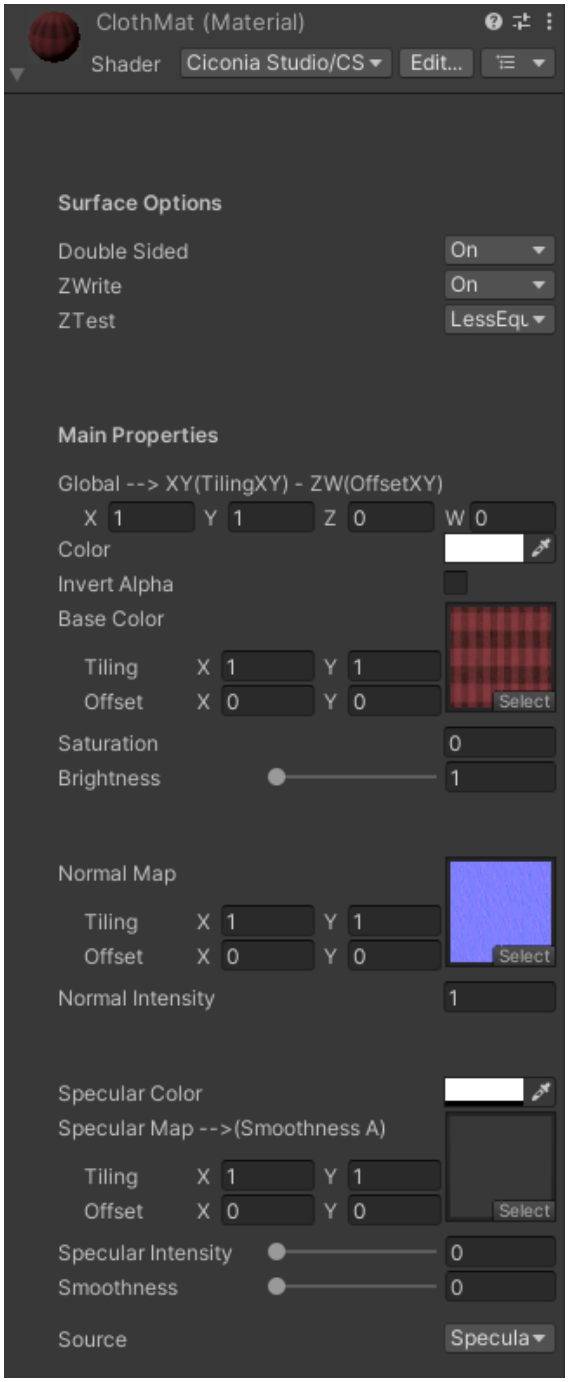


Figura 51-Cloth Shaders

MAIN MENU AND LOADING SCREEN

This is the Main Menu, which is quite simple. Clicking "Play" transitions to a loading screen scene, while clicking "Exit" exits the game. All of this is controlled through a loadingScreen script.

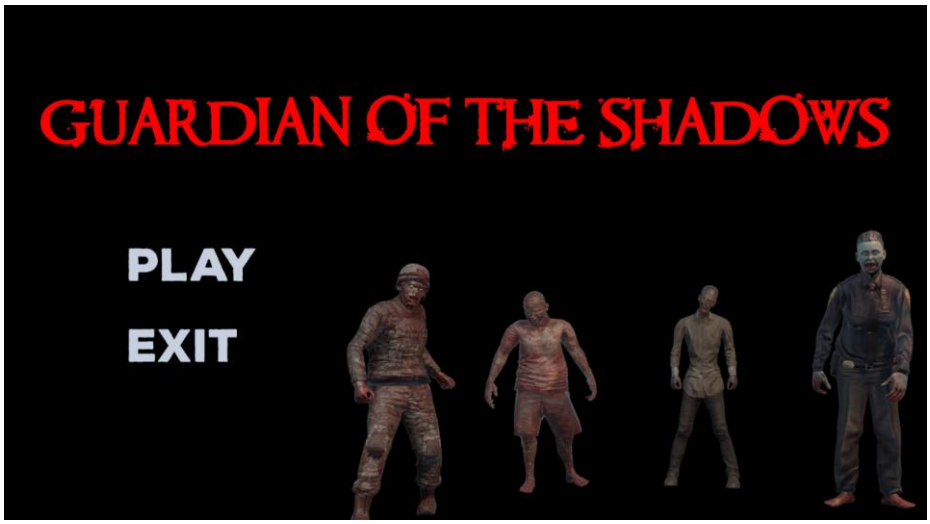


Figura 10-Main Menu

This is the loading screen that displays controls and explains the player's story and objectives. It includes a loading wheel to keep the user engaged while loading the main game level. Once the loading is complete, it transitions to the next screen.

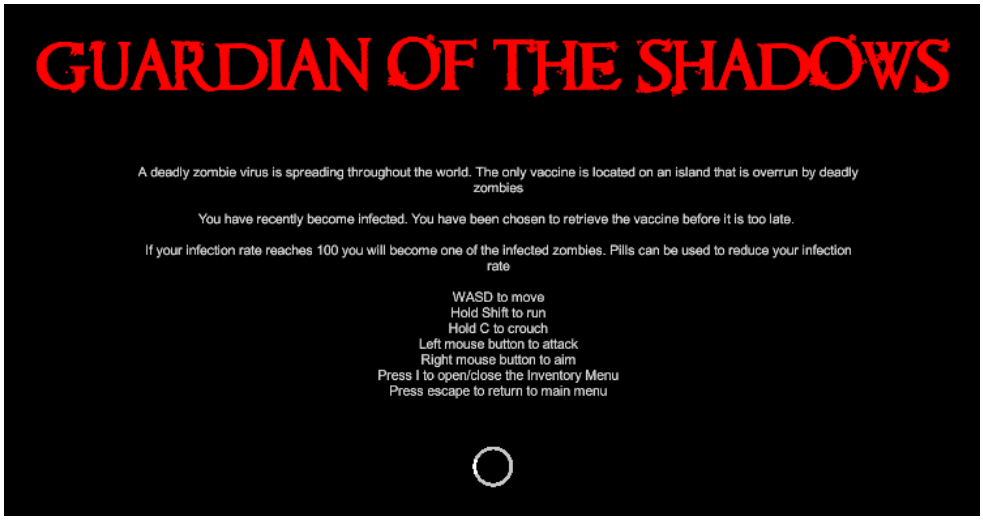


Figura 52-Loading Screen

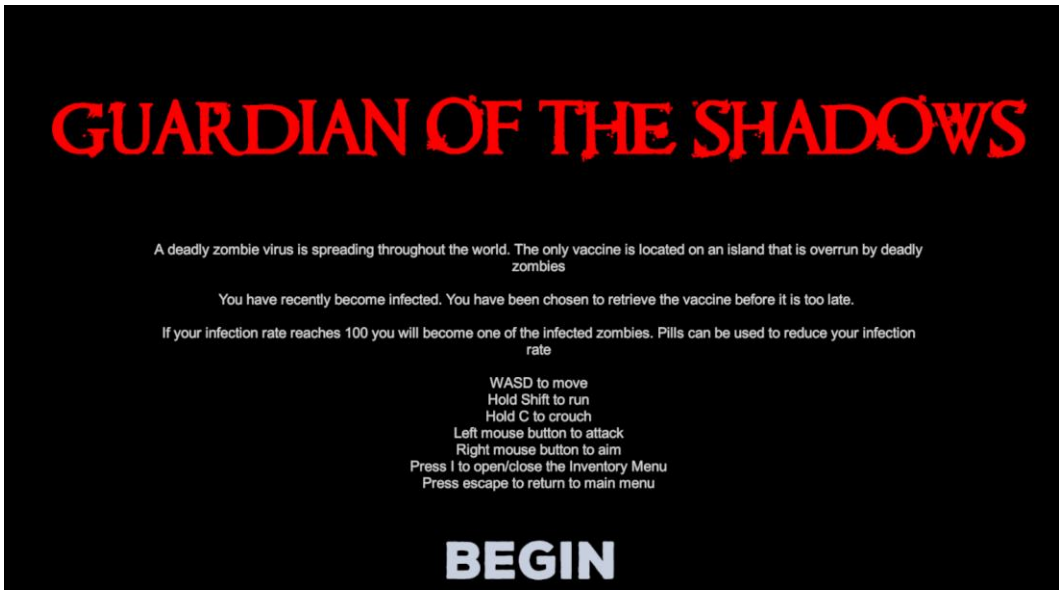
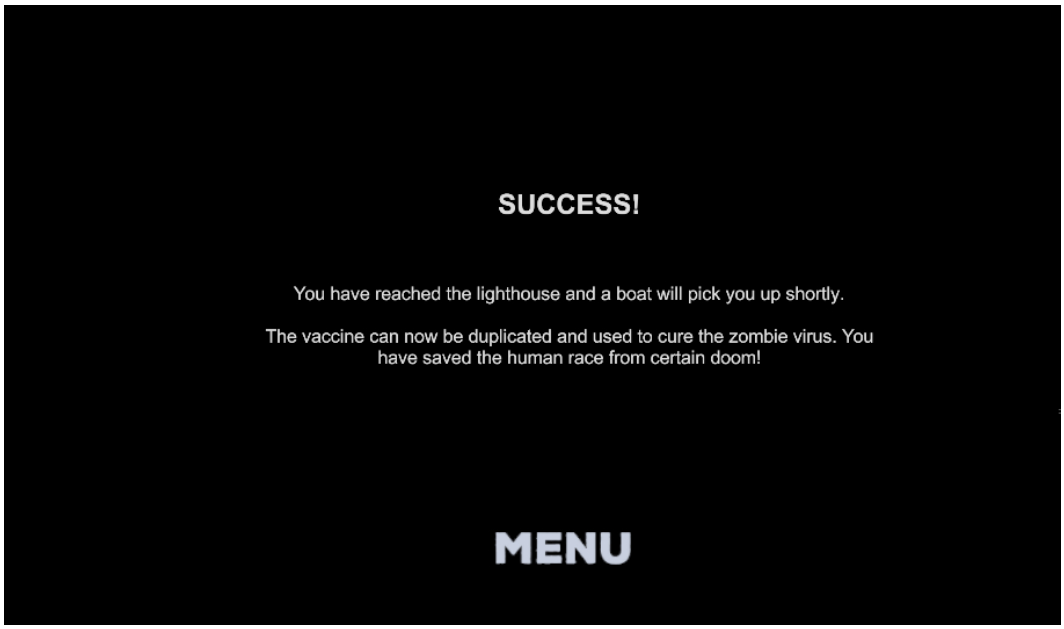


Figura 53- Main game loading screen

When the game is loaded It appears this screen , that its already in the Main game scene, when the player hits begin the game starts to run immediately and smoothly.

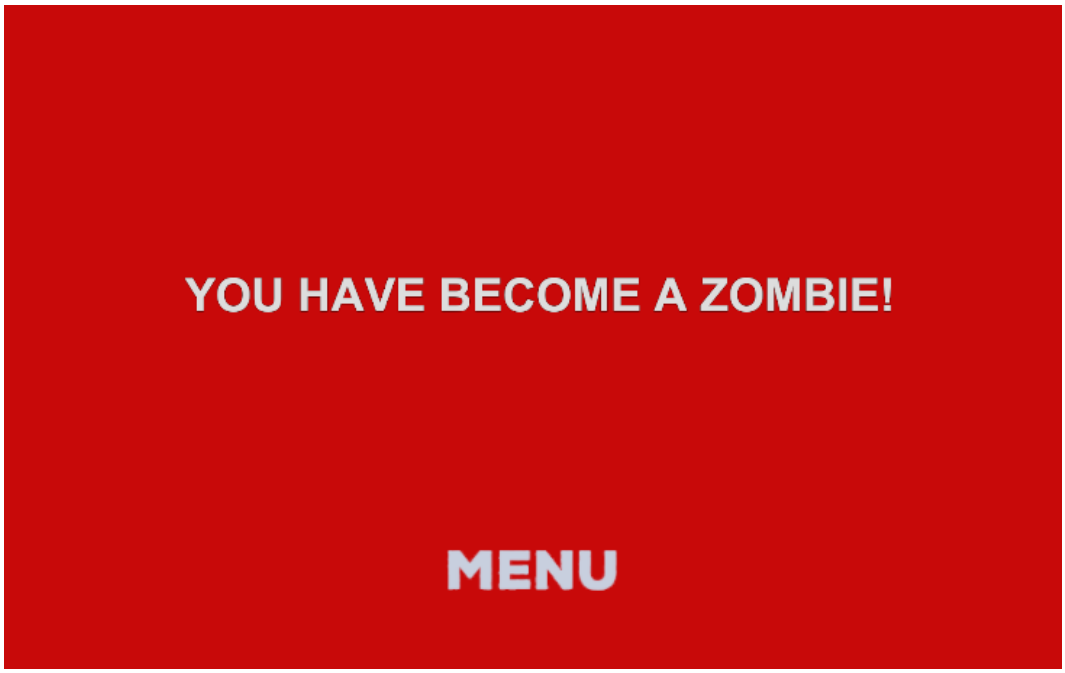
WinScreen



When the player enters the win zone this message is displayed.

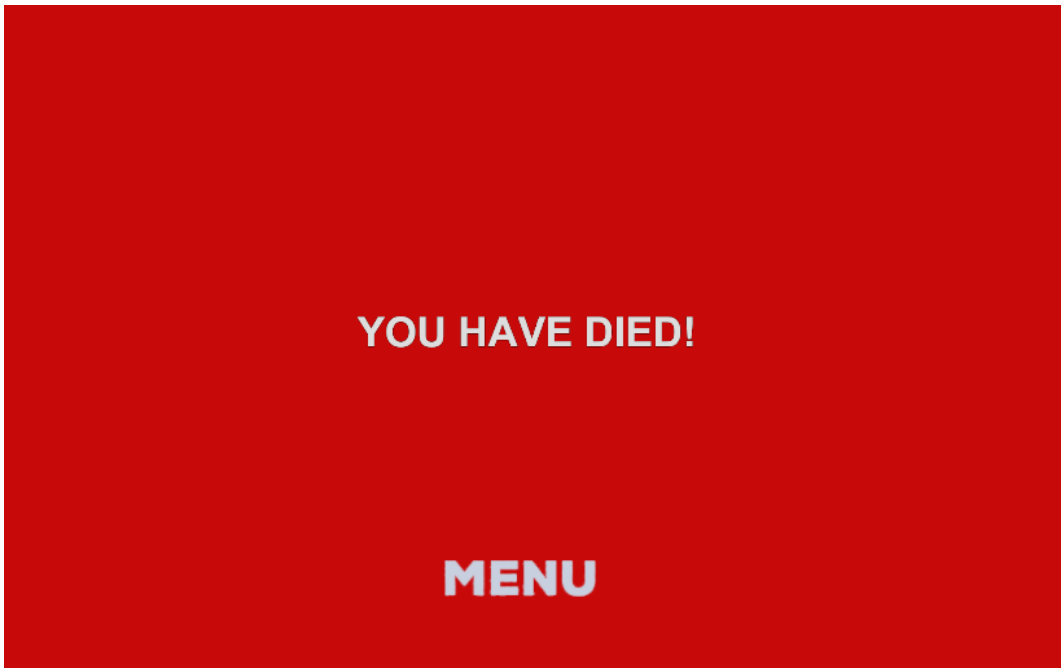
Zombie Screen

When the infection levels hit 100% , the player becomes a zombie.



Dead Screen

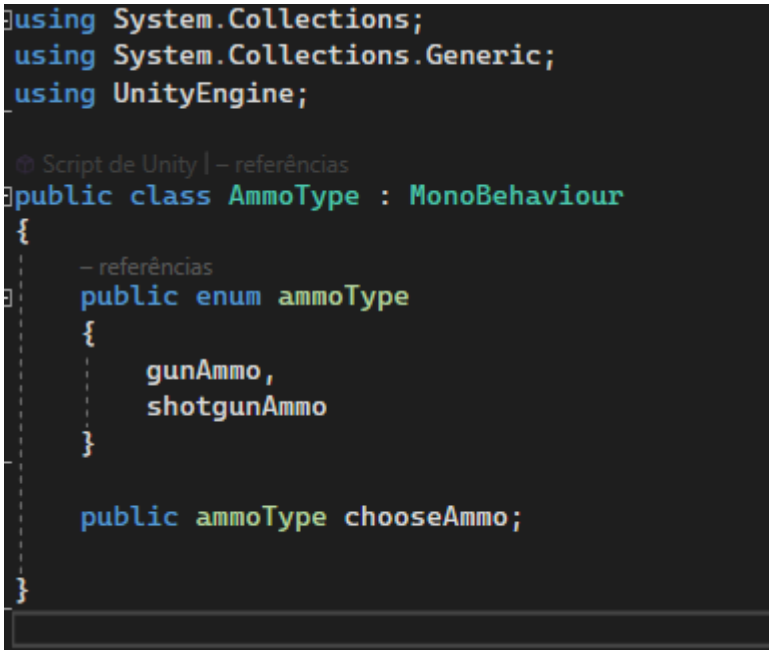
When health level hit 0% , the player dies and is shown this screen.



SCRIPTS

AmmoType

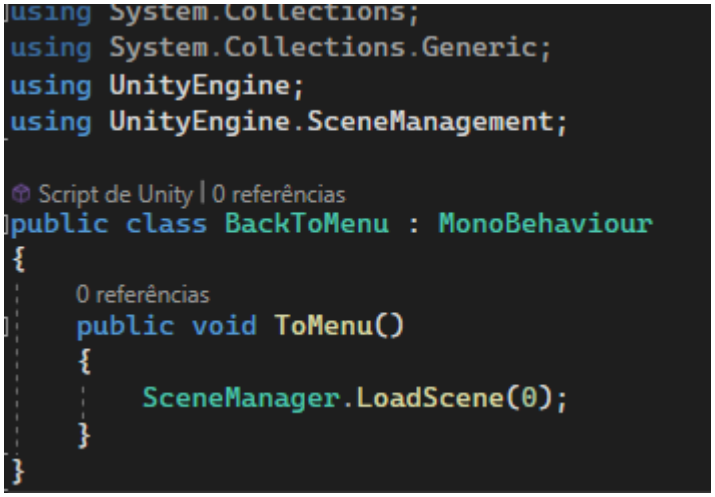
This script is used to categorize the type of ammo.



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

Script de Unity | – referências
public class AmmoType : MonoBehaviour
{
    – referências
    public enum ammoType
    {
        gunAmmo,
        shotgunAmmo
    }

    public ammoType chooseAmmo;

}
```

BackToMenu

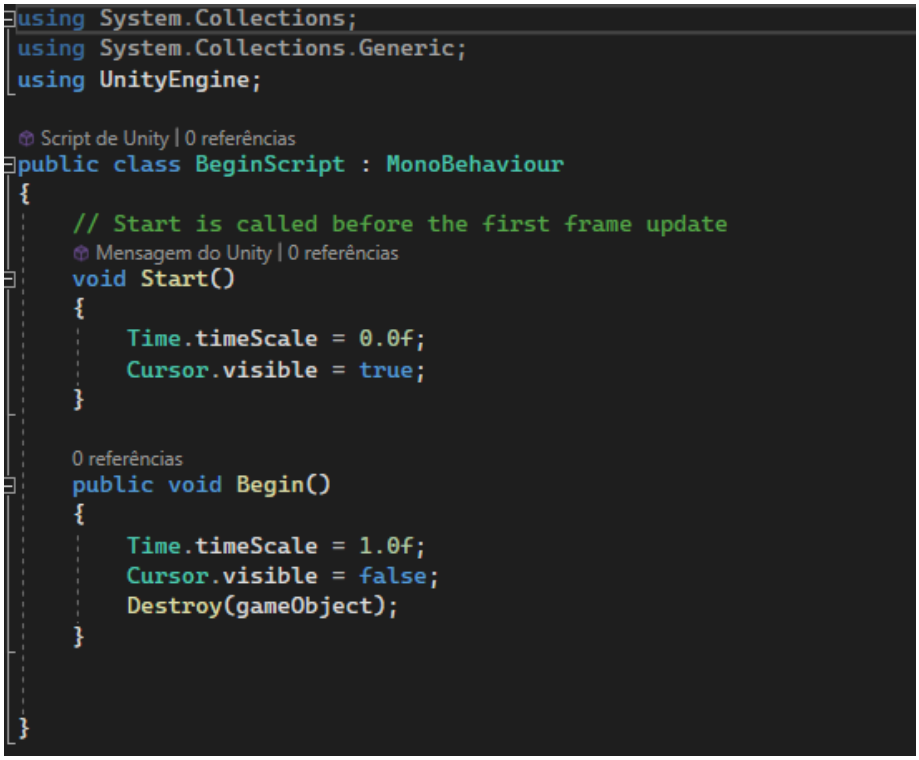
This script is used when using the ESC key , fordwards the player to the menu.



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

Script de Unity | 0 referências
public class BackToMenu : MonoBehaviour
{
    0 referências
    public void ToMenu()
    {
        SceneManager.LoadScene(0);
    }
}
```

BeginScript

This script is used when the player clicks in begin in the loading screen.



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

Script de Unity | 0 referências
public class BeginScript : MonoBehaviour
{
    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        Time.timeScale = 0.0f;
        Cursor.visible = true;
    }

    0 referências
    public void Begin()
    {
        Time.timeScale = 1.0f;
        Cursor.visible = false;
        Destroy(gameObject);
    }


}
```

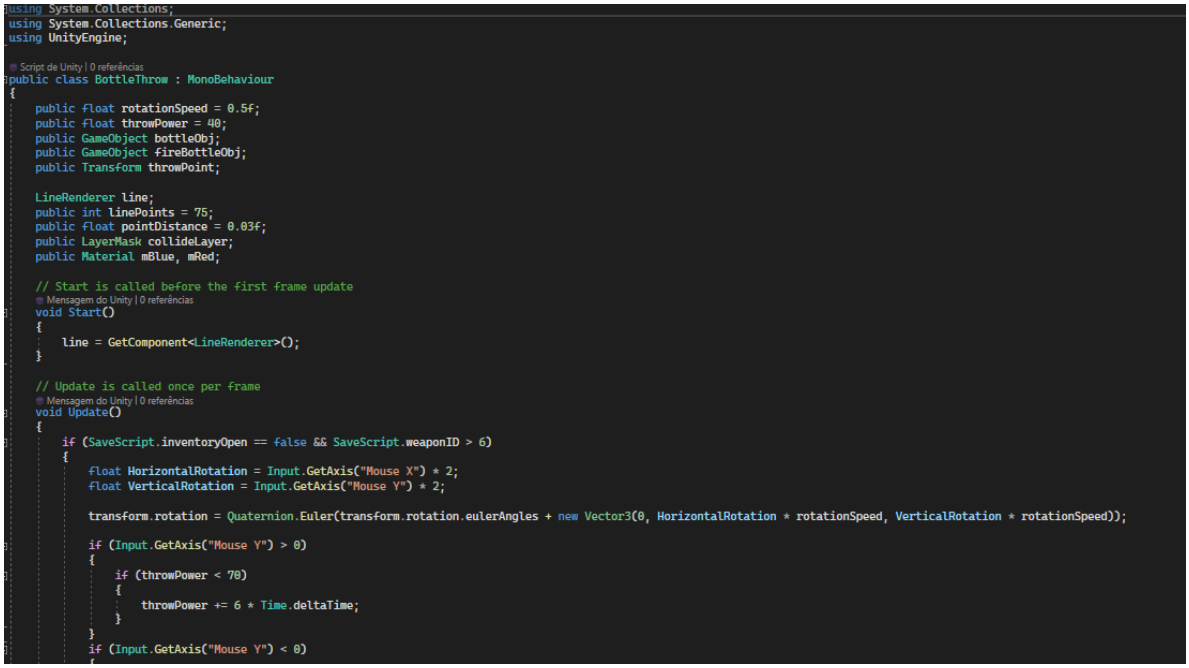
BottleSmash

This script controls the collision of the bottle with the ground , zombie or object , when collides It displays the effect and sound and destroys the bottle.

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

Script de Unity | 0 referências
public class BottleSmash : MonoBehaviour
{
    private AudioSource audioPlayer;
    private Rigidbody rb;
    private bool playSound = false;
    public GameObject bottleParent;
    public float destroyTime = 3;
    public bool flames = false;
    public GameObject explosion;


    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        audioPlayer = GetComponent<AudioSource>();
        rb = GetComponent<Rigidbody>();
        Destroy(bottleParent, 20);
    }

    Mensagem do Unity | 0 referências
    private void OnCollisionEnter(Collision collision)
    {
        if(playSound == false)
        {
            playSound = true;
            audioPlayer.Play();
            rb.isKinematic = true;
            SaveScript.bottlePos = this.transform.position;
            Destroy(bottleParent, destroyTime);

        }
        if(flames == true)
        {
            Instantiate(explosion, this.transform.position, this.transform.rotation);
        }
    }
}
```

BottleThrow

This script controls the throwing of bottles in the game. It adjusts aim and throwing power based on mouse input. When the right mouse button is held, it predicts and displays the bottle's trajectory using a LineRenderer, checking for collisions to limit its path. When conditions are met (such as WeaponManager.emptyBottleThrow or WeaponManager.fireBottleThrow being true), it instantiates bottles at the throw point with specified velocities. It manages inventory deductions and restricts throwing when the inventory is open or when certain weapons are equipped (SaveScript.inventoryOpen or SaveScript.weaponID > 6).



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

Script de Unity | 0 referências
public class BottleThrow : MonoBehaviour
{
    public float rotationSpeed = 0.5f;
    public float throwPower = 40;
    public GameObject bottleObj;
    public GameObject fireBottleObj;
    public Transform throwPoint;

    LineRenderer line;
    public int linePoints = 75;
    public float pointDistance = 0.03f;
    public LayerMask collideLayer;
    public Material mBlue, mRed;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        line = GetComponent<LineRenderer>();
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if (SaveScript.inventoryOpen == false && SaveScript.weaponID > 6)
        {
            float HorizontalRotation = Input.GetAxis("Mouse X") * 2;
            float VerticalRotation = Input.GetAxis("Mouse Y") * 2;

            transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles + new Vector3(0, HorizontalRotation * rotationSpeed, VerticalRotation * rotationSpeed));

            if (Input.GetAxis("Mouse Y") > 0)
            {
                if (throwPower < 70)
                {
                    throwPower += 6 * Time.deltaTime;
                }
            }
            if (Input.GetAxis("Mouse Y") < 0)
            {
```

```
        if (throwPower > 20)
        {
            throwPower -= 12 * Time.deltaTime;
        }
    }

    line.positionCount = linePoints;
    List<Vector3> points = new List<Vector3>();
    Vector3 startPos = throwPoint.position;
    Vector3 startVelocity = throwPoint.forward * throwPower;

    if (Input.GetMouseButton(1))
    {
        if(SaveScript.weaponID == 7)
        {
            line.material = mBlue;
        }
        if (SaveScript.weaponID == 8)
        {
            line.material = mRed;
        }
        for (float i = 0; i < linePoints; i += pointDistance)
        {
            Vector3 newPoint = startPos + i * startVelocity;
            newPoint.y = startPos.y + startVelocity.y + i + Physics.gravity.y / 2f * i * i;
            points.Add(newPoint);

            if (Physics.OverlapSphere(newPoint, 0.01f, collideLayer).Length > 0)
            {
                line.positionCount = points.Count;
                break;
            }
        }

        line.SetPositions(points.ToArray());
    }

    if (Input.GetMouseButtonUp(1))
    {
        line.positionCount = 0;
    }


    if (WeaponManager.emptyBottleThrow == true)
    {
        WeaponManager.emptyBottleThrow = false;
```

```
    {
        WeaponManager.emptyBottleThrow = false;
        GameObject createBottle = Instantiate(bottleObj, throwPoint.position, throwPoint.rotation);
        createBottle.GetComponentInChildren<Rigidbody>().velocity = throwPoint.transform.forward * throwPower;
        SaveScript.weaponAmts[7]--;
        SaveScript.change = true;
    }

    if (WeaponManager.fireBottleThrow == true)
    {
        WeaponManager.fireBottleThrow = false;
        GameObject createBottle = Instantiate(fireBottleObj, throwPoint.position, throwPoint.rotation);
        createBottle.GetComponentInChildren<Rigidbody>().velocity = throwPoint.transform.forward * throwPower;
        SaveScript.weaponAmts[7]--;
        SaveScript.itemAmts[3]--;
        SaveScript.change = true;
    }
}
```

BreathingScript

This script controls the breathing sound, when the stamina is below 20 the heavy breathing sound starts to play.

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

Script de Unity | 0 referências
public class BreathingScript : MonoBehaviour
{
    private AudioSource audioPlayer;
    private bool heavyBreathing = false;
    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        audioPlayer = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(SaveScript.stamina < 20 && heavyBreathing == false)
        {
            heavyBreathing = true;
            audioPlayer.Play();
        }
        if(SaveScript.stamina > 19)
        {
            heavyBreathing = false;
            audioPlayer.Stop();
        }
    }
}
```

Destroyer

This script is specifically used to eliminate the bloodsplat effect from the screen after 10 seconds.

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

Script de Unity | 0 referências
public class Destroyer : MonoBehaviour
{
    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        Destroy(gameObject, 10);
    }
}
```

DoorType

This script categorizes the door type ,displays the message when the player points to a door and triggers the animation. The message is public to be acessed from another script , but It's not shown in the inspector to prevent being altered ([HideInInspector])

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


public class DoorType : MonoBehaviour
{

    public enum typeOfDoor
    {
        cabinet,
        house,
        cabin
    }

    public typeOfDoor chooseDoor;
    public bool opened = false;
    public bool locked = false;
    [HideInInspector]
    public string message = "Press E to open the door";
    private Animator anim;
    public bool electricDoor = false;

    // Start is called before the first frame update

    void Start()
    {
        anim = GetComponent<Animator>();
        if(opened == true)
        {
            anim.SetTrigger("Open");
            message = "Press E to close the door";
        }
    }

}
```

Flashlight script

This script manages the flashlight battery, power and its drain as well as the drain time.

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

Script de Unity | 5 referências
public class FlashLightScript : MonoBehaviour
{
    private Image batteryChunks;
    public float batteryPower = 1.0f;
    public float drainTime = 2;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void OnEnable()
    {
        batteryChunks = GameObject.Find("FLBatteryChunks").GetComponent<Image>();
        InvokeRepeating("FLBatteryDrain", drainTime, drainTime);
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        batteryChunks.fillAmount = batteryPower;
    }

    0 referências
    private void FLBatteryDrain()
    {
        if (batteryPower > 0.0f)
            batteryPower -= 0.25f;
    }

    3 referências
    public void StopDrain()
    {
        CancelInvoke("FLBatteryDrain");
    }
}
```

Hide Script

This script manages the player being hidden , when the player collides with the hide zone It communicates with the SaveScript , passing the information that the player is hided.

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

Script de Unity | 0 referências
public class HideScript : MonoBehaviour
{
    Mensagem do Unity | 0 referências
    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("PlayerHide"))
        {
            SaveScript.isHidden = true;
        }
    }

    Mensagem do Unity | 0 referências
    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("PlayerHide"))
        {
            SaveScript.isHidden = false;
        }
    }
}
```

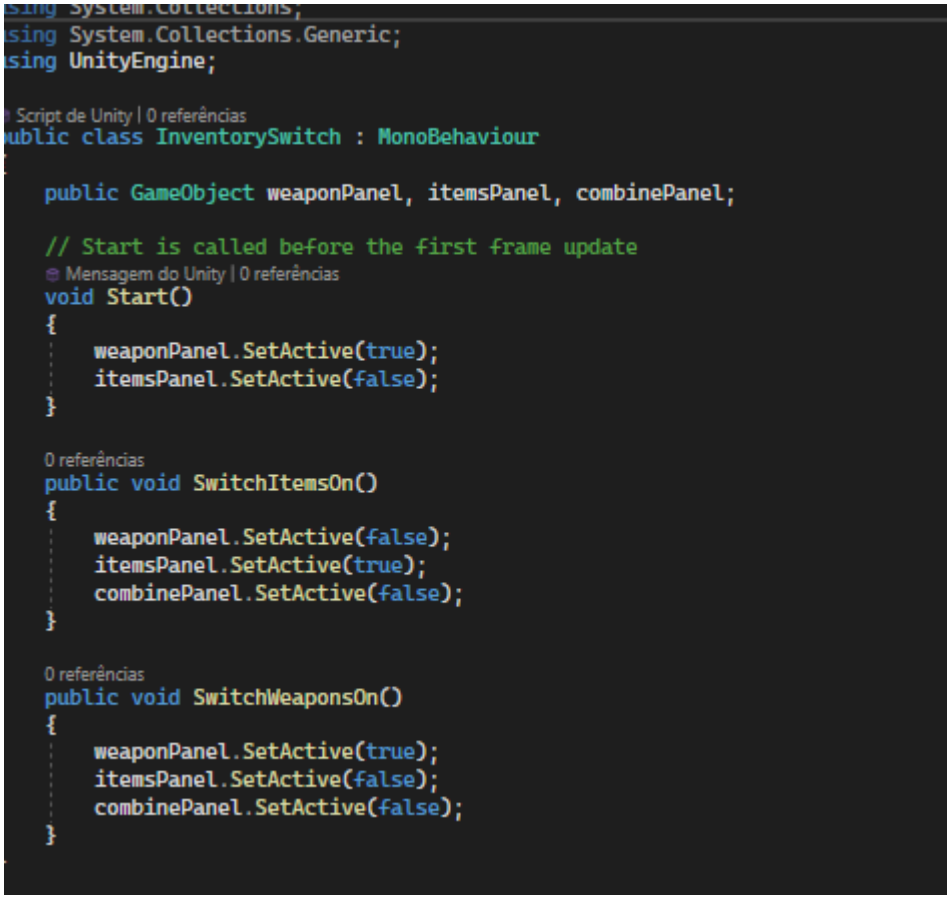
HouseTrigger

When a player (with the tag “player”) collides with the house enter zone , the zombie patrol is triggered and they start to spawn inside the house.

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

Script de Unity | 0 referências
public class HouseTrigger : MonoBehaviour
{
    public GameObject spawnZone;

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player"))
        {
            spawnZone.GetComponent<ZombieSpawnPatrol>().canSpawn = true;
        }
    }
}
```

InventorySwitch

This script manages the inventory that is show , wether is the weapons inventory or the items inventory. When the weapons inventory is turned on the items inventory is off and vice-versa.



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

Script de Unity | 0 referências
public class InventorySwitch : MonoBehaviour
{
    public GameObject weaponPanel, itemsPanel, combinePanel;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        weaponPanel.SetActive(true);
        itemsPanel.SetActive(false);
    }

    0 referências
    public void SwitchItemsOn()
    {
        weaponPanel.SetActive(false);
        itemsPanel.SetActive(true);
        combinePanel.SetActive(false);
    }

    0 referências
    public void SwitchWeaponsOn()
    {
        weaponPanel.SetActive(true);
        itemsPanel.SetActive(false);
        combinePanel.SetActive(false);
    }
```

ItemsInventory

This script controlls all that is the item inventory. At the start , the script fetches the audio source , displays the description , tittle and icon for the first item. It turns off the use button and the electric lights. On enable , when the player opens the inventory it shows the items the player possesses in a blue color , if the player does not have the items its shown a “grayish” color.

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

Script de Unity | 0 referências
public class ItemsInventory : MonoBehaviour
{
    public Sprite[] bigIcons;
    public Image bigIcon;
    public string[] titles;
    public Text title;
    public string[] descriptions;
    public Text description;
    public Button[] itemButtons;
    public GameObject useButton;
    public Text amtsText;

    private AudioSource audioPlayer;
    public AudioClip click, select;
    private int chosenItemNumber;

    private int updateHealth;
    private float updateStamina;
    private float updateInfection;

    private bool addHealth = false;
    private bool addStamina = false;
    private bool reduceInfection = false;

    public GameObject flashlightPanel, nightVisionPanel;
    private bool flashLightRefill = false;
    private bool nightVisionRefill = false;

    public GameObject electricDoorObj;
    public GameObject electricLight1, electricLight2;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        audioPlayer = GetComponent<AudioSource>();

        bigIcon.sprite = bigIcons[0];
        title.text = titles[0];
        description.text = descriptions[0];
        useButton.SetActive(false);
```

```
        electricLight1.SetActive(false);
        electricLight2.SetActive(false);
    }

    Mensagem do Unity | 0 referências
    private void OnEnable()
    {
        for (int i = 0; i < itemButtons.Length; i++)
        {
            if (SaveScript.itemsPickedUp[i] == false)
            {
                itemButtons[i].image.color = new Color(1, 1, 1, 0.06f);
                itemButtons[i].image.raycastTarget = false;
            }
            if (SaveScript.itemsPickedUp[i] == true)
            {
                itemButtons[i].image.color = new Color(1, 1, 1, 1);
                itemButtons[i].image.raycastTarget = true;
            }
        }

        if(SaveScript.itemAmts[chosenItemNumber] <= 0)
        {
            ChooseItem(0);
        }
        ChooseItem(chosenItemNumber);
    }

    3 referências
    public void ChooseItem(int itemNumber)
    {
        bigIcon.sprite = bigIcons[itemNumber];
        title.text = titles[itemNumber];
        description.text = descriptions[itemNumber];
        if (audioPlayer != null)
        {
            audioPlayer.clip = click;
            audioPlayer.Play();
        }
        chosenItemNumber = itemNumber;
        amtsText.text = "Amts: " + SaveScript.itemAmts[itemNumber];

        if(itemNumber < 4)
        {
            useButton.SetActive(false);
        }
        else
```

ChooseItem function updates UI with the selected item's image, title, description, and quantity. It plays a click sound, manages the visibility of a use button, and adjusts behaviors for items like flashlight and night vision. If the item is id 8 or 9 it can refill the flashlight or nightvision battery as that are the ids of the batteries.

```
            useButton.SetActive(true);
        }
        if(itemNumber != 8)
        {
            flashLightRefill = false;
        }
        if (itemNumber != 9)
        {
            nightVisionRefill = false;
        }
    }

    0 referências
    public void AddHealth(int healthUpdate)
    {
        updateHealth = healthUpdate;
        addHealth = true;
    }

    0 referências
    public void AddStamina(int staminaUpdate)
    {
        updateStamina = staminaUpdate;
        addStamina = true;
    }

    0 referências
    public void ReduceInfection(int infectionUpdate)
    {
        updateInfection = infectionUpdate;
        reduceInfection = true;
    }

    0 referências
    public void FillFLBattery()
    {
        flashLightRefill = true;
    }

    0 referências
    public void FillNVBattery()
    {
        nightVisionRefill = true;
    }
```

```
0 referências
public void AssignItem()
{
    SaveScript.itemID = chosenItemNumber;
    audioPlayer.clip = select;
    audioPlayer.Play();

    if (chosenItemNumber != 10 && chosenItemNumber != 11)
    {
        SaveScript.itemAmts[chosenItemNumber]--;
        ChooseItem(chosenItemNumber);
        if (SaveScript.itemAmts[chosenItemNumber] == 0)
        {
            SaveScript.itemsPickedUp[chosenItemNumber] = false;
            useButton.SetActive(false);
        }
    }


    if (addHealth == true)
    {
        addHealth = false;
        if (SaveScript.health < 100)
        {
            SaveScript.health += updateHealth;
        }
        if (SaveScript.health > 100)
        {
            SaveScript.health = 100;
        }
    }

    if (addStamina == true)
    {
        addStamina = false;
        if (SaveScript.stamina < 100)
        {
            SaveScript.stamina += updateStamina;
        }
        if (SaveScript.stamina > 100)
        {
            SaveScript.stamina = 100;
        }
    }

    if (reduceInfection == true)
```

```
if (reduceInfection == true)
{
    reduceInfection = false;
    if (SaveScript.infection > 0.0f)
    {
        SaveScript.infection -= updateInfection;
    }
    if (SaveScript.infection < 0.0f)
    {
        SaveScript.infection = 0.0f;
    }
}
if(flashLightRefill == true)
{
    flashLightRefill = false;
    flashlightPanel.GetComponent<FlashLightScript>().batteryPower = 1.0f;
}
if (nightVisionRefill == true)
{
    nightVisionRefill = false;
    nightVisionPanel.GetComponent<NightVisionScript>().batteryPower = 1.0f;
}

if(chosenItemNumber == 10)
{
    if(SaveScript.doorObject != null)
    {
        if((int)SaveScript.doorObject.GetComponent<DoorType>().chooseDoor == 1)
        {
            if(SaveScript.doorObject.GetComponent<DoorType>().locked == true)
            {
                SaveScript.doorObject.GetComponent<DoorType>().locked = false;
            }
        }
    }
}

if (chosenItemNumber == 11)
{
    if (SaveScript.doorObject != null)
    {
        if ((int)SaveScript.doorObject.GetComponent<DoorType>().chooseDoor == 2)
        {
            if (SaveScript.doorObject.GetComponent<DoorType>().locked == true)
            {
```

```
            {
                if (SaveScript.doorObject.GetComponent<DoorType>().locked == true)
                {
                    SaveScript.doorObject.GetComponent<DoorType>().locked = false;
                }
            }
        }
    }

    if(chosenItemNumber == 12)
    {
        if(SaveScript.generator != null)
        {
            SaveScript.generatorOn = true;
            SaveScript.generator.GetComponent<AudioSource>().Play();

            electricDoorObj.GetComponent<DoorType>().locked = false;
            electricLight1.SetActive(true);
            electricLight2.SetActive(true);
        }
    }

}
```

If the item is the pills ID it reduces the infection, if it's the apple or the water It increases the stamina and health , if it's the health pack it increases health . If it's the jerrycan in contact with the generator It unlocks the lab door, and turns on the lights of the lab room. If the items are the cabin or house key in contact with the specific doors It unlocks them.

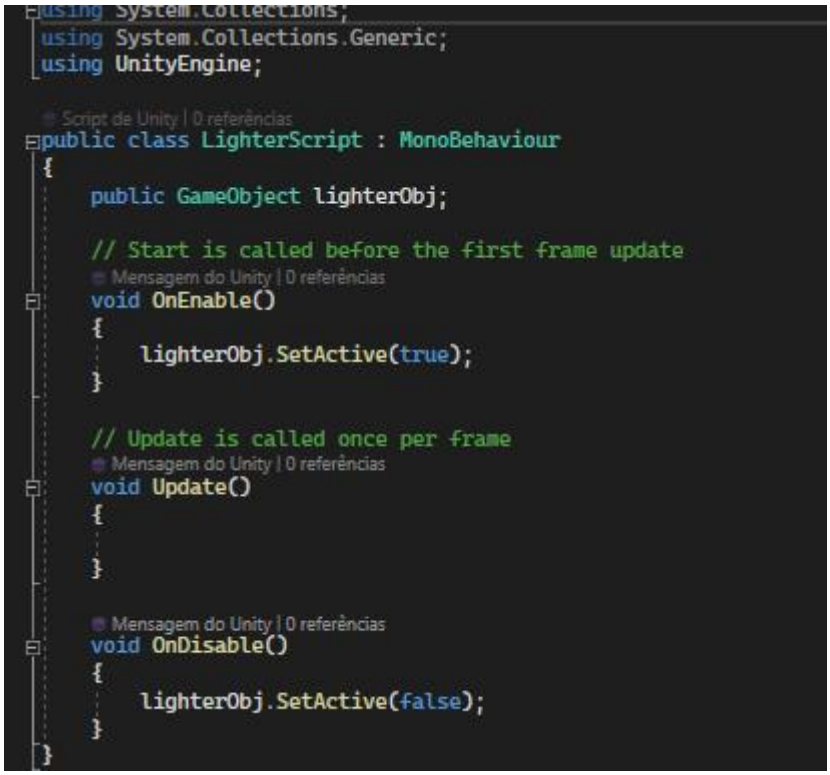
ItemsType

This script categorizes the types of items.

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

Script de Unity | 1 referência
public class ItemsType : MonoBehaviour
{
    1 referência
    public enum typeOfItem
    {
        flashlight,
        nightvision,
        lighter,
        rags,
        healthPack,
        pills,
        waterBottle,
        apple,
        flashlightBattery,
        nightvisionBattery,
        houseKey,
        cabinKey,
        jerryCan
    }

    public typeOfItem chooseItem;

}
```

LighterScript

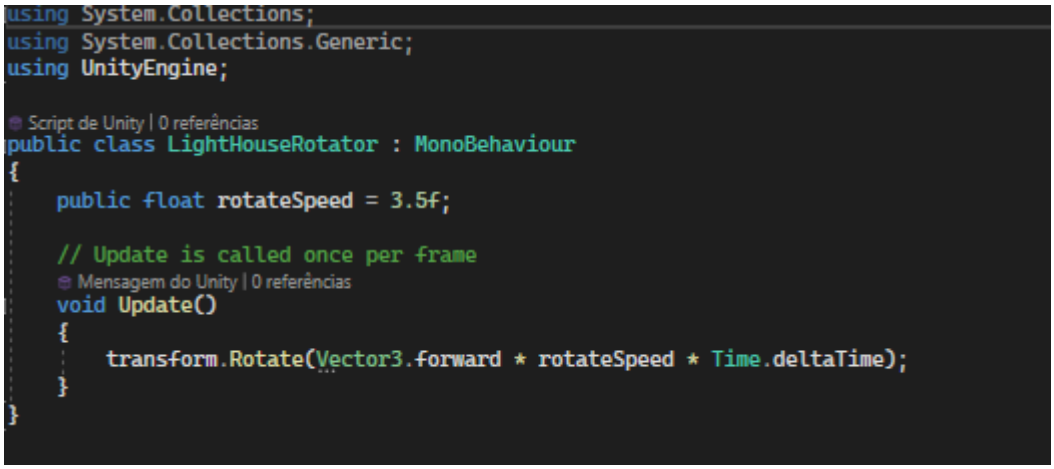
When the player possesses a lighter , this script turns it on or off when needed .



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

Script de Unity | 0 referências
public class LighterScript : MonoBehaviour
{
    public GameObject lighterObj;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void OnEnable()
    {
        lighterObj.SetActive(true);
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {

    }

    Mensagem do Unity | 0 referências
    void OnDisable()
    {
        lighterObj.SetActive(false);
    }
}
```

LightHouseRotator

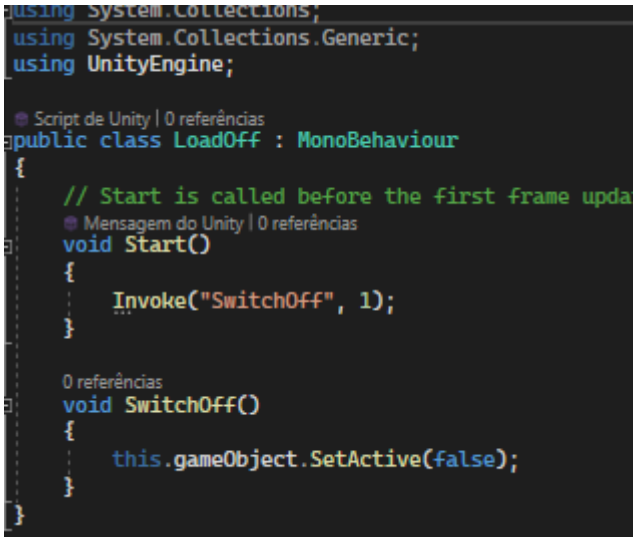
This script rotates the sphere that emits light in top of the lighthouse, completing its purpose of being a lighthouse.



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

Script de Unity | 0 referências
public class LightHouseRotator : MonoBehaviour
{
    public float rotateSpeed = 3.5f;

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        transform.Rotate(Vector3.forward * rotateSpeed * Time.deltaTime);
    }
}
```

LoadOff

This script controls the “loadOff” of the zombies.



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

Script de Unity | 0 referências
public class LoadOff : MonoBehaviour
{
    // Start is called before the first frame upda
    Mensagem do Unity | 0 referências
    void Start()
    {
        Invoke("SwitchOff", 1);
    }

    0 referências
    void SwitchOff()
    {
        this.gameObject.SetActive(false);
    }
}
```

LookMode

This script controls the vision mode , in the start it gets the post processing volume , the flashlight game object , it turns the flashlight , flashlight overlay and night vision overlay off as well as the inventory menu.

When F is pressed the flashlight turn on and it commences to drain the battery , when Its pressed again It turns off and stops the draining. This works also for the nightvision (N key).

When the inventory is opened (I key) , It assures that the nightvision and flashlight as well as their overlays are off and then It enables the inventory and Its post processing filters.

If the player had zoomed , when entered the inventory Its restablished the field of view to the original parameters (60).

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

Script de Unity | 0 referências
public class LookMode : MonoBehaviour
{
    private PostProcessVolume vol;
    public PostProcessProfile standard;
    public PostProcessProfile nightVision;
    public PostProcessProfile inventory;
    public GameObject nightVisionOverlay;
    public GameObject flashlightOverlay;
    public GameObject inventoryMenu;
    public GameObject combinePanel;
    private Light flashLight;
    private bool nightVisionOn = false;
    private bool flashLightOn = false;
    public GameObject pointer;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        vol = GetComponent<PostProcessVolume>();
        flashLight = GameObject.Find("FlashLight").GetComponent<Light>();
        flashLight.enabled = false;
        nightVisionOverlay.SetActive(false);
        flashlightOverlay.SetActive(false);
        inventoryMenu.SetActive(false);
        vol.profile = standard;
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.N))
        {
            if (SaveScript.inventoryOpen == false)
            {
                if (nightVisionOn == false)
                {
                    vol.profile = nightVision;
                    nightVisionOverlay.SetActive(true);
                    nightVisionOn = true;
```

```
                NightVisionOff();
            }
            else if (nightVisionOn == true)
            {
                vol.profile = standard;
                nightVisionOverlay.SetActive(false);
                nightVisionOverlay.GetComponent<NightVisionScript>().StopDrain();
                this.gameObject.GetComponent<Camera>().fieldOfView = 60;
                nightVisionOn = false;
            }
        }
    }

    if(Input.GetKeyDown(KeyCode.F))
    {
        if (SaveScript.inventoryOpen == false)
        {
            if (flashLightOn == false)
            {
                flashlightOverlay.SetActive(true);
                flashLight.enabled = true;
                flashLightOn = true;
                FlashLightSwitchOff();
            }
            else if (flashLightOn == true)
            {
                flashlightOverlay.SetActive(false);
                flashLight.enabled = false;
                flashlightOverlay.GetComponent<FlashLightScript>().StopDrain();
                flashLightOn = false;
            }
        }
    }

    if(Input.GetKeyDown(KeyCode.I))
    {
        if(SaveScript.inventoryOpen == false)
        {
            vol.profile = inventory;
            inventoryMenu.SetActive(true);

            if(flashLightOn == true)
            {
                flashlightOverlay.SetActive(false);
                flashLight.enabled = false;
                flashlightOverlay.GetComponent<FlashLightScript>().StopDrain();
```

```
                flashLightOn = false;
            }
            if(nightVisionOn == true)
            {
                nightVisionOverlay.SetActive(false);
                nightVisionOverlay.GetComponent<NightVisionScript>().StopDrain();
                this.gameObject.GetComponent<Camera>().fieldOfView = 60;
                nightVisionOn = false;
            }
        }
        else if (SaveScript.inventoryOpen == true)
        {
            vol.profile = standard;
            combinePanel.SetActive(false);
            inventoryMenu.SetActive(false);
        }
    }

    if(nightVisionOn == true)
    {
        NightVisionOff();
    }

    if(flashLightOn == true)
    {
        FlashLightSwitchOff();
    }

    if(SaveScript.inventoryOpen == true || Time.timeScale == 0)
    {
        Cursor.visible = true;
        pointer.SetActive(false);
    }
    else
    {
        Cursor.visible = false;
        pointer.SetActive(true);
    }
}

2 referências
private void NightVisionOff()
{
    if(nightVisionOverlay.GetComponent<NightVisionScript>().batteryPower <= 0)
    {
        vol.profile = standard;
```

```
            nightVisionOverlay.SetActive(false);
            this.gameObject.GetComponent<Camera>().fieldOfView = 60;
            nightVisionOn = false;
        }
    }

2 referências
private void FlashLightSwitchOff()
{
    if(flashlightOverlay.GetComponent<FlashLightScript>().batteryPower <= 0)
    {
        flashlightOverlay.SetActive(false);
        flashLight.enabled = false;
        flashlightOverlay.GetComponent<FlashLightScript>().StopDrain();
        flashLightOn = false;
    }
}
```

MainGUI

This controls the HUD levels of health , stamina and infection .The F0 is needed to format the numbers , not showing decimal parcels.

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

Script de Unity | 0 referências
public class MainGUI : MonoBehaviour
{
    public Text healthAmt;
    public Text staminaAmt;
    public Text infectionAmt;

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        healthAmt.text = SaveScript.health + "%";
        staminaAmt.text = SaveScript.stamina.ToString("F0") + "%";
        infectionAmt.text = SaveScript.infection.ToString("F0") + "%";
    }
}
```

Nightvision Script

This script controls the nightvision , its drain and the zoom.

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

Script de Unity | 4 referências
public class NightVisionScript : MonoBehaviour
{
    private Image zoomBar;
    private Image batteryChunks;
    private Camera cam;
    public float batteryPower = 1.0f;
    public float drainTime = 2;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        zoomBar = GameObject.Find("ZoomBar").GetComponent<Image>();
        batteryChunks = GameObject.Find("BatteryChunks").GetComponent<Image>();
        cam = GameObject.Find("FirstPersonCharacter").GetComponent<Camera>();
    }

    Mensagem do Unity | 0 referências
    private void OnEnable()
    {
        InvokeRepeating("BatteryDrain", drainTime, drainTime);
        if (zoomBar != null)
        zoomBar.fillAmount = 0.6f;
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(Input.GetAxis("Mouse ScrollWheel") > 0)
        {
            if(cam.fieldOfView > 10)
            {
                cam.fieldOfView -= 5;
                zoomBar.fillAmount = cam.fieldOfView / 100;
            }
        }
        if(Input.GetAxis("Mouse ScrollWheel") < 0)
        {
            if (cam.fieldOfView < 60)
            {
```

```
                cam.fieldOfView += 5;
                zoomBar.fillAmount = cam.fieldOfView / 100;
            }
        }
        batteryChunks.fillAmount = batteryPower;
    }

    0 referências
    private void BatteryDrain()
    {
        if (batteryPower > 0.0f)
            batteryPower -= 0.25f;
    }

    2 referências
    public void StopDrain()
    {
        CancelInvoke("BatteryDrain");
    }
```

PickupScript

The PickupsScript manages interactions with various game objects like weapons, items, ammo, doors, generators, and vaccines. It uses raycasting to detect objects in front of the player within a specified distance. When an object is detected, it displays a pickup panel with relevant information and allows the player to pick up the object using the "E" key. For weapons, items, and ammo, it updates the player's inventory and plays a pickup sound. For doors, it checks if the door is locked or requires power, updates the door message, and allows the player to open or close the door using the "E" key if it is unlocked. For generators and vaccines, it displays relevant messages. The script also handles shooting mechanics, applying damage to zombies when the player shoots them with specific weapons.

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

Script de Unity | 0 referências
public class PickupsScript : MonoBehaviour
{
    private RaycastHit hit;
    public LayerMask excludeLayers;
    public GameObject pickupPanel;
    public float pickupDisplayDistance = 3;

    public Image mainImage;
    public Sprite[] weaponIcons;
    public Sprite[] itemIcons;
    public Sprite[] ammoIcons;
    public Text mainTitle;
    public string[] weaponTitles;
    public string[] itemTitles;
    public string[] ammoTitles;

    private int objID = 0;
    private AudioSource audioPlayer;
    public GameObject doorMessageObj;
    public GameObject generatorMessageObj;
    public GameObject vaccineMessageObj;
    public Text doorMessage;
    public AudioClip[] pickupSounds;

    private RaycastHit gunHit;
    private RaycastHit[] shotgunHits;


    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        pickupPanel.SetActive(false);
        audioPlayer = GetComponent<AudioSource>();
        doorMessageObj.SetActive(false);
        generatorMessageObj.SetActive(false);
        vaccineMessageObj.SetActive(false);
    }
```

```
void Update()
{
    if(Physics.SphereCast(transform.position, 0.3f, transform.forward, out hit, 30, ~excludeLayers))
    {
        if (Vector3.Distance(transform.position, hit.transform.position) < pickupDisplayDistance)
        {
            if (hit.transform.gameObject.CompareTag("weapon"))
            {
                pickupPanel.SetActive(true);
                objID = (int)hit.transform.gameObject.GetComponent<WeaponType>().chooseWeapon;
                mainImage.sprite = weaponIcons[objID];
                mainTitle.text = weaponTitles[objID];

                if(Input.GetKeyDown(KeyCode.E))
                {
                    SaveScript.weaponAmts[objID]++;
                    audioPlayer.clip = pickupSounds[3];
                    audioPlayer.Play();
                    SaveScript.change = true;
                    Destroy(hit.transform.gameObject, 0.2f);
                }
            }
            else if (hit.transform.gameObject.CompareTag("item"))
            {
                pickupPanel.SetActive(true);
                objID = (int)hit.transform.gameObject.GetComponent<ItemsType>().chooseItem;
                mainImage.sprite = itemIcons[objID];
                mainTitle.text = itemTitles[objID];

                if (Input.GetKeyDown(KeyCode.E))
                {
                    SaveScript.itemAmts[objID]++;
                    audioPlayer.clip = pickupSounds[3];
                    audioPlayer.Play();
                    SaveScript.change = true;
                    Destroy(hit.transform.gameObject, 0.2f);
                }
            }
            else if (hit.transform.gameObject.CompareTag("ammo"))
            {
                pickupPanel.SetActive(true);
                objID = (int)hit.transform.gameObject.GetComponent<AmmoType>().chooseAmmo;
                mainImage.sprite = ammoIcons[objID];
                mainTitle.text = ammoTitles[objID];
```

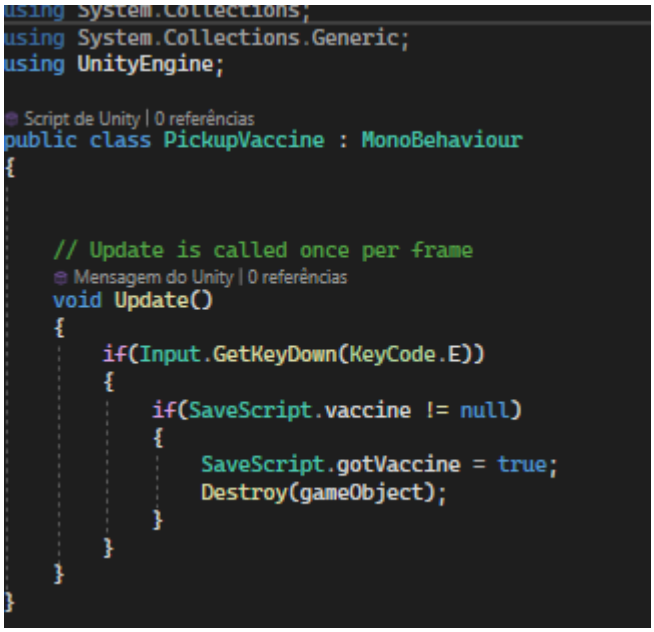
```
                if (Input.GetKeyDown(KeyCode.E))
                {
                    if (objID == 0)
                    {
                        SaveScript.ammoAmts[0] += 12;
                    }
                    if (objID == 1)
                    {
                        SaveScript.ammoAmts[1] += 8;
                    }
                    audioPlayer.clip = pickupSounds[3];
                    audioPlayer.Play();
                    SaveScript.change = true;
                    Destroy(hit.transform.gameObject, 0.2f);
                }
            }
            else if (hit.transform.gameObject.CompareTag("door"))
            {
                SaveScript.doorObject = hit.transform.gameObject;
                objID = (int)hit.transform.gameObject.GetComponent<DoorType>().chooseDoor;
                if(hit.transform.gameObject.GetComponent<DoorType>().locked == true)
                {
                    if (hit.transform.gameObject.GetComponent<DoorType>().electricDoor == false)
                    {
                        hit.transform.gameObject.GetComponent<DoorType>().message = "Locked. You need to use the " + hit.transform.gameObject.GetComponent<DoorType>().chooseDoor + " key";
                    }
                    if (hit.transform.gameObject.GetComponent<DoorType>().electricDoor == true && SaveScript.generatorOn == false)
                    {
                        hit.transform.gameObject.GetComponent<DoorType>().message = "This door needs a power supply to open";
                    }
                }
                if (hit.transform.gameObject.GetComponent<DoorType>().electricDoor == true && SaveScript.generatorOn == true)
                {
                    if (hit.transform.gameObject.GetComponent<DoorType>().opened == false)
                    {
                        hit.transform.gameObject.GetComponent<DoorType>().message = "Press E to open the door";
                    }
                }
                doorMessageObj.SetActive(true);
                doorMessage.text = hit.transform.gameObject.GetComponent<DoorType>().message;
                if (Input.GetKeyDown(KeyCode.E) && hit.transform.gameObject.GetComponent<DoorType>().locked == false)
                {
                    audioPlayer.clip = pickupSounds[objID];
                    audioPlayer.Play();
                    if (hit.transform.gameObject.GetComponent<DoorType>().opened == false)
```

```
                    hit.transform.gameObject.GetComponent<DoorType>().message = "Press E to close the door";
                    hit.transform.gameObject.GetComponent<DoorType>().opened = true;
                    hit.transform.gameObject.GetComponent<Animator>().SetTrigger("Open");
                }
                else if (hit.transform.gameObject.GetComponent<DoorType>().opened == true)
                {
                    hit.transform.gameObject.GetComponent<DoorType>().message = "Press E to open the door";
                    hit.transform.gameObject.GetComponent<DoorType>().opened = false;
                    hit.transform.gameObject.GetComponent<Animator>().SetTrigger("Close");
                }

            }
        }
        else if (hit.transform.gameObject.CompareTag("generator"))
        {
            SaveScript.generator = hit.transform.gameObject;
            if(SaveScript.generatorOn == false)
            {
                generatorMessageObj.SetActive(true);
            }
            if (SaveScript.generatorOn == true)
            {
                generatorMessageObj.SetActive(false);
            }
        }
        else if (hit.transform.gameObject.CompareTag("Vaccine"))
        {
            SaveScript.vaccine = hit.transform.gameObject;
            vaccineMessageObj.SetActive(true);
        }

    }
    else
    {
        pickupPanel.SetActive(false);
        doorMessageObj.SetActive(false);
        SaveScript.doorObject = null;

        generatorMessageObj.SetActive(false);
        SaveScript.generator = null;

        vaccineMessageObj.SetActive(false);
        SaveScript.vaccine = null;
    }
```

```
    if(Physics.SphereCast(transform.position, 0.01f, transform.forward, out gunHit, 500))
    {
        if(gunHit.transform.gameObject.name == "Body" && SaveScript.weaponID == 4)
        {
            if (Input.GetMouseButtonDown(0) && SaveScript.currentAmmo[4] > 0)
            {
                gunHit.transform.gameObject.GetComponent<ZombieGunDamage>().SendGunDamage(gunHit.point);
            }
        }
    }
    if(SaveScript.weaponID == 5 && SaveScript.currentAmmo[5] > 0)
    {
        shotgunHits = Physics.SphereCastAll(transform.position, 0.3f, transform.forward, 50);

        for(int i = 0; i < shotgunHits.Length; i++)
        {
            if(shotgunHits[i].transform.gameObject.name == "Body")
            {
                if(Input.GetMouseButtonDown(0))
                {
                    shotgunHits[i].transform.gameObject.GetComponent<ZombieGunDamage>().SendGunDamage(shotgunHits[i].point);
                }
            }
        }

    }
}
}
```

PickupVaccine

This script controls the pickup of the vaccine and It destroys the gameObject when picked.



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

Script de Unity | 0 referências
public class PickupVaccine : MonoBehaviour
{


    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.E))
        {
            if(SaveScript.vaccine != null)
            {
                SaveScript.gotVaccine = true;
                Destroy(gameObject);
            }
        }
    }
}
```

SaveScript

The SaveScript script manages various game state variables and player interactions. It keeps track of inventory items, weapons, ammo amounts, player health, stamina, and infection levels. At the start, it initializes these values and sets up the UI elements for zombie and death messages. During each update, it handles the player's stamina depletion and regeneration based on movement and actions, updates the infection rate over time, and monitors for changes in the inventory. If the health reaches zero or infection reaches 100%, it displays relevant messages and pauses the game after a delay. It also includes functionality to display or hide an FPS display based on key inputs and limits the number of zombies in the game by destroying excess ones. The script also manages the state of specific game objects like doors, generators, and vaccines through static references. Additionally, it has coroutine methods to reset a bottle's position after 30 seconds and to pause the game after displaying death or infection messages.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;
using UnityEngine.SceneManagement;

Script de Unity | 99+ referências
public class SaveScript : MonoBehaviour
{
    public static bool inventoryOpen = false;
    public static int weaponID = 0;
    public static bool[] weaponsPickedUp = new bool[9];
    public static int itemID = 0;
    public static bool[] itemsPickedUp = new bool[13];
    public static int[] weaponAmts = new int[9];
    public static int[] itemAmts = new int[13];
    public static bool change = false;
    public static int[] ammoAmts = new int[2];
    public static int[] currentAmmo = new int[9];
    public static float stamina;
    public static float infection;
    public static int health;
    public static GameObject doorObject;
    public static bool gunUsed = false;
    public static Vector3 bottlePos = new Vector3(0, 0, 0);
    private bool hasSmashed = false;
    public static bool isHidden = false;
    public static List<GameObject> zombiesChasing = new List<GameObject>();
    public static int zombiesInGameAmt = 0;
    public static bool generatorOn = false;
    public static GameObject generator;
    public static bool gotVaccine = false;
    public static GameObject vaccine;

    private GameObject[] zombies;

    public GameObject zombieMessage, deathMessage;
    public GameObject fpsDisplay;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        stamina = FirstPersonController.FPSstamina;
        health = 100;
        infection = 0;
        inventoryOpen = false;
```

```
        weaponID = 0;
        itemID = 0;
        stamina = 100;
        generatorOn = false;
        gotVaccine = false;
        weaponsPickedUp[0] = true;
        weaponAmts[0] = 1;
        itemsPickedUp[0] = true;
        itemsPickedUp[1] = true;
        itemAmts[0] = 1;
        itemAmts[1] = 1;
        ammoAmts[0] = 12;
        ammoAmts[1] = 2;

        for (int i = 0; i < currentAmmo.Length; i++)
        {
            currentAmmo[i] = 2;
        }
        currentAmmo[4] = 12;
        currentAmmo[6] = 0;

        zombieMessage.SetActive(false);
        deathMessage.SetActive(false);
        fpsDisplay.SetActive(false);
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(zombiesInGameAmt > 140)
        {
            zombies = GameObject.FindGameObjectsWithTag("zombie");
            for(int i = 140; i < zombies.Length; i++)
            {
                Destroy(zombies[i]);
            }
        }

        if(Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene(0);
        }

        if(Input.GetKeyDown(KeyCode.V))
        {
```

```
    {
        fpsDisplay.SetActive(true);
    }
    if (Input.GetKeyDown(KeyCode.B))
    {
        fpsDisplay.SetActive(false);
    }

    if (zombiesInGameAmt < 0)
    {
        zombiesInGameAmt = 0;
    }
    if (FirstPersonController.inventorySwitchedOn == true)
    {
        inventoryOpen = true;
    }
    if (FirstPersonController.inventorySwitchedOn == false)
    {
        inventoryOpen = false;
    }

    if(Input.GetAxis("Vertical") != 0 && Input.GetKey(KeyCode.LeftShift) && FirstPersonController.FPSstamina > 0.0f)
    {
        FirstPersonController.FPSstamina -= 10 * Time.deltaTime;
        stamina = FirstPersonController.FPSstamina;
    }
    if(stamina < 100)
    {
        FirstPersonController.FPSstamina += 3.35f * Time.deltaTime;
        stamina = FirstPersonController.FPSstamina;
    }
    if(stamina >= 100)
    {
        FirstPersonController.FPSstamina = stamina;
    }
    if(Input.GetMouseButtonDown(0) && stamina > 10 && weaponID < 4 && inventoryOpen == false)
    {
        FirstPersonController.FPSstamina -= 10;
        stamina = FirstPersonController.FPSstamina;
    }

    if(Input.GetKey(KeyCode.C))
    {
        FirstPersonController.FPSstamina -= 10f * Time.deltaTime;
        stamina = FirstPersonController.FPSstamina;
    }
```

```
if (infection < 50)
{
    infection += 0.1f * Time.deltaTime;
}
if (infection > 49 && infection < 100)
{
    infection += 0.4f * Time.deltaTime;
}

if (change == true)
{
    change = false;
    for(int i = 1; i < weaponAmts.Length; i++)
    {
        if(weaponAmts[i] > 0)
        {
            weaponsPickedUp[i] = true;
        }
        else if (weaponAmts[i] == 0)
        {
            weaponsPickedUp[i] = false;
        }
    }
    for (int i = 2; i < itemAmts.Length; i++)
    {
        if (itemAmts[i] > 0)
        {
            itemsPickedUp[i] = true;
        }
        else if (itemAmts[i] == 0)
        {
            itemsPickedUp[i] = false;
        }
    }
}

if(bottlePos != Vector3.zero)
{
    if(hasSmashed == false)
    {
        StartCoroutine(ResetBottlePos());
        hasSmashed = true;
    }
}
```

```
        if(health <= 0)
        {
            deathMessage.SetActive(true);
            StartCoroutine(PauseTime());
        }

        if (infection >= 100)
        {
            zombieMessage.SetActive(true);
            StartCoroutine(PauseTime());
        }

    }

    2 referências
    IEnumerator PauseTime()
    {
        yield return new WaitForSeconds(3.2f);
        Time.timeScale = 0;
    }

    1 referência
    IEnumerator ResetBottlePos()
    {
        yield return new WaitForSeconds(30);
        bottlePos = Vector3.zero;
        hasSmashed = false;
    }

}
```

SpawnDirection

The SpawnDirection script controls the spawn direction for zombies. It has two variables, forward and back. When the player enters the trigger collider , the script checks the forward and back booleans. If forward is true, it sets the spawnForward property of ZombieSpawnPatrol component to true, making the zombies spawn fordward . If back is true, it sets spawnForward to false, indicating zombies should spawn backward.

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

Script de Unity | 2 referências
public class SpawnDirection : MonoBehaviour
{
    public bool forward = true;
    public bool back = false;
    public Transform[] targetList;

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player"))
        {
            if(forward == true)
            {
                gameObject.GetComponentInParent<ZombieSpawnPatrol>().spawnForward = true;
            }
            if (back == true)
            {
                gameObject.GetComponentInParent<ZombieSpawnPatrol>().spawnForward = false;
            }
        }
    }
}
```

SprayScript

This script controlls the drain of the spray when pressed.

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

Script de Unity | 5 referências
public class SprayScript : MonoBehaviour
{
    public Image sprayFill;
    public float sprayAmount = 1.0f;
    public float drainTime = 0.1f;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void OnEnable()
    {
        sprayFill.fillAmount = sprayAmount;
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(Input.GetMouseButton(0))
        {
            sprayAmount -= drainTime * Time.deltaTime;
            sprayFill.fillAmount = sprayAmount;
        }
    }
}
```

UIScale

This script mantains the size of the UI independently of the resolution of the screen.

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

Script de Unity | 0 referências
public class UIScale : MonoBehaviour
{
    public float scaleValue = 1;
    public float UHDScale = 2;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        if(Screen.width > 1920)
        {
            scaleValue = UHDScale;
        }
        this.transform.localScale = new Vector3(scaleValue, scaleValue, scaleValue);
    }
}
```

WeaponInventory

This script works as the itemInventory, displays the first weapon when turned on , gets the Ammo for the weapon, and the audiosource component for the inventory.

If the weapon is the molotov or the spray can it shows the combine button , if the player posess a lighter it can combine it with the spray can . In addition if the player has cloth , it can combine it with a a lighter and a bottle to make a molotov.

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

Script de Unity | 0 referências
public class WeaponInventory : MonoBehaviour
{
    public Sprite[] bigIcons;
    public Image bigIcon;
    public string[] titles;
    public Text title;
    public string[] descriptions;
    public Text description;
    public Button[] weaponButtons;
    public Text amtsText;

    private AudioSource audioPlayer;
    public AudioClip click, select;
    private int chosenWeaponNumber;

    public GameObject useButton, combineButton;
    public GameObject combinePanel, combineUseButton;
    public Image[] combineItems;
    public GameObject sprayPanel;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        audioPlayer = GetComponent<AudioSource>();

        bigIcon.sprite = bigIcons[0];
        title.text = titles[0];
        description.text = descriptions[0];
        amtsText.text = "Amts: 1";
        combinePanel.SetActive(false);
        combineButton.SetActive(false);
    }

    Mensagem do Unity | 0 referências
    private void OnEnable()
    {
        for(int i = 0; i < weaponButtons.Length; i++)
        {
            if(SaveScript.weaponsPickedUp[i] == false)
            {
```

```
                weaponButtons[i].image.color = new Color(1, 1, 1, 0.06f);
                weaponButtons[i].image.raycastTarget = false;
            }
            if (SaveScript.weaponsPickedUp[i] == true)
            {
                weaponButtons[i].image.color = new Color(1, 1, 1, 1);
                weaponButtons[i].image.raycastTarget = true;
            }
        }

        if(chosenWeaponNumber < 6)
        {
            combinePanel.SetActive(false);
            combineButton.SetActive(false);
        }

        if(SaveScript.itemsPickedUp[2] == true)
        {
            combineItems[0].color = new Color(1, 1, 1, 1);
        }
        if (SaveScript.itemsPickedUp[2] == false)
        {
            combineItems[0].color = new Color(1, 1, 1, 0.06f);
        }
        if (SaveScript.itemsPickedUp[3] == true)
        {
            combineItems[1].color = new Color(1, 1, 1, 1);
        }
        if (SaveScript.itemsPickedUp[3] == false)
        {
            combineItems[1].color = new Color(1, 1, 1, 0.06f);
        }

        if(SaveScript.weaponAmts[chosenWeaponNumber] <= 0)
        {
            ChooseWeapon(0);
        }

        ChooseWeapon(chosenWeaponNumber);
    }

    2 referências
    public void ChooseWeapon(int weaponNumber)
    {
        bigIcon.sprite = bigIcons[weaponNumber];
        title.text = titles[weaponNumber];
```

```
        description.text = descriptions[weaponNumber];
        if (audioPlayer != null)
        {
            audioPlayer.clip = click;
            audioPlayer.Play();
        }
        chosenWeaponNumber = weaponNumber;
        amtsText.text = "Amts: " + SaveScript.weaponAmts[weaponNumber];

        if (chosenWeaponNumber > 5)
        {
            combineButton.SetActive(true);
            combinePanel.SetActive(false);
        }
        if (chosenWeaponNumber < 6)
        {
            combinePanel.SetActive(false);
            combineButton.SetActive(false);
        }

        if(chosenWeaponNumber == 6)
        {
            useButton.SetActive(false);
        }
        else
        {
            useButton.SetActive(true);
        }

    }

    0 referências
    public void CombineAction()
    {
        combinePanel.SetActive(true);

        if(chosenWeaponNumber == 6)
        {
            combineItems[1].transform.gameObject.SetActive(false);
            if(SaveScript.itemsPickedUp[2] == true)
            {
                combineUseButton.SetActive(true);
            }
            if (SaveScript.itemsPickedUp[2] == false)
            {
                combineUseButton.SetActive(false);
```

```
        }
        if (chosenWeaponNumber == 7)
        {
            combineItems[1].transform.gameObject.SetActive(true);
            if (SaveScript.itemsPickedUp[2] == true && SaveScript.itemsPickedUp[3] == true)
            {
                combineUseButton.SetActive(true);
            }
            if (SaveScript.itemsPickedUp[2] == false || SaveScript.itemsPickedUp[3] == false)
            {
                combineUseButton.SetActive(false);
            }
        }
    }

    0 referências
    public void CombineAssignWeapon()
    {
        if (chosenWeaponNumber == 6)
        {
            SaveScript.weaponID = chosenWeaponNumber;
            if (sprayPanel.GetComponent<SprayScript>().sprayAmount <= 0.0f)
            {
                sprayPanel.GetComponent<SprayScript>().sprayAmount = 1.0f;
            }
        }
        if (chosenWeaponNumber == 7)
        {
            SaveScript.weaponID = chosenWeaponNumber += 1;
        }
        audioPlayer.clip = select;
        audioPlayer.Play();
    }

    0 referências
    public void AssignWeapon()
    {
        SaveScript.weaponID = chosenWeaponNumber;
        audioPlayer.clip = select;
        audioPlayer.Play();
    }
}
```

WeaponManager

The WeaponManager script is responsible for handling the player's weapons. It declares an enumeration called weaponSelect that categorizes various types of weapons. The code assigns the chosen weapon, deals with weapon switches, and controls attacks and reloading. In the start function, the selected weapon, animator, and audio source are initialized. Within the Update function, it examines the current state of the game, makes adjustments to weapons, deals with attacks, uses spray cans, and reloads. The changeWeapons function turns off all weapons and turns on the chosen one, while also updating the animator and weapon's position. The position of the weapon is determined by its type in the Move method. The bottle throwing states are managed by the BottleThrowEmpty and BottleThrowFire techniques. The methods LoadAnotherBottle and LoadAnotherFireBottle both serve to refill bottles. Both the WeaponReset and StartSpraySound coroutines handle the states of weapon animations and the play of the spray sound.

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

Script de Unity | 4 referências
public class WeaponManager : MonoBehaviour
{
    10 referências
    public enum weaponSelect
    {
        knife,
        cleaver,
        bat,
        axe,
        pistol,
        shotgun,
        sprayCan,
        bottle,
        bottleWithCloth
    }

    public weaponSelect chosenWeapon;
    public GameObject[] weapons;
    //private int weaponID = 0;
    private Animator anim;
    private AudioSource audioPlayer;
    public AudioClip[] weaponSounds;
    private int currentWeaponID;
    private bool spraySoundOn = false;
    public GameObject sprayPanel;
    public static bool emptyBottleThrow = false;
    public static bool fireBottleThrow = false;
    private AnimatorStateInfo animInfo;
    private bool canAttack = true;
    private bool sprayEmpty = false;
    private bool stopSpray = false;
    public AudioClip[] reloadSounds;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        SaveScript.weaponID = (int)chosenWeapon;
        anim = GetComponent<Animator>();
        audioPlayer = GetComponent<AudioSource>();
        ChangeWeapons();
    }
```

```
void Update()
{
    if (Time.timeScale == 1)
    {
        animInfo = anim.GetCurrentAnimatorStateInfo(0);
        if (animInfo.IsTag("BottleThrown"))
        {
            canAttack = false;
        }
        else
        {
            canAttack = true;
        }
        if (SaveScript.weaponID != currentWeaponID)
        {
            ChangeWeapons();
        }

        if (Input.GetMouseButtonDown(0) && canAttack == true)
        {
            if (SaveScript.inventoryOpen == false)
            {
                if (SaveScript.currentAmmo[SaveScript.weaponID] > 0 && SaveScript.stamina > 20)
                {
                    anim.SetTrigger("Attack");
                    audioPlayer.clip = weaponSounds[SaveScript.weaponID];
                    audioPlayer.Play();

                    if (SaveScript.weaponID == 4 || SaveScript.weaponID == 5)
                    {
                        SaveScript.currentAmmo[SaveScript.weaponID]--;
                        SaveScript.gunUsed = true;
                    }
                }
                else
                {
                    if (SaveScript.weaponID == 4 || SaveScript.weaponID == 5)
                    {
                        audioPlayer.clip = weaponSounds[9];
                        audioPlayer.Play();
                    }
                }
            }
        }
        if (Input.GetMouseButton(0) && sprayPanel.GetComponent<SprayScript>().sprayAmount > 0.0f)
        {
```

```
            sprayEmpty = false;
            stopSpray = false;
            if (SaveScript.weaponID == 6 && SaveScript.inventoryOpen == false)
            {
                if (spraySoundOn == false)
                {
                    spraySoundOn = true;
                    anim.SetTrigger("Attack");
                    StartCoroutine(StartSpraySound());

                }
            }
        }
        if (Input.GetMouseButtonUp(0) || sprayPanel.GetComponent<SprayScript>().sprayAmount <= 0.0f)
        {
            if (SaveScript.weaponID == 6 && SaveScript.inventoryOpen == false && stopSpray == false)
            {
                stopSpray = true;
                anim.SetTrigger("Release");
                spraySoundOn = false;
                audioPlayer.Stop();
                audioPlayer.loop = false;
            }
        }
        if (sprayPanel.GetComponent<SprayScript>().sprayAmount <= 0 && sprayEmpty == false)
        {
            sprayEmpty = true;
            SaveScript.weaponAmts[6]--;
            if (SaveScript.weaponAmts[6] == 0)
            {
                SaveScript.weaponsPickedUp[6] = false;
            }
        }

        if (SaveScript.weaponID == 4 || SaveScript.weaponID == 5)
        {
            if (Input.GetMouseButtonDown(1))
            {
                if (SaveScript.ammoAmts[SaveScript.weaponID - 4] > 0)
                {
                    SaveScript.currentAmmo[SaveScript.weaponID] += SaveScript.ammoAmts[SaveScript.weaponID - 4];
                    SaveScript.ammoAmts[SaveScript.weaponID - 4] = 0;
                    anim.SetTrigger("Reload");
                    audioPlayer.clip = reloadSounds[SaveScript.weaponID - 4];
                    audioPlayer.Play();
                }
```

```
public void ChangeWeapons()
{
    foreach (GameObject weapon in weapons)
    {
        weapon.SetActive(false);
    }
    weapons[SaveScript.weaponID].SetActive(true);
    chosenWeapon = (weaponSelect)SaveScript.weaponID;
    anim.SetInteger("WeaponID", SaveScript.weaponID);
    anim.SetBool("weaponChanged", true);
    currentWeaponID = SaveScript.weaponID;

    Move();
    StartCoroutine(WeaponReset());
}

1 referência
private void Move()
{
    switch (chosenWeapon)
    {
        case weaponSelect.knife:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.cleaver:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.bat:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.axe:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.pistol:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.shotgun:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.46f);
            break;
        case weaponSelect.sprayCan:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
        case weaponSelect.bottle:
            transform.localPosition = new Vector3(0.02f, -0.193f, 0.66f);
            break;
    }
```

```
public void BottleThrowEmpty()
{
    emptyBottleThrow = true;
}

0 referências
public void BottleThrowFire()
{
    fireBottleThrow = true;
}

0 referências
public void LoadAnotherBottle()
{
    if(SaveScript.weaponID == 7)
    {
        ChangeWeapons();
    }
}

0 referências
public void LoadAnotherFireBottle()
{
    if (SaveScript.weaponID == 8)
    {
        ChangeWeapons();
    }
}

1 referência
IEnumerator WeaponReset()
{
    yield return new WaitForSeconds(0.5f);
    anim.SetBool("weaponChanged", false);
}

1 referência
IEnumerator StartSpraySound()
{
    yield return new WaitForSeconds(0.3f);
    audioPlayer.clip = weaponSounds[SaveScript.weaponID];
    audioPlayer.Play();
    audioPlayer.loop = true;
}
```

WeaponUIManager

This script shows an overlay depending on the weapon type if applicable (spray can with lighter, shotgun, and pistol) and controls the ammo for the pistol and shotgun.

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

Script de Unity | 0 referências
public class WeaponsUIManager : MonoBehaviour
{
    public GameObject pistolPanel, shotgunPanel, sprayPanel;
    public Text pistolTotalAmmo, pistolCurrentAmmo, shotgunTotalAmmo, shotgunCurrentAmmo;
    private bool panelOn = false;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        pistolPanel.SetActive(false);
        shotgunPanel.SetActive(false);
        sprayPanel.SetActive(false);
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(SaveScript.weaponID == 4)
        {
            if(panelOn == false)
            {
                panelOn = true;
                pistolPanel.SetActive(true);
            }
        }
        if (SaveScript.weaponID == 5)
        {
            if (panelOn == false)
            {
                panelOn = true;
                shotgunPanel.SetActive(true);
            }
        }
        if (SaveScript.weaponID == 6)
        {
            if (panelOn == false)
            {
                panelOn = true;
                sprayPanel.SetActive(true);
```

```
        }
        if (SaveScript.inventoryOpen == true)
        {
            pistolPanel.SetActive(false);
            shotgunPanel.SetActive(false);
            sprayPanel.SetActive(false);
            panelOn = false;
        }

    }

    Mensagem do Unity | 0 referências
    private void OnGUI()
    {
        pistolTotalAmmo.text = SaveScript.ammoAmts[0].ToString();
        shotgunTotalAmmo.text = SaveScript.ammoAmts[1].ToString();
        pistolCurrentAmmo.text = SaveScript.currentAmmo[4].ToString();
        shotgunCurrentAmmo.text = SaveScript.currentAmmo[5].ToString();
    }
```

WeaponType

This script works as the itemsType , It categorizes the weapons.

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

Script de Unity | 1 referência
public class WeaponType : MonoBehaviour
{
    1 referência
    public enum typeOfWeapon
    {
        knife,
        cleaver,
        bat,
        axe,
        pistol,
        shotgun,
        sprayCan,
        bottle,
        bottleWithCloth
    }

    public typeOfWeapon chooseWeapon;

}
```

WinScript

This script works when the player enters the winZone with the needed conditions (has the vaccine) , It shows the player the winScreen.

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

Script de Unity | 0 referências
public class WinScript : MonoBehaviour
{
    public GameObject winMessage;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        winMessage.SetActive(false);
    }

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player"))
        {
            if(SaveScript.gotVaccine == true)
            {
                winMessage.SetActive(true);
                Time.timeScale = 0;
            }
        }
    }
}
```

ZombieAttack

This script controlls the damage , infection and effects of the zombie attacks. If the zombie attacks it deals 3 damage on the health and 3% infection. When hit it displays the blood effect and It plays the effect sound.

```
Script de Unity | 0 referências
public class ZombieAttack : MonoBehaviour
{
    private bool canDamage = false;
    private Collider col;
    private Animator bloodEffect;
    private AudioSource hitSound;
    public int damageAmt = 3;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        col = GetComponent<Collider>();
        bloodEffect = GameObject.Find("Blood").GetComponent<Animator>();
        hitSound = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(col.enabled == false)
        {
            canDamage = true;
        }
    }

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player"))
        {
            if(canDamage == true)
            {
                canDamage = false;
                if (SaveScript.health > 0)
                {
                    SaveScript.health -= damageAmt;
                }
                if (SaveScript.infection < 100)
                {
                    SaveScript.infection += damageAmt;
                }
                bloodEffect.SetTrigger("blood");
                hitSound.Play();
            }
        }
    }
}
```

ZombieDamage

This script manages zombie damage and death animations. It reduces zombie health when hit by specified weapons, shows blood splatters, plays damage sounds, and triggers animations. It detects hits using a trigger collider and handles instant death from gunshots and delayed death from fire. If zombie health reaches zero, it triggers death animations and sets the zombie to a dead state.

```
public class ZombieDamage : MonoBehaviour
{
    private bool damaging = true;
    private int zombieHealth = 100;
    private Animator zombieAnim;
    private AudioSource damagePlayer;
    private bool death = false;
    public GameObject bloodSplat;
    public string[] weaponTag;
    public int[] damageAmts;
    public AudioClip[] damageSounds;

    private bool flameDeath = false;

    // Start is called before the first frame update
    Mensagem do Unity | 0 referências
    void Start()
    {
        zombieAnim = GetComponentInParent<Animator>();
        damagePlayer = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if(Input.GetMouseButtonDown(0))
        {
            damaging = true;
        }

        if (zombieHealth <= 0)
        {
            if(death == false)
            {
                death = true;
                zombieAnim.SetTrigger("dead");
                zombieAnim.SetBool("isDead", true);
            }
        }

    }

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        for(int i = 0; i < weaponTag.Length; i++)
        {
            if(other.CompareTag(weaponTag[i]))
            {
                if(damaging == true)
```

```
                damaging = false;
                zombieHealth -= damageAmts[i];
                Vector3 pos = other.ClosestPoint(transform.position);
                Instantiate(bloodSplat, pos, other.transform.rotation);
                this.transform.gameObject.GetComponentInParent<ZombieScript>().isAngry = true;
                damagePlayer.clip = damageSounds[i];
                damagePlayer.Play();
                if(weaponTag[i] == "bat")
                {
                    zombieAnim.SetTrigger("react");
                }
                if(weaponTag[i] == "axe")
                {
                    zombieAnim.SetTrigger("axeReact");
                }

            }
        }
    }
}

1 referência
public void gunDamage(Vector3 hitPoint)
{
    zombieHealth -= 100;
    if (death == false)
    {
        Instantiate(bloodSplat, hitPoint, this.transform.rotation);
        death = true;
        zombieAnim.SetTrigger("dead");
        zombieAnim.SetBool("isDead", true);
    }
}

1 referência
public void FlameDeath()
{
    if(flameDeath == false)
    {
        flameDeath = true;
        StartCoroutine(ZombieFireWalk());
    }
}

1 referência
IEnumerator ZombieFireWalk()
{
    yield return new WaitForSeconds(5);
    if (death == false)
    {
        death = true;
```

```
bieAnim.SetTrigger("fireDie");
bieAnim.SetBool("isDead", true);
```

ZombieGunDamage

This script handles damage to zombies caused by guns and fire. It has a reference to the zombie damage object, flames, a burned skin material… On start, it initializes the animator. When a gunshot hits the zombie, it calls the gunDamage method on the ZombieDamage script. If the zombie collides with fire particles, it triggers the flame death sequence, activates the flames, changes the skin material to a burned appearance, and sets the burn animation trigger.

```
public class ZombieGunDamage : MonoBehaviour
{
    public GameObject zombieDamageObj;
    public GameObject flames;
    public Material skinBurn;
    public GameObject[] LODs;

    private Animator bodyAnim;


    private void Start()
    {
        bodyAnim = GetComponent<Animator>();
    }

    public void SendGunDamage(Vector3 hitPoint)
    {
        zombieDamageObj.GetComponent<ZombieDamage>().gunDamage(hitPoint);
    }

    private void OnParticleCollision(GameObject other)
    {
        zombieDamageObj.GetComponent<ZombieDamage>().FlameDeath();
        flames.SetActive(true);
        foreach (GameObject skin in LODs)
        {
            skin.GetComponent<Renderer>().material = skinBurn;
        }

        bodyAnim.SetTrigger("burn");
    }
}
```

ZombieScript

This script controls the behavior of zombies in the game. It defines zombie types (shuffle, dizzy, alert) and states (Idle, Walking, Eating). The script manages animations, navigation, and interactions with the player and targets. It initializes variables like animator, nav mesh agent, audio source, and targets. The zombie's behavior changes based on its distance to the player and targets, random state changes, and whether it detects the player or other stimuli like a thrown bottle or gunshots.

When the player is close, the zombie stops and attacks. If the player is far, the zombie chases them or moves to random targets. The script also handles chase music based on the number of zombies chasing the player. If the player uses a gun, the zombie's alert range increases temporarily. The zombie can also react to being hidden or start in a house with specific behavior. When the zombie dies, it is removed from the list of chasing zombies and destroyed after a delay.

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

Script de Unity | 1 referência
public class ZombieScript : MonoBehaviour
{
    2 referências
    public enum ZombieType
    {
        shuffle,
        dizzy,
        alert
    }
    7 referências
    public enum ZombieState
    {
        Idle,
        Walking,
        Eating
    }

    public ZombieType zombieStyle;
    public ZombieState chooseState;
    public float yAdjustment = 0.0f;
    private Animator anim;
    private AnimatorStateInfo animInfo;
    private NavMeshAgent agent;
    public bool randomState = false;
    public float randomTiming = 5f;
    private int newState = 0;
    private int currentState;
    private GameObject[] targets;
    private float[] walkSpeed = { 0.15f, 1.0f, 0.75f };
    private float distanceToTarget;
    private int currentTarget = 0;
    private float distanceToPlayer;
    private GameObject player;
    private float zombieAlertRange = 20f;
    private bool awareOfPlayer = false;
    private bool adding = true;
    private AudioSource chaseMusicPlayer;
    private float attackDistance = 2.0f;
    private float rotateSpeed = 2.5f;
    private AudioSource zombieSound;

    private float gunAlertRange = 400;
    public bool isAngry = false;
```

```
private float hiddenRange = 2;
public bool startInHouse = false;

private float alertSpeed;

// Start is called before the first frame update
Mensagem do Unity | 0 referências
void Start()
{
    anim = GetComponent<Animator>();
    agent = GetComponent<NavMeshAgent>();
    zombieSound = GetComponent<AudioSource>();
    targets = GameObject.FindGameObjectsWithTag("Target");
    player = GameObject.Find("FPSController");
    chaseMusicPlayer = GameObject.Find("ChaseMusic").GetComponent<AudioSource>();
    zombieAlertRange = Random.Range(55f, 200f);
    anim.SetLayerWeight(((int)zombieStyle + 1), 1);
    if(zombieStyle == ZombieType.shuffle)
    {
        transform.position = new Vector3(transform.position.x, transform.position.y + yAdjustment, transform.position.z);
    }
    anim.SetTrigger(chooseState.ToString());
    currentState = (int)chooseState;
    if(randomState == true)
    {
        InvokeRepeating("SetAnimState", randomTiming, randomTiming);
    }
    agent.destination = targets[0].transform.position;
    agent.speed = walkSpeed[(int)zombieStyle];
    alertSpeed = Random.Range(1.0f, 2.0f);
    currentTarget = Random.Range(0, targets.Length);
}

// Update is called once per frame
Mensagem do Unity | 0 referências
void Update()
{
    if (anim.GetBool("isDead") == false)
    {
        distanceToPlayer = Vector3.Distance(transform.position, player.transform.position);
        if (SaveScript.bottlePos == Vector3.zero)
        {
            isAngry = false;
        }
        if (SaveScript.bottlePos != Vector3.zero && distanceToPlayer > attackDistance && isAngry == false)
        {
            agent.destination = SaveScript.bottlePos;
            anim.SetBool("attacking", false);
            chooseState = ZombieState.Walking;
        }
        else
```

```
    {
        if (distanceToPlayer <= attackDistance)
        {
            agent.isStopped = true;
            anim.SetBool("attacking", true);
            anim.speed = 1.0f;
            Vector3 pos = (player.transform.position - transform.position).normalized;
            Quaternion posRotation = Quaternion.LookRotation(new Vector3(pos.x, 0, pos.z));
            transform.rotation = Quaternion.Slerp(transform.rotation, posRotation, rotateSpeed * Time.deltaTime);
        }
        else
        {
            anim.SetBool("attacking", false);

            if (SaveScript.zombiesChasing.Count > 0)
            {
                if (chaseMusicPlayer.volume < 0.4f)
                {
                    if (chaseMusicPlayer.isPlaying == false)
                    {
                        chaseMusicPlayer.Play();
                    }
                    chaseMusicPlayer.volume += 0.5f * Time.deltaTime;
                }
            }
            if (SaveScript.zombiesChasing.Count == 0)
            {
                if (chaseMusicPlayer.volume > 0.0f)
                {
                    chaseMusicPlayer.volume -= 0.5f * Time.deltaTime;
                }
                if (chaseMusicPlayer.volume == 0.0f)
                {
                    chaseMusicPlayer.Stop();
                }
            }

            distanceToTarget = Vector3.Distance(transform.position, targets[currentTarget].transform.position);
            animInfo = anim.GetCurrentAnimatorStateInfo((int)zombieStyle);

            if (distanceToPlayer < zombieAlertRange && chooseState == ZombieState.Walking )
            {
                agent.destination = player.transform.position;
                awareOfPlayer = true;
                anim.speed = alertSpeed;
                if (adding == true)
                {
                    if (SaveScript.zombiesChasing.Contains(this.gameObject))
                    {
                        adding = false;
                        return;
```

```
            }
            else
            {
                SaveScript.zombiesChasing.Add(this.gameObject);
                adding = false;
            }
        }
    }

    if (distanceToPlayer < 10 && startInHouse == true)
    {
        agent.destination = player.transform.position;
        awareOfPlayer = true;
        chooseState = ZombieState.Walking;
        anim.SetTrigger("Walking");
        ZombiesStartInHouse();
        if (adding == true)
        {
            if (SaveScript.zombiesChasing.Contains(this.gameObject))
            {
                adding = false;
                return;
            }
            else
            {
                SaveScript.zombiesChasing.Add(this.gameObject);
                adding = false;
            }
        }
    }

    if (distanceToPlayer > zombieAlertRange)
    {
        awareOfPlayer = false;
        anim.speed = 1.0f;
        if (SaveScript.zombiesChasing.Contains(this.gameObject))
        {
            SaveScript.zombiesChasing.Remove(this.gameObject);
            adding = true;
        }
    }

    if(distanceToPlayer > 200)
    {
        awareOfPlayer = false;
        if (SaveScript.zombiesChasing.Contains(this.gameObject))
        {
            SaveScript.zombiesChasing.Remove(this.gameObject);
            adding = true;
        }
```

```
                    Destroy(gameObject);
                }

                if (animInfo.IsTag("motion"))
                {
                    if (anim.IsInTransition((int)zombieStyle))
                    {
                        agent.isStopped = true;
                    }
                }

                if (chooseState == ZombieState.Walking)
                {
                    if (distanceToTarget < 1.5f)
                    {
                        if (currentTarget < targets.Length - 1)
                        {
                            currentTarget = Random.Range(0, targets.Length);
                        }
                    }
                }
            }
        }
    }

    else
    {
        if (SaveScript.zombiesChasing.Contains(this.gameObject))
        {
            SaveScript.zombiesChasing.Remove(this.gameObject);
            adding = true;
        }

        if (SaveScript.zombiesChasing.Count == 0)
        {
            if (chaseMusicPlayer.volume > 0.0f)
            {
                chaseMusicPlayer.volume -= 0.5f * Time.deltaTime;
            }
            if (chaseMusicPlayer.volume == 0.0f)
            {
                chaseMusicPlayer.Stop();
            }
        }
        CancelInvoke();
        Destroy(gameObject, 20);
    }

    if(SaveScript.gunUsed == true)
    {
```

```
    {
        zombieAlertRange = gunAlertRange;
        StartCoroutine(ResetGunRange());
    }
    else if (SaveScript.isHidden == true)
    {
        zombieAlertRange = hiddenRange;
    }
    else
    {
        zombieAlertRange = 20;
    }
}

Mensagem do Unity | 0 referências
private void OnDestroy()
{
    SaveScript.zombiesInGameAmt--;
}

0 referências
void SetAnimState()
{
    if (awareOfPlayer == false)
    {
        newState = Random.Range(0, 3);
        if (newState != currentState)
        {
            chooseState = (ZombieState)newState;
            currentState = (int)chooseState;
            anim.SetTrigger(chooseState.ToString());
        }
    }
    if(awareOfPlayer == true)
    {
        chooseState = ZombieState.Walking;
    }
    zombieSound.Play();
}

0 referências
public void WalkOn()
{
    agent.isStopped = false;
    agent.destination = targets[currentTarget].transform.position;
}

0 referências
public void WalkOff()
{
    agent.isStopped = true;
}
```

```
1 referência
void ZombiesStartInHouse()
{
    if(chaseMusicPlayer.isPlaying == false)
    {
        chaseMusicPlayer.Play();
    }
    chaseMusicPlayer.volume = 0.4f;
}

1 referência
IEnumerator ResetGunRange()
{
    yield return new WaitForSeconds(10);
    SaveScript.gunUsed = false;
}
```

ZombieSpawnPatrol

This script manages zombie spawning in the game. It uses arrays for spawn points and zombie types, and controls how many zombies can be active (max 140). The script starts by initializing variables and checking if zombies should spawn in a house. During gameplay, it updates timers to regulate when new zombies can spawn. When the player enters a trigger area, it determines where zombies should spawn based on whether they're supposed to spawn forward or backward.If the zombies arent at a max, they are spawned at designated points. This script ensures zombies appear strategically and maintain game challenge.

```
public class ZombieSpawnPatrol : MonoBehaviour
{
    public Transform[] spawnPoints;
    public GameObject[] zombies;
    public int zombieSpawnAmt = 3;

    private float reSpawnTimer = 10;
    private float resetTimer = 0;
    [HideInInspector]
    public bool canSpawn = true;

    public bool houseSpawn = false;
    [HideInInspector]
    public bool spawnForward = true;
    public GameObject goingForward;
    public GameObject goingBack;


    Mensagem do Unity | 0 referências
    private void Start()
    {
        if(houseSpawn == true)
        {
            canSpawn = false;
        }
    }
    // Update is called once per frame
    Mensagem do Unity | 0 referências
    void Update()
    {
        if (canSpawn == false && houseSpawn == false)
        {
            resetTimer += 1 * Time.deltaTime;
            if(resetTimer >= reSpawnTimer)
            {
                canSpawn = true;
                resetTimer = 0;
            }
        }
    }

    Mensagem do Unity | 0 referências
    private void OnTriggerEnter(Collider other)
    {
        if(spawnForward == true)
        {
            spawnPoints = goingForward.GetComponent<SpawnDirection>().targetList;
        }
        if (spawnForward == false)
        {
            spawnPoints = goingBack.GetComponent<SpawnDirection>().targetList;
```

```
    if (other.CompareTag("Player") && canSpawn == true && SaveScript.zombiesInGameAmt < 140 - zombieSpawnAmt)
    {
        SpawnZombies();
    }
    if (other.CompareTag("Player") && canSpawn == true && SaveScript.zombiesInGameAmt >= 140 - zombieSpawnAmt)
    {
        GameObject[] zombiesToDestroy = GameObject.FindGameObjectsWithTag("zombie");
        for(int i = 0; i < zombieSpawnAmt; i++)
        {
            if (zombiesToDestroy.Length >= zombieSpawnAmt)
            {
                float furthestDistance = Vector3.Distance(transform.position, zombiesToDestroy[i].transform.position);
                if (furthestDistance > 30)
                {
                    Destroy(zombiesToDestroy[i]);
                }
            }
        }
        SpawnZombies();
    }

}

2 referências
void SpawnZombies()
{
    for (int i = 0; i < zombieSpawnAmt; i++)
    {
        if (houseSpawn == false)
        {
            int spawnRandom = Random.Range(0, spawnPoints.Length);
            Instantiate(zombies[Random.Range(0, zombies.Length)], new Vector3(spawnPoints[spawnRandom].position.x - Random.Range(0, 10), spawnPoints[spawnRandom].position.y, spawnPoints[spawnRandom].position.z - Random.Range(0, 5)), spawnPoints[spawnRandom].rotation);
        }
        else
        {
            Instantiate(zombies[Random.Range(0, zombies.Length)], spawnPoints[i].position, spawnPoints[i].rotation);
        }
        SaveScript.zombiesInGameAmt++;
    }
    canSpawn = false;
}
}
```

LoadingScreenBarSystem

This script manages the loading screen , allowing scene transitions with a loading message. It uses coroutines to delay and control scene loading, activating scenes when loading reaches 90%. The script also includes a method to exit the game and initializes settings like cursor visibility and time scale.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using UnityEngine.SceneManagement;

Script de Unity | 0 referências
public class LoadingScreenBarSystem : MonoBehaviour {

    //public GameObject bar;
    //public Text loadingText;
    //public bool backGroundImageAndLoop;
    //public float LoopTime;
    //public GameObject[] backgroundImages;
    //[Range(0,1f)]public float vignetteEfectVolue; // Must be a value between 0 and 1
    AsyncOperation async;
    //Image vignetteEfect;
    public GameObject loadingScreenText;



    0 referências
    public void ExitGame()
    {
        Application.Quit();
    }

    0 referências
    public void loadingScreen (int sceneNo)
    {
        loadingScreenText.gameObject.SetActive(true);
        StartCoroutine(WaitToLoad(sceneNo));
    }

    1 referência
    IEnumerator WaitToLoad(int sceneNo)
    {
        yield return new WaitForSeconds(1);
        StartCoroutine(Loading(sceneNo));
    }
```

```
Mensagem do Unity | 0 referências
private void Start()
{
    Time.timeScale = 1;
    Cursor.visible = true;
}

// Activate the scene
1 referência
IEnumerator Loading (int sceneNo)
{
    async = SceneManager.LoadSceneAsync(sceneNo);
    async.allowSceneActivation = false;

    // Continue until the installation is completed
    while (async.isDone == false)
    {

        if (async.progress == 0.9f)
        {
            async.allowSceneActivation = true;
        }
        yield return null;
    }
}
```

RotateBar

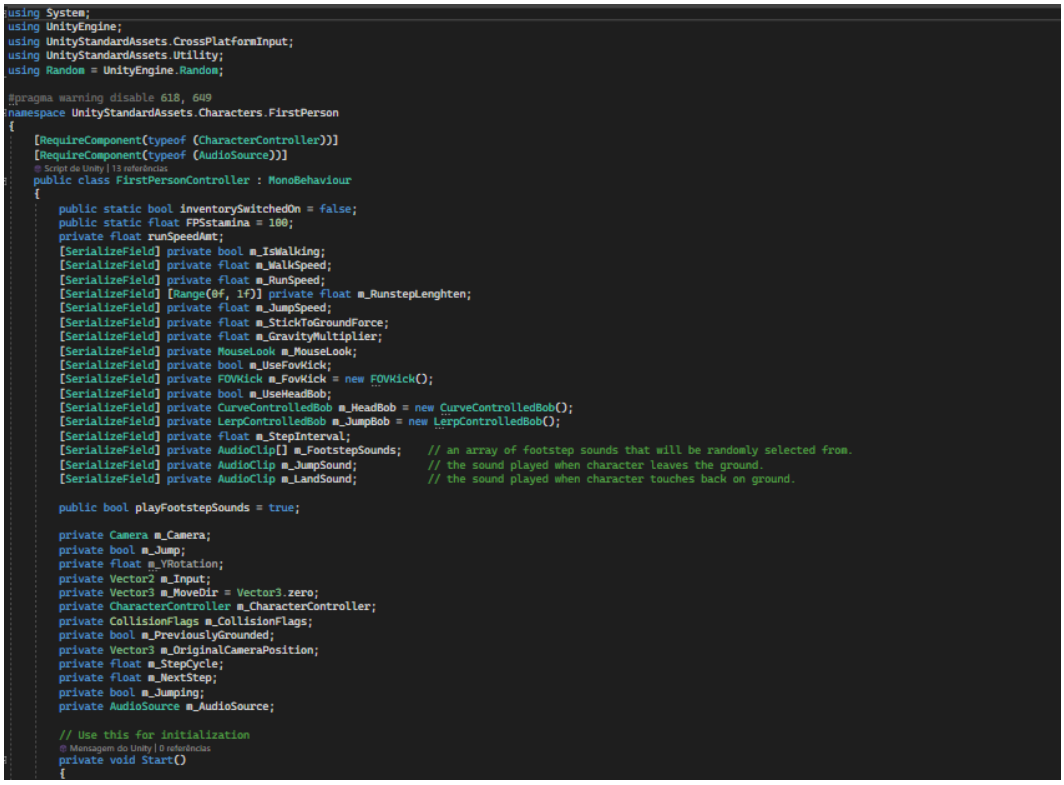
This script simply rotates the round bar of loading.

```
Script de Unity | 0 referências
public class RotateBar : MonoBehaviour {

    public float speed = 3;

    Mensagem do Unity | 0 referências
    void Update () {
        transform.Rotate (0,0,speed * Time.deltaTime); // rotation on the Z axis.
    }

}
```

FirstPersonController

This script is imported from a old standard assets from the unity store, this script has been discontinued but I managed to find it through github.

This script controlls the player movement , jump , crouch and sprint. Just needed to change the attributes.



```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
using UnityStandardAssets.Utility;
using Random = UnityEngine.Random;

#pragma warning disable 618, 649
namespace UnityStandardAssets.Characters.FirstPerson
{
    [RequireComponent(typeof (CharacterController))]
    [RequireComponent(typeof (AudioSource))]
    Script de Unity | 13 referências
    public class FirstPersonController : MonoBehaviour
    {
        public static bool inventorySwitchedOn = false;
        public static float FPSstamina = 100;
        private float runSpeedAmt;
        [SerializeField] private bool m_IsWalking;
        [SerializeField] private float m_WalkSpeed;
        [SerializeField] private float m_RunSpeed;
        [SerializeField] [Range(0f, 1f)] private float m_RunstepLenghten;
        [SerializeField] private float m_JumpSpeed;
        [SerializeField] private float m_StickToGroundForce;
        [SerializeField] private float m_GravityMultiplier;
        [SerializeField] private MouseLook m_MouseLook;
        [SerializeField] private bool m_UseFovKick;
        [SerializeField] private FOVKick m_FovKick = new FOVKick();
        [SerializeField] private bool m_UseHeadBob;
        [SerializeField] private CurveControlledBob m_HeadBob = new CurveControlledBob();
        [SerializeField] private LerpControlledBob m_JumpBob = new LerpControlledBob();
        [SerializeField] private float m_StepInterval;
        [SerializeField] private AudioClip[] m_FootstepSounds;    // an array of footstep sounds that will be randomly selected from.
        [SerializeField] private AudioClip m_JumpSound;           // the sound played when character leaves the ground.
        [SerializeField] private AudioClip m_LandSound;           // the sound played when character touches back on ground.

        public bool playFootstepSounds = true;

        private Camera m_Camera;
        private bool m_Jump;
        private float m_YRotation;
        private Vector2 m_Input;
        private Vector3 m_MoveDir = Vector3.zero;
        private CharacterController m_CharacterController;
        private CollisionFlags m_CollisionFlags;
        private bool m_PreviouslyGrounded;
        private Vector3 m_OriginalCameraPosition;
        private float m_StepCycle;
        private float m_NextStep;
        private bool m_Jumping;
        private AudioSource m_AudioSource;

        // Use this for initialization
        Mensagem do Unity | 0 referências
        private void Start()
        {
```

```
    runSpeedAmt = m_RunSpeed;
    m_CharacterController = GetComponent<CharacterController>();
    m_Camera = Camera.main;
    m_OriginalCameraPosition = m_Camera.transform.localPosition;
    m_FovKick.Setup(m_Camera);
    m_HeadBob.Setup(m_Camera, m_StepInterval);
    m_StepCycle = 0f;
    m_NextStep = m_StepCycle/2f;
    m_Jumping = false;
    m_AudioSource = GetComponent<AudioSource>();
    m_MouseLook.Init(transform , m_Camera.transform);
}


// Update is called once per frame
Mensagem do Unity | 0 referências
private void Update()
{
    if(Input.GetKeyDown(KeyCode.C))
    {
        m_CharacterController.height = 0.2f;
        m_CharacterController.radius = 0.1f;
    }
    if (Input.GetKeyUp(KeyCode.C) || FPSstamina < 20)
    {
        m_CharacterController.height = 1.8f;
        m_CharacterController.radius = 0.5f;
    }
    if (Input.GetKeyDown(KeyCode.I))
    {
        if(inventorySwitchedOn == false)
        {
            inventorySwitchedOn = true;
        }
        else if (inventorySwitchedOn == true)
        {
            inventorySwitchedOn = false;
        }
    }
    if(FPSstamina < 20)
    {
        m_RunSpeed = m_WalkSpeed;
    }
    else
    {
        m_RunSpeed = runSpeedAmt;
    }

    if (inventorySwitchedOn == false)
    {
        RotateView();
        // the jump state needs to read here to make sure it is not missed
        if (!m_Jump)
        {
            m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
        }
```

```
            if (!m_PreviouslyGrounded && m_CharacterController.isGrounded)
            {
                StartCoroutine(m_JumpBob.DoBobCycle());
                PlayLandingSound();
                m_MoveDir.y = 0f;
                m_Jumping = false;
            }
            if (!m_CharacterController.isGrounded && !m_Jumping && m_PreviouslyGrounded)
            {
                m_MoveDir.y = 0f;
            }

            m_PreviouslyGrounded = m_CharacterController.isGrounded;
        }

    }


    1 referência
    private void PlayLandingSound()
    {
        m_AudioSource.clip = m_LandSound;
        m_AudioSource.Play();
        m_NextStep = m_StepCycle + .5f;
    }


    Mensagem do Unity | 0 referências
    private void FixedUpdate()
    {
        if (inventorySwitchedOn == false)
        {
            float speed;
            GetInput(out speed);
            // always move along the camera forward as it is the direction that it being aimed at
            Vector3 desiredMove = transform.forward * m_Input.y + transform.right * m_Input.x;

            // get a normal for the surface that is being touched to move along it
            RaycastHit hitInfo;
            Physics.SphereCast(transform.position, m_CharacterController.radius, Vector3.down, out hitInfo,
                               m_CharacterController.height / 2f, Physics.AllLayers, QueryTriggerInteraction.Ignore);
            desiredMove = Vector3.ProjectOnPlane(desiredMove, hitInfo.normal).normalized;

            m_MoveDir.x = desiredMove.x * speed;
            m_MoveDir.z = desiredMove.z * speed;


            if (m_CharacterController.isGrounded)
            {
                m_MoveDir.y = -m_StickToGroundForce;

                if (m_Jump)
                {
                    m_MoveDir.y = m_JumpSpeed;
                    PlayJumpSound();
                    m_Jump = false;
```

```
                m_Jumping = true;
            }
        }
        else
        {
            m_MoveDir += Physics.gravity * m_GravityMultiplier * Time.fixedDeltaTime;
        }
        m_CollisionFlags = m_CharacterController.Move(m_MoveDir * Time.fixedDeltaTime);

        ProgressStepCycle(speed);
        UpdateCameraPosition(speed);

        m_MouseLook.UpdateCursorLock();
    }

}


1 referência
private void PlayJumpSound()
{
    m_AudioSource.clip = m_JumpSound;
    m_AudioSource.Play();
}


1 referência
private void ProgressStepCycle(float speed)
{
    if (m_CharacterController.velocity.sqrMagnitude > 0 && (m_Input.x != 0 || m_Input.y != 0))
    {
        m_StepCycle += (m_CharacterController.velocity.magnitude + (speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*
                     Time.fixedDeltaTime;
    }

    if (!(m_StepCycle > m_NextStep))
    {
        return;
    }

    m_NextStep = m_StepCycle + m_StepInterval;

    PlayFootStepAudio();
}


1 referência
private void PlayFootStepAudio()
{
    if (playFootstepSounds == true)
    {
        if (!m_CharacterController.isGrounded)
        {
            return;
        }
```

```
            int n = Random.Range(1, m_FootstepSounds.Length);
            m_AudioSource.clip = m_FootstepSounds[n];
            m_AudioSource.PlayOneShot(m_AudioSource.clip);
            // move picked sound to index 0 so it's not picked next time
            m_FootstepSounds[n] = m_FootstepSounds[0];
            m_FootstepSounds[0] = m_AudioSource.clip;
        }
    }


    1 referência
    private void UpdateCameraPosition(float speed)
    {
        Vector3 newCameraPosition;
        if (!m_UseHeadBob)
        {
            return;
        }
        if (m_CharacterController.velocity.magnitude > 0 && m_CharacterController.isGrounded)
        {
            m_Camera.transform.localPosition =
                m_HeadBob.DoHeadBob(m_CharacterController.velocity.magnitude +
                                    (speed*(m_IsWalking ? 1f : m_RunstepLenghten)));
            newCameraPosition = m_Camera.transform.localPosition;
            newCameraPosition.y = m_Camera.transform.localPosition.y - m_JumpBob.Offset();
        }
        else
        {
            newCameraPosition = m_Camera.transform.localPosition;
            newCameraPosition.y = m_OriginalCameraPosition.y - m_JumpBob.Offset();
        }
        m_Camera.transform.localPosition = newCameraPosition;
    }


    1 referência
    private void GetInput(out float speed)
    {
        // Read input
        float horizontal = CrossPlatformInputManager.GetAxis("Horizontal");
        float vertical = CrossPlatformInputManager.GetAxis("Vertical");

        bool waswalking = m_IsWalking;

#if !MOBILE_INPUT
        // On standalone builds, walk/run speed is modified by a key press.
        // keep track of whether or not the character is walking or running
        m_IsWalking = !Input.GetKey(KeyCode.LeftShift);
#endif
        // set the desired speed to be walking or running
        speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed;
        m_Input = new Vector2(horizontal, vertical);

        // normalize input if it exceeds 1 in combined length:
        if (m_Input.sqrMagnitude > 1)
        {
```

```
            m_Input.Normalize();
        }

        // handle speed change to give an fov kick
        // only if the player is going to a run, is running and the fovkick is to be used
        if (m_IsWalking != waswalking && m_UseFovKick && m_CharacterController.velocity.sqrMagnitude > 0)
        {
            StopAllCoroutines();
            StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown());
        }
    }


    1 referência
    private void RotateView()
    {
        m_MouseLook.LookRotation (transform, m_Camera.transform);
    }


    Mensagem do Unity | 0 referências
    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        Rigidbody body = hit.collider.attachedRigidbody;
        //dont move the rigidbody if the character is on top of it
        if (m_CollisionFlags == CollisionFlags.Below)
        {
            return;
        }

        if (body == null || body.isKinematic)
        {
            return;
        }
        body.AddForceAtPosition(m_CharacterController.velocity*0.1f, hit.point, ForceMode.Impulse);
    }
}
}
```

CONSIDERATIONS

Considering the time I had and the scale of the project I aimed to complete, I believe my project ended well, and I managed to create the game as intended. However, I think there are some important elements missing from this game, such as a mission guide for the user, perhaps a cutscene to explain the game or a big boss to make the game a little bit more interesting.

Task	Percentage Done	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9
		15/04/2024	22/04/2024	29/04/2024	06/05/2024	13/05/2024	20/05/2024	27/05/2024	03/06/2024	10/06/2024
Technical Design Document	100%									
Game Design Document	100%									
Create Envioronment	80%									
Create Characters	40%									
Create Objects & Weapons	60%									
Create Triggers & Events	40%									
Code Development	40%									
Menu implementation	10%									
Audio Implementation	10%									
Testing	20%									
Implement last details & polish	0%									
Evaluation	-									
Create Project Report	30%									

FONTS

Letter fonts

https://www.1001fonts.com/dosis-font.html

https://www.1001fonts.com/requiem-font.html

https://www.1001fonts.com/glacial-indifference-font.html

Pickups

Pills:

https://sketchfab.com/3d-models/pills-429536ff8dd84e538168c17ce6e82e28

First aid kit:

https://sketchfab.com/3d-models/tactical-first-aid-kit-011c33c121284bc88bb765a85511dae1

keys:

https://sketchfab.com/3d-models/key-ad78fc71092849ca9cd2f264e14a8167

https://sketchfab.com/3d-models/key4-96517c1c7be642c78ee4645f0fd741ff

Jerry can:

https://sketchfab.com/3d-models/jerrycan-a2c502a2eb60487e9faf4a87e19b57f3

shotgun ammo:

https://sketchfab.com/3d-models/shotgun-ammo-box-7d4eeb8ce38648f285cad5626c1b7e3a

gun ammo:

https://sketchfab.com/3d-models/38-special-ammo-1cbec5a6b0d845ffa09abc0356a8fd5a

Cloths:

https://sketchfab.com/3d-models/frosted-sheets-3e3fb2880a184a418dfa62581d07e12d

locker cabinet:

https://sketchfab.com/3d-models/metal-cabinet-bc3025eba4af431fa0785cdfc0e56ed9

wooden cabinet:

https://sketchfab.com/3d-models/wooden-cabinet-with-doors-0ca10f4929a04a5b894d445caa7920c1

lab cabinet:

https://sketchfab.com/3d-models/laboratory-cabinet-storage-pbr-low-poly-free-6c5d3bde09dd4f55ad973d8429441190

trash can:

https://sketchfab.com/3d-models/trash-can-3836ff67877248c2a820e6a969984aac

Map

https://assetstore.unity.com/packages/3d/environments/flooded-grounds-48529

Shaders

https://assetstore.unity.com/packages/vfx/shaders/free-double-sided-shaders-23087

Weapons

bottle:

https://sketchfab.com/3d-models/bottle-d2c6dbdfac514b01aac842d3f89a08e9

baseball bat:

https://sketchfab.com/3d-models/survival-melee-weapon-4-1de2d38ecb2a4883bd5de635bf9ea614

Cleaver:

https://sketchfab.com/3d-models/cleaver-butchers-best-friend-8b3a394b1e6741e187e15b8dc5075b54

spray can:

https://sketchfab.com/3d-models/realistic-spray-paint-can-761fd13227bd411a9a10d9ecae0e6b8d

Survival kit lite:

https://assetstore.unity.com/packages/3d/props/tools/survival-kit-lite-92549

Horror Axe:

https://assetstore.unity.com/packages/3d/props/tools/horror-axe-107507

Throwing

https://sketchfab.com/3d-models/arms-throwing-1f9a5c717aae4d0f9b77232b7da4b875

shotgun

https://sketchfab.com/3d-models/shotgun-animated-c3d3cf425869463a84d650c15e3af0d2

Knife

https://sketchfab.com/3d-models/knife-animated-e5db6b73878f4123baa48cea6d9af84c

Gun

https://sketchfab.com/3d-models/pistol-animated-cbe1e14b09094a09b8b59deca84e2922

Sound Effects

https://assetstore.unity.com/packages/audio/sound-fx/free-deadly-kombat-228835

https://assetstore.unity.com/packages/audio/sound-fx/horror-elements-112021

Other sounds through

https://pixabay.com , https://www.storyblocks.com/ and https://youtube.com

Particles

https://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-5-x-73777

AutoLOD

https://github.com/Unity-Technologies/AutoLOD

Scripts

https://stackoverflow.com/questions/tagged/unity-game-engine?tab=Newest

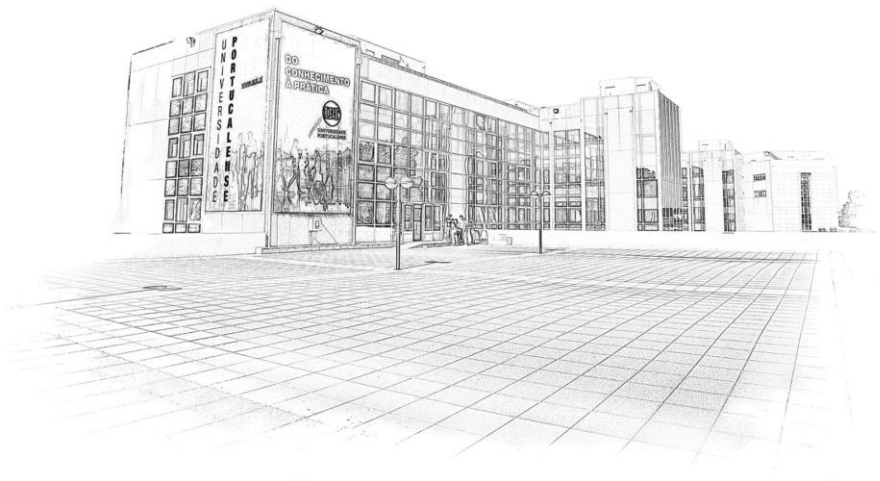
https://www.reddit.com/r/Unity3D/

https://youtube.com

Standard Assets

https://github.com/Unity-Technologies/Standard-Assets-Characters

}



Universidade Portucalense Infante D. Henrique | Rua Dr. António Bernardino de Almeida, 541
4200-072 Porto | Telefone: +351 225 572 000 | email: upt@upt.pt