P3D Report 3D Unity Game

Ethan Chan

Candidate No. 246667

Table of Contents

1. Introduction

This section contains the backstory of the game.

1.1. Backstory

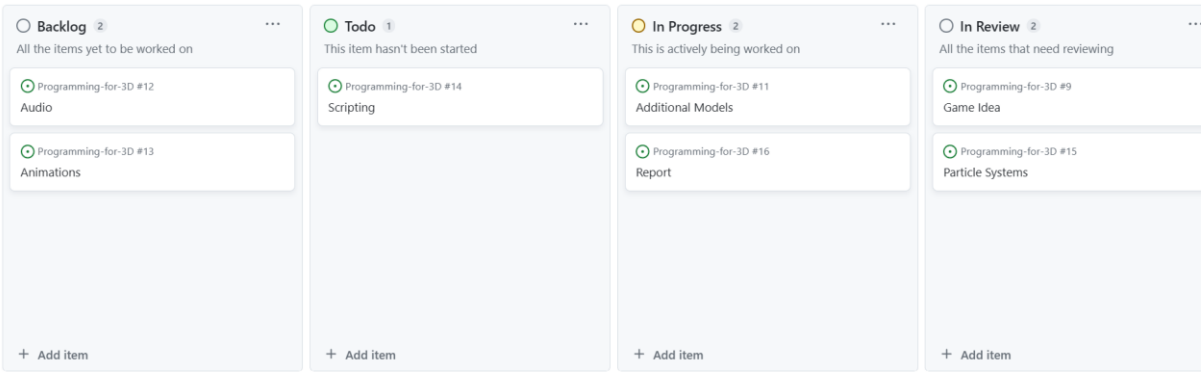
Leonard (the player) starts off in a cabin overlooking a computer which contains a series of emails instructing him on what to do. Upon looking outside, a monster is lurking and observing in the cold, wintery moonlight. However, it has not breached the cabin yet. Leonard is instructed to complete a ritual using items scattered around the house with a riddle as his only guide as to what these are. Only by completing the ritual will he receive the weapon which can take down the monster, however time is against him as the mysterious cabin slowly deteriorates his sanity, thirst, and energy. Perhaps this is why the monster stays outside.

2. Project Planning

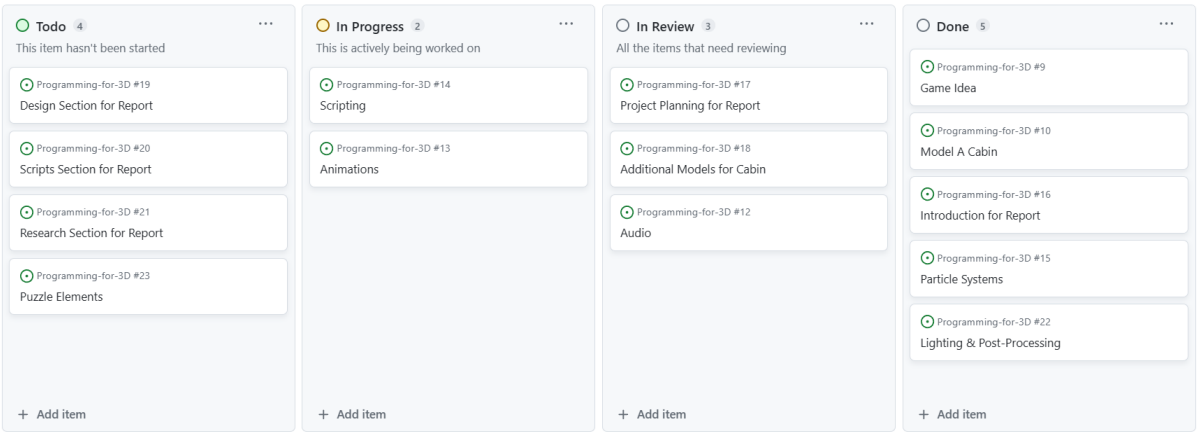
This section contains the planning done throughout development of this project via the use of GitHub Kanban boards and Microsoft Sticky Notes. These applications aided greatly to monitor what tasks to complete and aided in time management for this project as weekly goals were been set.

2.1. Kanban Board

The Kanban board was used early on in development as well as midway through to plan and guide the development process. As the project progressed, more items were added on to this board. Using the Kanban board helped massively as it served as a reminder for what tasks had yet to be done, as well as what I needed to finish that week. It was also extremely simple to add new items where needed as well as to make comments whenever a task needed more explanation.



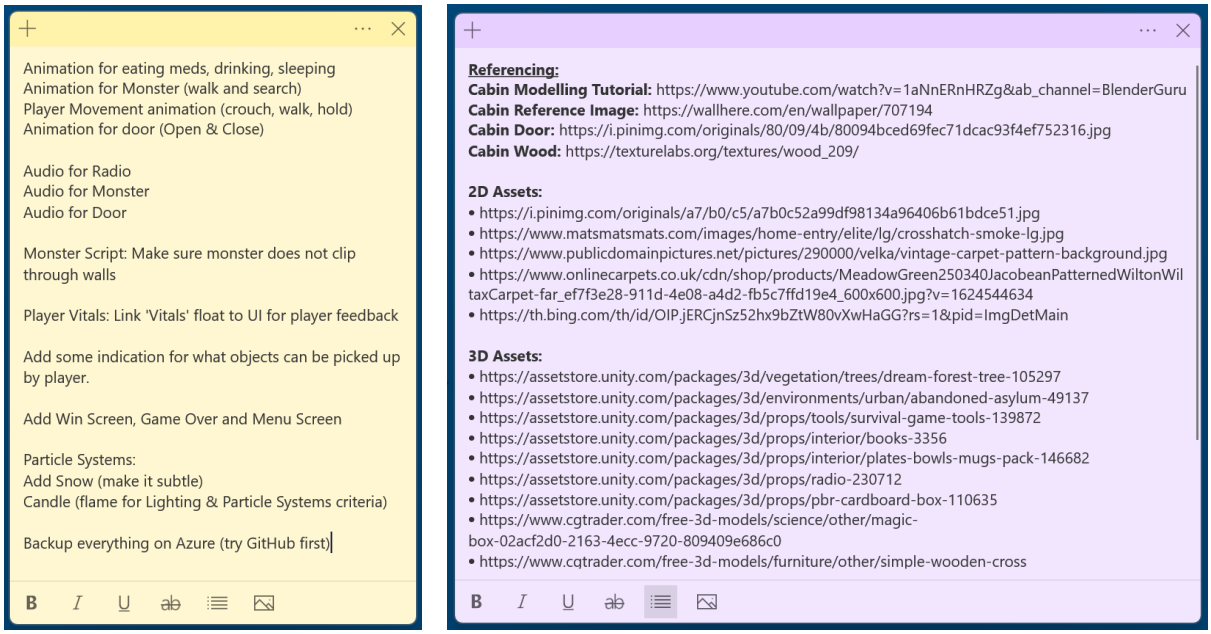
2.1 The Kanban board within the first weeks of development



2.1.2 The Kanban board midway through development

2.2. Microsoft Sticky Notes

The use of the Windows Sticky Notes app was utilised alongside the Kanban board and towards the end of the development. It allowed myself to track what there was left to do all on one monitor. This helped track what had to be done by the end of each week. I had also created a dedicated sticky note to contain all the references of assets used which could be added to this report in the 'References' chapter.



2.2 Notes on what tasks to do (Left) and references noted (Right).

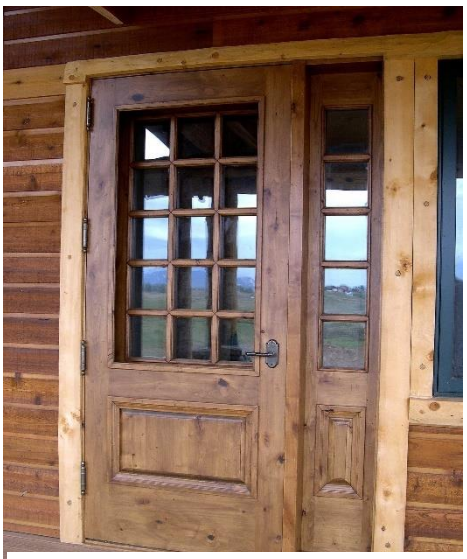
3. Design

This section contains design choices made to develop the game, and explained how it goes beyond the concepts taught in the Labs from week two to five. The design process for the cabin is also shown alongside software used for this project.

3.1. The Cabin Exterior

The cabin was created in Blender[1] following a tutorial from BlenderGuru. I used the reference image (shown by figure 3.1.1) rather than to completely follow the YouTube tutorial. This allowed me to create my own unique house which fit the criteria I needed for an eerie scene. As the reference image had no door, I used figure 3.1.2 as the main door for the game. I chose this door as it provides a good view to the outside allowing a line of sight to the monster which creates a further sense of unease.

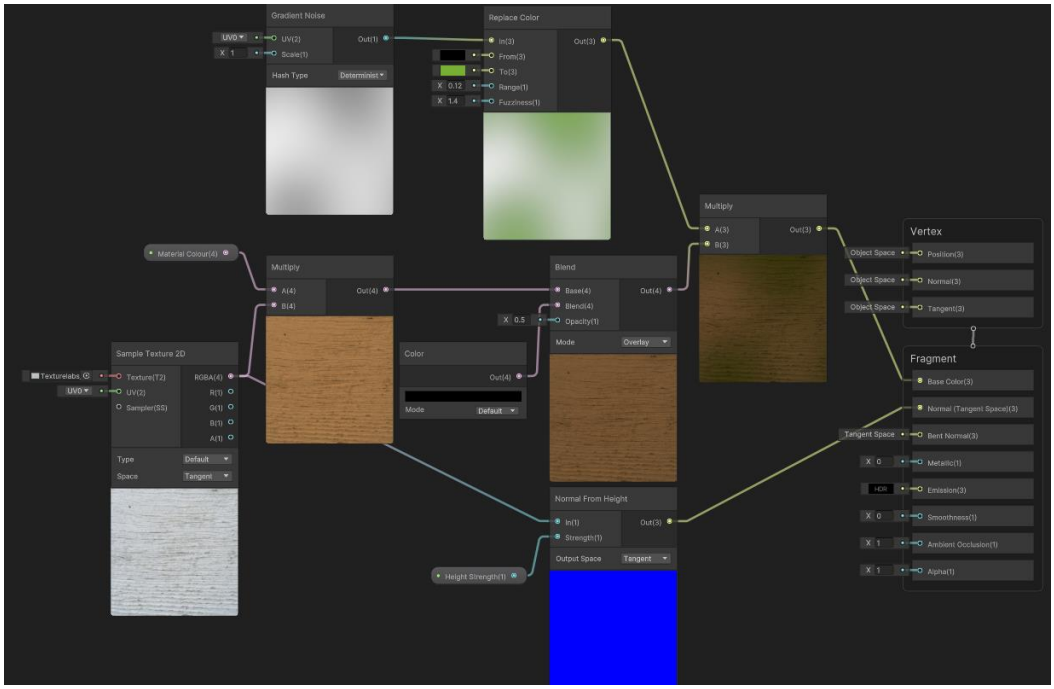
3.1.1 Reference image used to model the cabin.



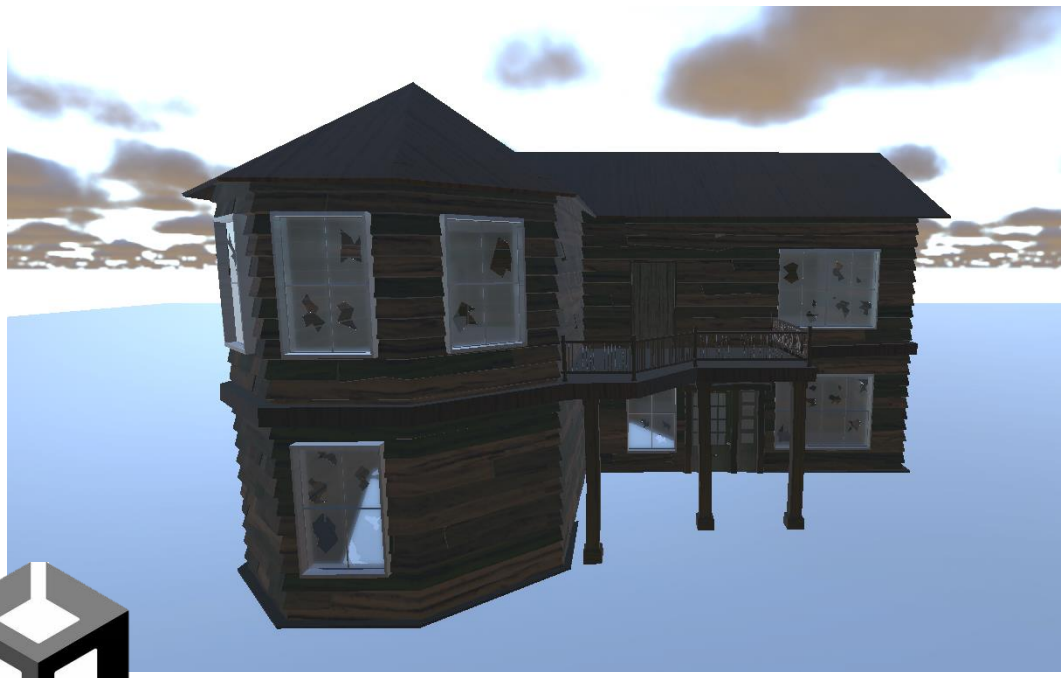
3.1.2 Reference image for the cabins main front door.

3.1.3 Cabin for the game (modelled in Blender).

The Blender model was imported into Unity[2] and given materials created using Shadergraphs (Figure 3.1.4). The original wood texture was sampled from texture labs with a subtle green hue added to create the illusion of mould in the cabin indicating that it's been abandoned for quite some time. The gradient noise also allows for more contrast in the wood so that the texture does not have a repetitive pattern on the cabin walls.



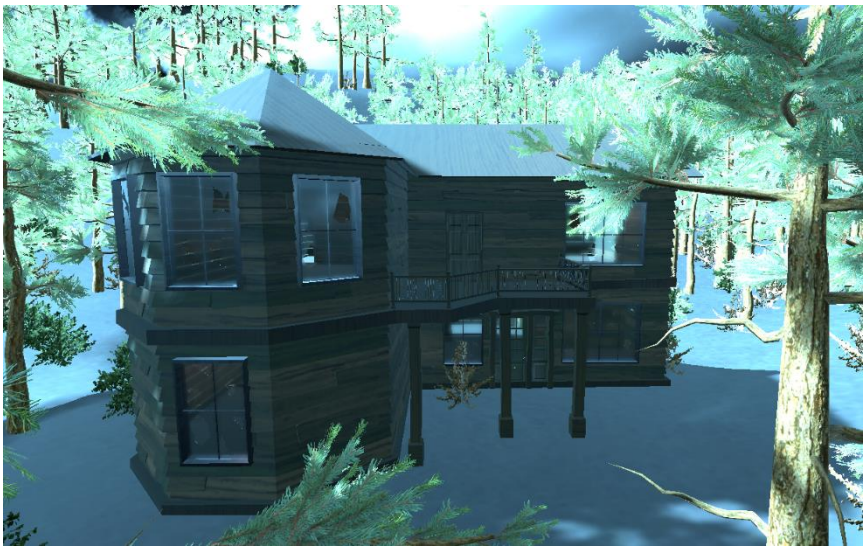
3.1.4 Wood material created using Unity's built-in shader graph.



3.1.4 Fully textured cabin imported into Unity.

3.2. Environment and Lighting

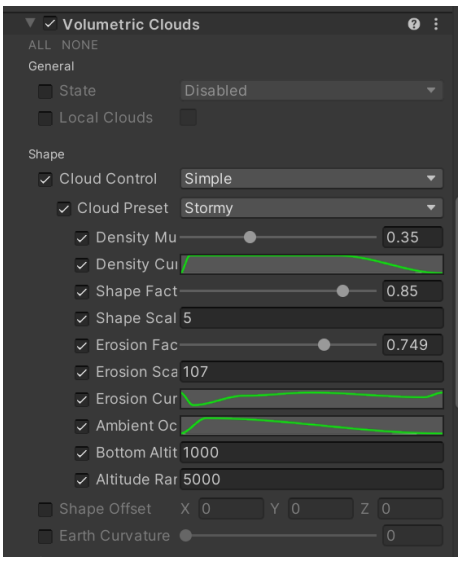
The environment was created using Unity's built-in terrain system. This aided in creating hills and uneven terrain so the ground would be more realistic. An asset pack was used to create the surrounding forest and bushes. Using The terrain system meant that 3D trees could be placed easily and are only drawn onto the screen when the player is within a certain threshold. This resolved some lag issues at the time as it meant less trees could be rendered in the game.



3.2.1 Terrain and foliage added to the scene.

As the project was made in HDRP, the project could have a more realistic environment. Fog and Post-Processing was used to limit the players vision and create a more immersive atmosphere of a cold wintery night. The dense volumetric clouds and directional light created rays of light from the moon. This simulated the light peeking through the trees during the night. The directional light was also set to have a blue hue. This caused the scene to have a cooler tone which contrasted with the warm cabin interior imitating a safer environment.

3.2.2 Rays of light shining through the clouds and trees in the scene.



3.2.3 Volumetric Clouds adjusted its highest preset.

3.3. The Cabin Interior

After adding materials to the cabin and foliage to the environment, the interiors had to be decorated. This went through multiple iterations before I used the Abandoned Asylum asset pack. One large asset pack was used to maintain consistency between art styles. I chose the abandoned asylum in particular as it had all the furnishing for multiple different kinds of rooms found in a house. Rooms were allocated specifically so the player would constantly have to traverse the house looking for items. This reinforced the puzzle elements, rewarding the player for being more observant for each rooms contents.

Listed below are noteworthy items in each room:

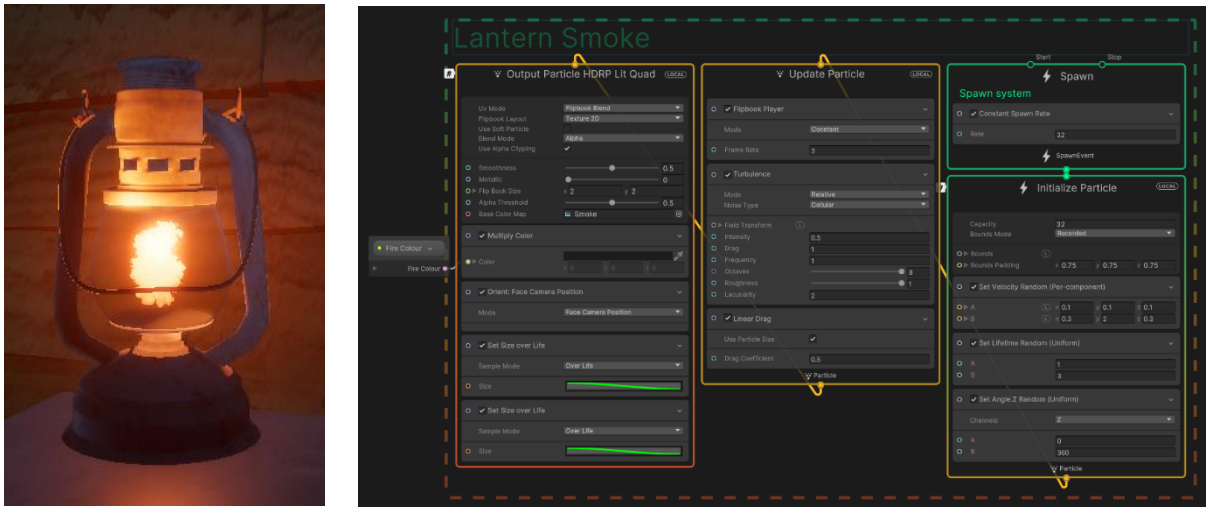
- **Living Room:** Contains the Vitals TV (to monitor time before death), Matchbox (Exclusively found in this room), Radio for ambient music, Monster TV to see where the monster is, and player spawn (in front of the computer providing gameplay instructions)
- **Main Entrance/ Foyer:** Links to all other rooms in the house, and leads to the main door where the monster can be seen outside.
- **Kitchen:** Contains the kitchen sink to replenish the players thirst.
- **Bathroom:** Contains the medicine shelf to replenish the players sanity, contains the Med Kit (Exclusively found in this room), and the dead monster who points at the door (providing the answer for the final puzzle).
- **Upstairs Lounge:** Contains the Satanic Ring which the player frequents to deposit items, also contains the riddle on what items may be sacrificed.
- **Balcony:** Provides a safe view of the monster
- **Bedroom:** Contains the bed to replenish the players energy, and a second radio to provide ambient music on the upper floor.

3.3 Rooms of the cabin with furnishing and lighting.

3.4. Particle Systems

3.4.1. Lantern Smoke

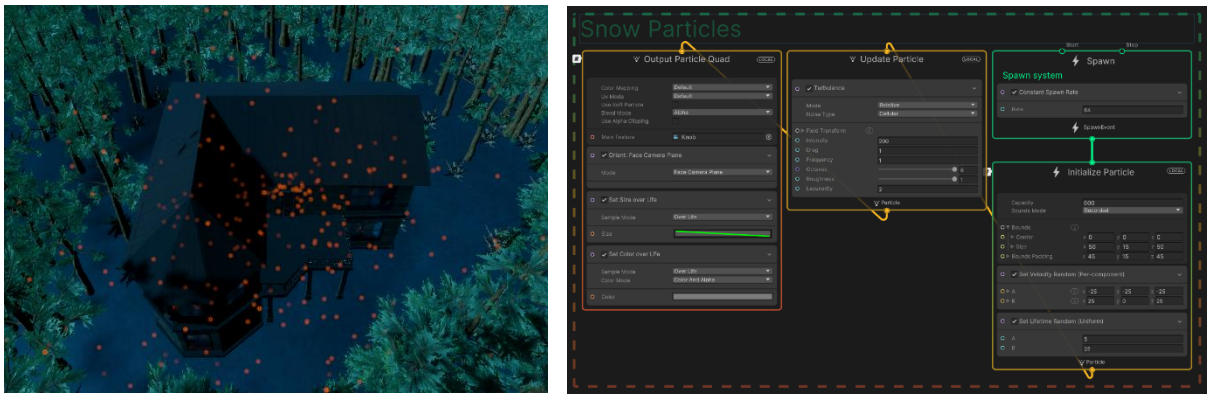
The smoke of the lantern was created using a VFX graph. It uses the same smoke flipbook from the coursework, however it has been scaled down, with a shortened lifetime and given an orange hue. This simulated a gas lantern which is used throughout the scene. There is also a point light slightly below the smoke to create a glow. This is used in conjunction with the fog post-processing which gives off a faint warm hue.



3.4.1 A lantern created with Unity's VFX graph and a point light.

3.4.2. Snow Particles

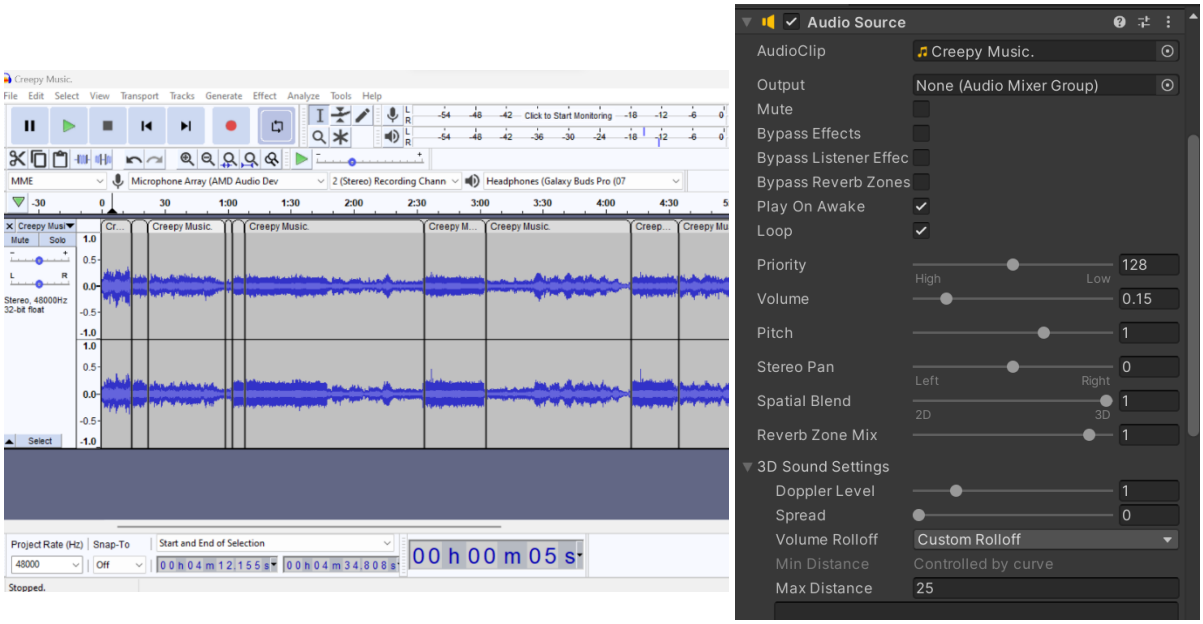
Snow Particles were added to the scene early on in development alongside the terrain. This reinforced the cold winter night setting and simultaneously added a bit of ambience to the world. The addition of snow particles in particular interested me as it goes beyond what has been taught in the labs, which has led to me becoming more comfortable with Unity's new particle system.



3.4.2 Snowfall created with Unity's VFX graph.

3.5. Audio

In the game, there are many audio sources present, such as from the monster, the radio, player footsteps, et cetera. These were sourced from YouTube, Freesound[3], and from recordings on my phone. These were then imported into Audacity[4] where I could edit the tempo and splice audio clips. As shown by figure 3.6.1, the music sourced from YouTube was merged together with noise added in Audacity. This was done to create an unpleasant music track on the radio, as if it were malfunctioning.



3.5.1 Audacity (shown left) and Unity's Audio Source (shown right).

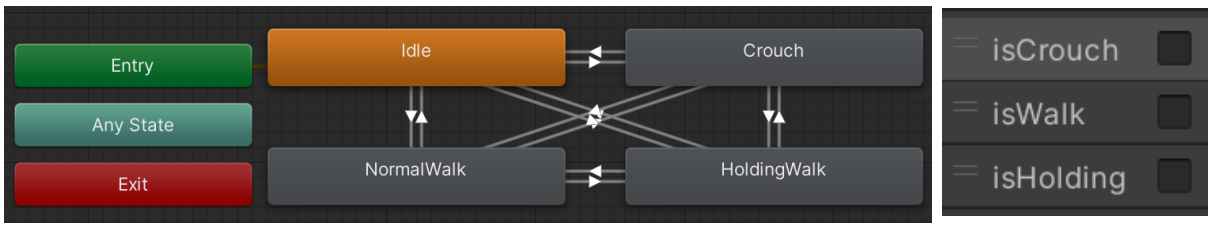
Some audio clips such as mouse clicks for the computers and doors opening had pitch variations which were determined randomly by code. This created non-repetitive audio for tasks that are repeated multiple times in a single instance as it would be expected for a player to click through the UI pages, as well as constantly walk between rooms.

```
audioClip.Play();
audioClip.pitch = Random.Range(1f, 2f);
```

3.5.2 Code snippet which adjusts the pitch to simulate slight variations in audio.

3.6. Animations

Player and Monster animations were downloaded from Mixamo[5]. This allowed for dynamic animations for humanoid creatures (shown in figure 3.7.1) such as walking, looking, and crouching. The game also features other animations including user interfaces and doors. Doors had to have an open and close animation due to the criteria's of the coursework. This was done by an OnTriggerEnter event which played the animation accompanying the audio. As the door models pivot was in the centre of the object, each door had a parent Gameobject which repositioned the doors mesh so it would rotate correctly.



3.6.1 Animator Graph showing the links between animations for the player. These are played depending on Boolean parameters changed in code.

3.6.2 Player Model downloaded with animations from Mixamo.

3.7. Physics (Interactable Objects)

Every interactable object has a box collider and a rigid body attached. The mass of the rigid body is a rough estimate determined by the size and density of the prop. The use of a box collider means that a Raycast can be shot from the camera. A box collider has been chosen for these objects as using a more complex mesh would become increasingly computationally expensive with multiple GameObjects. However, this has led to some complications with collisions due to the inaccuracy on some objects. The item pickup script is explained in-depth in the 'Scripts' chapter.

3.7.1 Objects from afar do not glow (left).

3.7.2 Approaching the object creates a blue glow (right)

4. Scripts

Overall thirteen C# scripts were created during the development of this coursework. However, due to the word limitations of this report, this chapter contains only two code snippets. I have chosen these two snippets as it explains player interactions and how the puzzle element works in the game.

4.1. playerPickup.cs

The player pickup script checks whenever the player clicks their mouse every frame. This then calls the grabItem function which detects if the player is already holding an item. If an item is already held, the player will drop that item, otherwise a Raycast will be shot from the camera and a valid object will be picked up if in reach. The player can also rotate the object with 'E' and 'Q' as well as reset its rotation to 0,0,0 with a right mouse click. This script is attached to the player with variables such as the player hand and camera.

```
private void Update()
{
    if (Input.GetMouseButtonDown(0)) { grabItem(); } //detects when the player has clicked (to try and pick up an item)
    if (hand.transform.childCount >= 1) { rotateItem(); } //true when player is holding an item
}

1 reference
void grabItem()
{
    //If the player is already holding an item, drop that item
    if (hand.transform.childCount >= 1)
    {
        hand.transform.GetChild(0).GetComponent<Rigidbody>().isKinematic = false;
        hand.transform.GetChild(0).gameObject.layer = 10;
        hand.transform.GetChild(0).parent = null;
    }
    //Otherwise, shoot a raycast and if it hits an object that can be picked up, set its parent to the player
    else
    {
        Ray ray = cam.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit, 10, mask))
        {
            hit.rigidbody.isKinematic = true;

            hit.transform.SetParent(hand.transform);
            hit.transform.position = hand.transform.position;
            hit.transform.gameObject.layer = 11;
        }
    }
}

//Called when player is holding an item
1 reference
void rotateItem()
{
    if (Input.GetKey(KeyCode.Q)) { hand.transform.localRotation *= Quaternion.Euler(0, rotAmount, 0); }
    if (Input.GetKey(KeyCode.E)) { hand.transform.localRotation *= Quaternion.Euler(0, -rotAmount, 0); }
    if (Input.GetMouseButtonDown(1)) { hand.transform.GetChild(0).rotation = Quaternion.identity; }
}
```

4.1 The playerPickup.cs script

4.2. investigationTable.cs and puzzleBox.cs

Both the investigation table and puzzle box script function in similar ways as the UI's both have a forward and back button. These buttons tell the script to increment to the next game object. This is done by deactivating all other gameobjects, and then enabling the next item in the index. In order to not get an 'out of range' error when iterating the index of the array, we take the modulo division to make sure the index always falls within the number of created gameobjects. The script also takes the absolute value of page numbers to avoid negative values when the back button is pressed.

```
//Called by button interaction. Increments to the next item in pages array
0 references
public void nextPage()
{
    index++;
    foreach (GameObject page in pages)
    {
        page.SetActive(false);
    }
    pages[Mathf.Abs(index) % pages.Length].SetActive(true);
    audioClip.Play();
    audioClip.pitch = Random.Range(1f, 2f);
}

//Called by button interaction. Decrements to the previous item in pages array
0 references
public void prevPage()
{
    index--;
    foreach (GameObject page in pages)
    {
        page.SetActive(false);
    }
    pages[Mathf.Abs(index) % pages.Length].SetActive(true);
    audioClip.Play();
    audioClip.pitch = Random.Range(1f, 2f);
}
```

4.2.1 The investigationTable.cs script (the puzzleBox.cs script functions similarly).

The puzzle box script has an additional line of code which determines whether the player has chosen the correct answer. This is done by assigning the 'answer' variable with a gameobject in the inspector. The coroutine runs every time a page has been changed however can only be true once as the Boolean will always return false after its first iteration. This stops multiple instances of weapons spawning into the scene.

```
//If the answer is not changed for 1 second, coroutine checks to see if the current option is the answer
//It does not do this if the correct answer has already been chosen once
2 references
IEnumerator checkAnswer()
{
    yield return new WaitForSeconds(1);
    if (options[Mathf.Abs(index) % options.Length] == answer && !unlocked)
    {
        GameObject spawnWeapon = Instantiate(weapon, transform.position + (Vector3.up * 5f), Quaternion.identity);
        unlocked = true;
        correctAnswer.Play();
    }
}
```

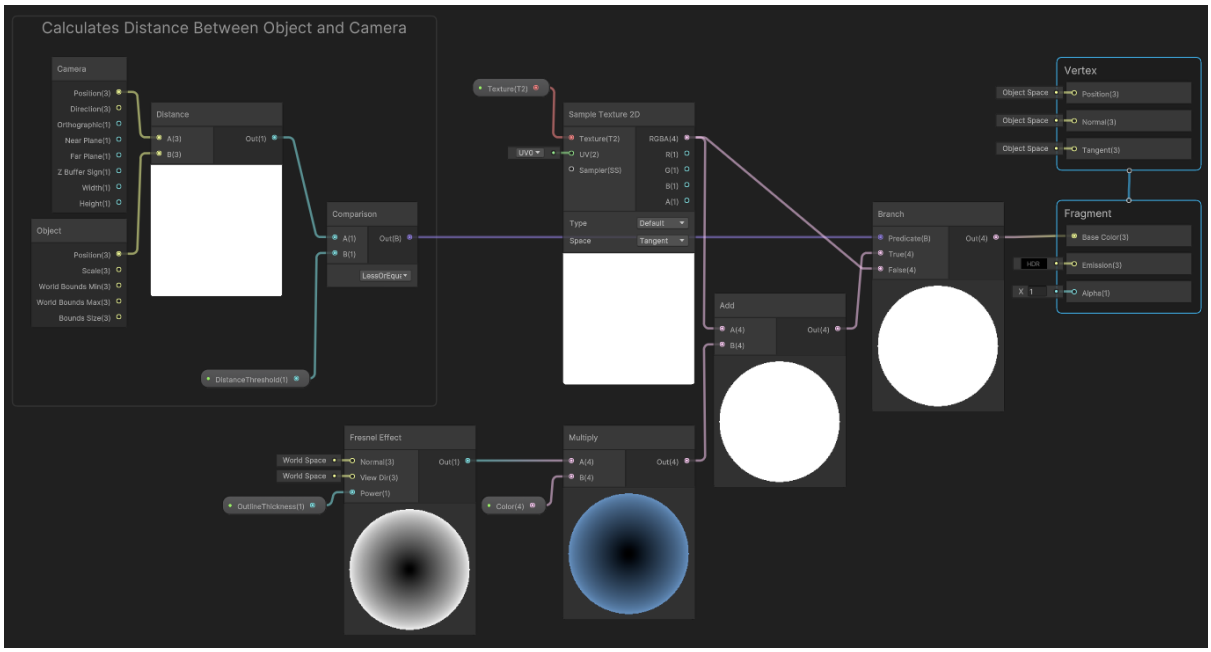
4.2.2 Additional code from the puzzleBox.cs script.

5. Research

This section delves into some research done to fix issues found whilst developing the game.

5.1. Fresnel effect to produce a glow for interactable objects

The majority of props in the scene can be picked up. These items glow a faint blue when the player approaches. This was accomplished using Shadergraphs. The distance is calculated between the object (with the material attached), and the player camera. If the camera is within a certain distance, a Fresnel effect overlays on top of the texture of the material. Otherwise, it keeps it's default texture without the effect. This shadergraph is used for all interactable objects that can be picked up which makes it clearer to the player what is part of the scenery and what is an interactable.



5.1 A shadergraph used for all interactable objects that can be picked up.

Utilising the Fresnel effect was inspired by a Brackey's tutorial on YouTube[6], at the time I was looking for a way to produce an outline for objects however I discovered that it would be computationally expensive as it would require rendering an additional texture scaled slightly larger behind the main object. The Fresnel effect does not have this issue as the effect is multiplied on top of the existing texture. This meant that the CPU no longer had to render double the amount for each object with this shadergraph.

5.2. Interactions with user interfaces set to World Space

Originally, approaching the tutorial table would activate a UI panel overlaying the player's screen. This felt intrusive and unclear that a tutorial is in that location as it was in the corner of the room. Instead the player starts off at the table and a canvas was created in World Space which would be placed on the computers monitor. This gave the player the chance to read the tutorial whilst still immersed in the cabin.

This decision had unforeseen circumstances as Unity prevents a locked cursor from interacting with UI buttons in world space. This meant that I could not use a first person perspective to click buttons. Despite this, a Unity forum had resolved this issue prior by overriding the GraphicRaycaster[7]. This allowed for all button clicks to be centred around the player's crosshair which seemed to work initially but created more issues later on in development as it meant that I could no longer have UI in screen space which became troublesome with the death and win UI screen.

This issue was fixed by adding an conditional statement which checks if the monster is still in the level. This worked well as the monster gameobject gets destroyed on death or win which meant the game knew when to switch between World Space interactions and Screen Space interactions.

```
Unity Script (5 asset references) | 0 references
public class RaycasterWorld : GraphicRaycaster
{
    public GameObject monster;

    0 references
    public override void Raycast(PointerEventData eventData, List<RaycastResult> resultAppendList)
    {
        //Set eventdata to the middle of the screen if the monster is alive, otherwise set it to the mouse position
        if (monster != null) { eventData.position = new(Screen.width / 2, Screen.height / 2); }
        else { eventData.position = Input.mousePosition; }

        base.Raycast(eventData, resultAppendList);
    }
}
```

5.2 Code provided by QpaCompany which has been modified to fit the criteria of the game.

6. Summary

I believe 'The Person, In the Cabin, In the Woods' was a success due to planning using the Kanban board and GitHub alongside feedback provided by Professor Huckle on a weekly basis early on in the module. This module has enabled me to delve further in games development with a piece of work I would be satisfied to add to my portfolio.

In making this game, I have learnt to use ProBuilder as well as improve my skills in Blender. I am also now more comfortable with using Unity's ShaderGraph which I was not aware of its usefulness prior to development.

Despite this, I believe there are some faults with the game which could be improved upon:

- Whilst playtesting, I realised that the framerate of my game averaged 15fps on my laptop. Ideally I would like the game to be optimised furthermore to resolve this issue.
- Assets could have varied furthermore as the majority of props can be noticeably similar (such as books on the shelf).
- The snow particle system could be more realistic. Currently it is possible for particles to clip through the cabin.
- The monster AI could be more aggressive when it sees the player, this was not done as it would require frequent Raycasts to detect if the player is in its line of sight which would furthermore contribute to the lag.

7. Future development

Listed below are some features I would like to add to the game:

- Overall optimisation to resolve potential issues with framerate.
- A more intelligent monster AI.
- Improvement to certain animations (e.g. Player Movement).
- Activities outside the cabin.
- Allowing the monster to enter the cabin, where the player must hide to not be caught.

I believe these features would create a more intense environment as well as improve general quality of life in gameplay.

8. Appendices & References

8.1. GitHub Repository

https://github.com/Ethano-ChaChano/Programming-for-3D

```
Uploading LFS objects:   0% (0/1), 0 B | 0 B/s, done.
batch response: This repository is over its data quota. Account responsible for LFS b
andwidth should purchase more data packs to restore access.
error: failed to push some refs to 'https://github.com/Ethano-ChaChano/Programming-fo
r-3D.git'
```

Due to files becoming too large to GitHub, including using GitHub LFS, I have uploaded the Assets folder to the University's OneDrive.

OneDrive Link: https://universityofsussex-my.sharepoint.com/:f:/r/personal/ec603_sussex_ac_uk/Documents/ProgrammingFor3D?csf=1&web=1&e=NmeYr0

8.2. Scripts

https://github.com/Ethano-ChaChano/Programming-for-3D/tree/main/src/ProgrammingFor3D/Scripts

playerDeath.cs

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

public class playerDeath : MonoBehaviour
{
    public Animator anim;
    AudioSource audioClip;
    playerMovement movement;
    //public GameObject playerCam;
    public TMP_Text deathText;
    public GameObject monster;

    bool isDead;

    private void Start()
    {
        movement = FindObjectOfType<playerMovement>();
        audioClip = GetComponent<AudioSource>();
    }

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape)) { killPlayer("Pausing to take a break is a definite way to get killed...", 0); }
    }



    //Calls the coroutine to kill the player if they are not yet dead
    public void killPlayer(string deathText, float timeTillDeath)
    {
        if (!isDead && monster != null)
        {
            StartCoroutine(PlayerDeath(deathText, timeTillDeath));
        }
    }

    //Display the death UI, jumpscare, and audio. Set the player to dead
    IEnumerator PlayerDeath(string text, float timeTillDeath)
    {
        yield return new WaitForSeconds(timeTillDeath);

        deathText.text = text.ToUpper();
        movement.lookSensitivity = 0;
        movement.speed = 0;

        Destroy(monster);
        anim.SetBool("isDead", true);
        audioClip.Play();

        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
        isDead = true;
    }
}
```

playerMovement.cs:

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

[RequireComponent(typeof(CharacterController))]
public class playerMovement : MonoBehaviour
{
    [Header("Movement Settings")]
    public int speed;
    public int normalSpeed = 5;
    public int sneakSpeed = 3;

    public float gravity = 0.5f;

    [Header("Look Settings")]
    public float lookSensitivity = 5;
    float lookX;
    float lookY;

    public Animator anim;
    public AudioSource clip;
    public Transform hand;
    public GameObject monster;
    CharacterController controller;
    Camera cam;
    Vector3 position = Vector3.zero;

    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;

        controller = GetComponent<CharacterController>();
        cam = GetComponentInChildren<Camera>();
    }

    void Update()
    {
        if (monster != null)
        {
            movement();
            sneak();
            look();
            if (!controller.isGrounded) { position.y -= gravity * Time.deltaTime; } //gravity

            position = transform.TransformDirection(position);
            controller.Move(position * Time.deltaTime);
        }
    }

    void movement() {
        Vector2 movement = new Vector2(Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"));
        position.x = movement.x * speed;   //affects players x-axis
        position.z = movement.y * speed;   //affects players z-axis

        //If Moving (and not holding or holding)
        if ((Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0) && !(hand.childCount > 0)) { anim.SetBool("isWalk", true); if (!clip.isPlaying) { clip.Play(); } }
        else if ((Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0) && hand.childCount > 0) { anim.SetBool("isHolding", true); if (!clip.isPlaying) { clip.Play(); } }
        else { anim.SetBool("isWalk", false); anim.SetBool("isHolding", false); clip.Stop(); }
    }

    void sneak()
    {
        if (Input.GetKey(KeyCode.LeftShift) && (Input.GetAxis("Horizontal") == 0 && Input.GetAxis("Vertical") == 0)) { controller.height = 1; speed = sneakSpeed; anim.SetBool("isCrouch", true); }
        else { controller.height = 2; speed = normalSpeed; anim.SetBool("isCrouch", false); }
    }

    void look() {
        lookX += Input.GetAxis("Mouse X") * lookSensitivity;
        lookY += Input.GetAxis("Mouse Y") * lookSensitivity;

        lookY = Mathf.Clamp(lookY, -60, 90);

        cam.transform.localRotation = Quaternion.Euler(-lookY, 0, 0);
        transform.rotation = Quaternion.Euler(0, lookX, 0);
    }
}
```

playerPickup.cs

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

public class playerPickup : MonoBehaviour
{
    public int rotAmount = 5;
    public Camera cam;
    public LayerMask mask;
    public GameObject hand;

    private void Start()
    {
        cam = GetComponentInChildren<Camera>();
    }

    private void Update()
    {
        if (Input.GetMouseButtonDown(0)) { grabItem(); } //detects when the player has clicked (to try and pick up an item)
        if (hand.transform.childCount >= 1) { rotateItem(); } //true when player is holding an item
    }

    void grabItem()
    {
        //If the player is already holding an item, drop that item
        if (hand.transform.childCount >= 1)
        {
            hand.transform.GetChild(0).GetComponent<Rigidbody>().isKinematic = false;
            hand.transform.GetChild(0).gameObject.layer = 10;
            hand.transform.GetChild(0).parent = null;
        }
        //Otherwise, shoot a raycast and if it hits an object that can be picked up, set its parent to the player
        else
        {
            Ray ray = cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;

            if (Physics.Raycast(ray, out hit, 10, mask))
            {
                hit.rigidbody.isKinematic = true;

                hit.transform.SetParent(hand.transform);
                hit.transform.position = hand.transform.position;
                hit.transform.gameObject.layer = 11;
            }
        }
    }

    //Called when player is holding an item
    void rotateItem()
    {
        if (Input.GetKey(KeyCode.Q)) { hand.transform.localRotation *= Quaternion.Euler(0, rotAmount, 0); }
        if (Input.GetKey(KeyCode.E)) { hand.transform.localRotation *= Quaternion.Euler(0, -rotAmount, 0); }
        if (Input.GetMouseButtonDown(1)) { hand.transform.GetChild(0).rotation = Quaternion.identity; }
    }
}
```

playerVitals.cs:

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

public class playerVitals : MonoBehaviour
{
    public Slider sanity;
    public Slider hydration;
    public Slider sleep;
    public Slider countdown;

    public Animator anim;

    playerDeath death;

    public float gameplayTime = 600f;
    public float currentTime = 0f;

    string sanityDeath = "The monster takes advantage of your delirious state. You have died";
    string hydrationDeath = "Your body fails you due to a lack of water, the monster quickly gets you while you are unconscious";
    string sleepDeath = "You pass out from exhaustion. The lack of sleep finally got the better of you";
    string timerDeath = "The monster has broken in. You have been caught";

    private void Start()
    {
        death = FindObjectOfType<playerDeath>();
    }


    void Update()
    {
        //If any of the vitals are below than a certain amount, play animation to indicate death soon
        if (sanity.value <= 0.2 || hydration.value <= 0.2 || sleep.value <= 0.2) { anim.SetBool("isDying", true); }
        else { anim.SetBool("isDying", false); }

        //Constanly decrease the values if it is not zero. otherwise kill the player
        if (sanity.value > 0) { sanity.value -= Time.deltaTime / 150; } else { death.killPlayer(sanityDeath, 0); }
        if (hydration.value > 0) { hydration.value -= Time.deltaTime / 175; } else { death.killPlayer(hydrationDeath, 0); }
        if (sleep.value > 0) { sleep.value -= Time.deltaTime / 200; } else { death.killPlayer(sleepDeath, 0); }

        //10 minute timer for the player.
        //If the player does not beat the game in 10 minutes, kill them to restart the game
        currentTime += Time.deltaTime;
        countdown.value = 1 - (currentTime/gameplayTime);
        if (currentTime >= gameplayTime)
        {
            death.killPlayer(timerDeath, 0);
        }
    }
}
```

MonsterAI.cs:

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

public class monsterAI : MonoBehaviour
{
    public int movementSpeed;
    public int distanceThreshold;
    public GameObject[] nodepoints;
    public Transform playerHand;
    public GameObject winScreen;

    private Vector3 selectedPoint;

    playerDeath death;
    Animator anim;

    private void Start()
    {
        StartCoroutine(moveToPoint());
        death = FindObjectOfType<playerDeath>();
        anim = GetComponent<Animator>();
    }

    private void Update()
    {
        transform.LookAt(selectedPoint);
        if (Vector3.Distance(transform.position, selectedPoint) > distanceThreshold)
        {
            anim.SetBool("isWalking", true);
            transform.position = Vector3.MoveTowards(transform.position, selectedPoint, movementSpeed * Time.deltaTime);
        }
        else { anim.SetBool("isWalking", false); }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            if (playerHand.childCount >= 1 && playerHand.GetChild(0).CompareTag("Weapon"))
            {
                killMonster();
                return;
            }

            death.killPlayer("The monster got you, you have died", 0);
            transform.LookAt(other.transform);
            selectedPoint = other.transform.position;
        }
    }


    IEnumerator moveToPoint() {
        selectedPoint = nodepoints[Random.Range(0, nodepoints.Length)].transform.position;

        yield return new WaitForSeconds(Random.Range(5, 15));
        StartCoroutine(moveToPoint());
    }

    void killMonster()
    {
        Destroy(gameObject, .75f);

        winScreen.SetActive(true);
        winScreen.GetComponent<Animation>().Play();
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
}
```

doorManager.cs:

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

[RequireComponent(typeof(BoxCollider))]
public class doorManager : MonoBehaviour
{
    public Animator doorAnimation;
    AudioSource audioClip;

    private void Start()
    {
        audioClip = GetComponent<AudioSource>();
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player") && doorAnimation != null)
        {
            audioClip.Play();
            audioClip.pitch = Random.Range(1f, 2f);
            doorAnimation.SetBool("isOpened", true);
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player") && doorAnimation != null)
        {
            //audioClip.Play();
            doorAnimation.SetBool("isOpened", false);
        }
    }
}
```

interactionItem.cs:

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

public class interactionItem : MonoBehaviour
{
    public string Name;
    [TextArea(5, 10)]
    public string Description;

    public bool isUnlocked;
    public Animation interactionAnim;
    public interactionManager manager;

    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("Player") && isUnlocked)
        {
            manager.buttonPrompt.gameObject.SetActive(true);
            if (Input.GetKeyDown(KeyCode.E))
            {
                manager.tmpName.text = Name.ToUpper();
                manager.tmpDescription.text = Description;

                interactionAnim.Play();
            }
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player")) { manager.buttonPrompt.gameObject.SetActive(false); }
    }
}
```

interactionManager.cs:

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

public class interactionManager : MonoBehaviour
{
    public Canvas buttonPrompt;
    public Canvas investigationScreen;
    public TMP_Text tmpName;
    public TMP_Text tmpDescription;
}
```

investigationTable.cs:

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

public class investigationTable : MonoBehaviour
{
    public GameObject[] pages;
    AudioSource audioClip;
    int index = 0;

    private void Start()
    {
        audioClip = GetComponent<AudioSource>();
    }

    //Called by button interaction. Increments to the next item in pages array
    public void nextPage()
    {
        index++;
        foreach (GameObject page in pages)
        {
            page.SetActive(false);
        }
        pages[Mathf.Abs(index) % pages.Length].SetActive(true);
        audioClip.Play();
        audioClip.pitch = Random.Range(1f, 2f);
    }

    //Called by button interaction. Decrements to the previous item in pages array
    public void prevPage()
    {
        index--;
        foreach (GameObject page in pages)
        {
            page.SetActive(false);
        }
        pages[Mathf.Abs(index) % pages.Length].SetActive(true);
        audioClip.Play();
        audioClip.pitch = Random.Range(1f, 2f);
    }
}
```

puzzleBox.cs:

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

public class puzzleBox : MonoBehaviour
{
    public GameObject weapon;
    public GameObject answer;
    public GameObject[] options;
    public Canvas canvas;
    AudioSource audioClip;
    public AudioSource correctAnswer;

    int index = 0;
    Coroutine check;
    bool unlocked;

    private void Start()
    {
        audioClip = GetComponent<AudioSource>();
    }

    //Called by button interaction. Increments to the next item in options array
    //Calls the coroutine checkAnswer() to check if the answer is correct
    public void next()
    {
        if (check != null) { StopCoroutine(check); }
        index++;
        foreach (GameObject page in options)
        {
            page.SetActive(false);
        }
        options[Mathf.Abs(index) % options.Length].SetActive(true);
        check = StartCoroutine(checkAnswer());
        audioClip.Play();
        audioClip.pitch = Random.Range(1f, 2f);
    }

    //Called by button interaction. Decrements to the previous item in options array
    //Calls the coroutine checkAnswer() to check if the answer is correct
    public void prev()
    {
        if (check != null) { StopCoroutine(check); }
        index--;
        foreach (GameObject page in options)
        {
            page.SetActive(false);
        }
        options[Mathf.Abs(index) % options.Length].SetActive(true);
        check = StartCoroutine(checkAnswer());
        audioClip.Play();
        audioClip.pitch = Random.Range(1f, 2f);
    }

    //If the answer is not changed for 1 second, coroutine checks to see if the current option is the answer
    //It does not do this if the correct answer has already been chosen once
    IEnumerator checkAnswer()
    {
        yield return new WaitForSeconds(1);
        if (options[Mathf.Abs(index) % options.Length] == answer && !unlocked)
        {
            GameObject spawnWeapon = Instantiate(weapon, transform.position + (Vector3.up * 5f), Quaternion.identity);
            unlocked = true;
            correctAnswer.Play();
        }
    }
}
```

satanicRing.cs:

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

public class satanicRing : MonoBehaviour
{
    public List<string> collectedTags = new List<string>();
    public Transform[] itemPoints;
    public GameObject puzzleBox;

    public string tag1;
    public string tag2;
    public string tag3;
    public string tag4;
    public string tag5;

    bool collectedAll;

    private void Update() { if (collectedAll) { activeSummon(); } }

    private void OnTriggerEnter(Collider other)
    {
        if ((other.CompareTag(tag1) || other.CompareTag(tag2) || other.CompareTag(tag3) || other.CompareTag(tag4) || other.CompareTag(tag5))
            && other.gameObject.layer == 10)
        {
            if (!collectedTags.Contains(other.tag))
            {
                collectedTags.Add(other.tag);

                foreach (Transform point in itemPoints)
                {
                    if (point.childCount <= 0)
                    {
                        other.transform.GetComponent<Rigidbody>().isKinematic = true;
                        other.transform.SetParent(point);
                        other.transform.position = point.position;
                        other.transform.rotation = point.rotation;
                        other.gameObject.layer = 0;
                        break;
                    }
                }
            }

            if (collectedTags.Count >= 5) { collectedAll = true; }
        }
    }

    void activeSummon() {
        transform.Rotate(Vector3.up, 0.2f);
        foreach (Transform point in itemPoints) { point.Rotate(new Vector3(45, 45, 45), 5); }
        puzzleBox.SetActive(true);
    }
}
```

vitalsInteraction.cs:

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

public class vitalInteraction : MonoBehaviour
{
    public Slider slider;

    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("Player") && Input.GetKeyDown(KeyCode.E))
        {
            slider.value = 1;
        }
    }
}
```

MenuManager.cs:

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

public class MenuManager : MonoBehaviour
{
    AudioSource audioClip;

    private void Start()
    {
        audioClip = GetComponent<AudioSource>();
    }

    public void menu() { audioClip.Play(); SceneManager.LoadScene(0); }
    public void loadGame() { audioClip.Play(); SceneManager.LoadScene(1); }
    public void quit() { audioClip.Play(); Application.Quit(); }
}
```

8.3. References

Cabin Modelling Tutorial:
https://www.youtube.com/watch?v=1aNnERnHRZg&ab_channel=BlenderGuru

Cabin Reference Image: https://wallhere.com/en/wallpaper/707194

Cabin Door: https://i.pinimg.com/originals/80/09/4b/80094bced69fec71dcac93f4ef752316.jpg

Cabin Wood Texture: https://texturelabs.org/textures/wood_209/

2D Assets:
https://i.pinimg.com/originals/a7/b0/c5/a7b0c52a99df98134a96406b61bdce51.jpg

https://www.matsmatsmats.com/images/home-entry/elite/lg/crosshatch-smoke-lg.jpg

https://www.publicdomainpictures.net/pictures/290000/velka/vintage-carpet-pattern-background.jpg

https://www.onlinecarpets.co.uk/cdn/shop/products/MeadowGreen250340JacobeanPatternedWiltonWiltaxCarpet-far_ef7f3e28-911d-4e08-a4d2-fb5c7ffd19e4_600x600.jpg?v=1624544634

https://th.bing.com/th/id/OIP.jERCjnSz52hx9bZtW80vXwHaGG?rs=1&pid=ImgDetMain

3D Assets:
https://assetstore.unity.com/packages/3d/vegetation/trees/dream-forest-tree-105297

https://assetstore.unity.com/packages/3d/environments/urban/abandoned-asylum-49137

https://assetstore.unity.com/packages/3d/props/tools/survival-game-tools-139872

https://assetstore.unity.com/packages/3d/props/interior/books-3356

https://assetstore.unity.com/packages/3d/props/interior/plates-bowls-mugs-pack-146682

https://assetstore.unity.com/packages/3d/props/radio-230712

https://assetstore.unity.com/packages/3d/props/pbr-cardboard-box-110635

https://www.cgtrader.com/free-3d-models/science/other/magic-box-02acf2d0-2163-4ecc-9720-809409e686c0

https://www.cgtrader.com/free-3d-models/furniture/other/simple-wooden-cross

https://www.cgtrader.com/items/4050301/download-page

Monster (Warrok W Kurnialeawan): https://www.mixamo.com/#/?page=1&type=Character
Player (Leonard): https://www.mixamo.com/#/?page=1&type=Character

Audio:
https://www.youtube.com/watch?v=WjJoerDdf-A&t=2393s&ab_channel=DarkestDreams

https://freesound.org/

8.4. Report References

[1] Blender. https://blender.org/. (Last Accessed 17/01/2024)

[2] Unity Technologies. Unity 3D. https://unity.com/. (Last Accessed 17/01/2024)

[3] Freesound. https://freesound.org/. (Last Accessed 17/01/2024)

[4] Audacity. https://www.audacityteam.org/. (Last Accessed 17/01/2024)

[5] Mixamo. https://www.mixamo.com/#/. (Last Accessed 17/01/2024)

[6] Unity. QpaCompany. Nov 2023. https://forum.unity.com/threads/world-space-canvas-cursorlockmode-locked-incompatible.566485/ (Last Accessed 16/01/2024)

[7] Brackeys. CREATE YOUR OWN RENDERER WITHOUT CODE in Unity!. https://www.youtube.com/watch?v=szsWx9IQVDI&t=488s&ab_channel=Brackeys. (Last Accessed 16/01/2024)