[Redacted]'s Game Design Document

Table of Contents

Introduction

This document specifies a design for the gameplay of a game with the provisional title "[Redacted]". In the group for this project are Muireann, Amy, Emily, Kiki , Diana, Vidhisha, Deeva, Raayem, Sumaya.

Scope

This document is intended to be read by the programmers, artists and lecturer (for marking purposes) involved in the design, implementation and testing of [Redacted].

Specification

Concept

The aim of [Redacted] is to produce a fun, intriguing and thought provoking story based point and click puzzle game, that will be made using Unity Engine and using 2D art assets.

Story

Here is the Script for the story
[Redacted] Script

Setting

The game is set in a fantasy mediaeval world, all the scenes will take place within the royal palace.

Game Structure

There will be three paintings that the player will be absorbed into, for each act of the story. The player will also have the chance to look around a bedroom that they spawn into, and a gallery where the first three paintings are located.

In each painting there will be three minigames, and a cipher piece that the player will collect before moving into the next painting.

Player

The game is a single player first person perspective game.

Action

The player will be able to retrieve items, interact with characters, drag and drop certain items for the puzzle minigames. The player can also choose certain dialog options which may lead to different endings such as an instant death if they do not listen to the narrator.

Objective

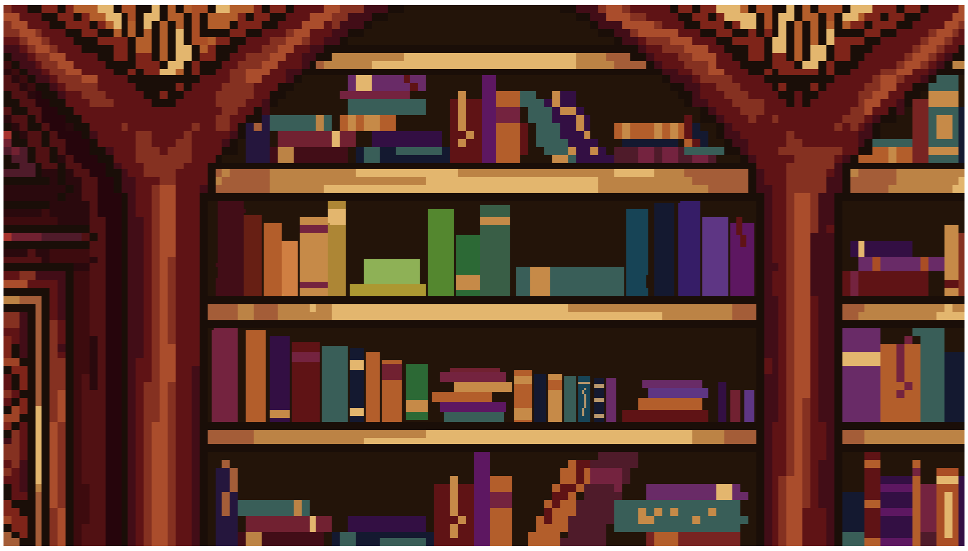
The player's objective is to enter and complete each painting, as they do they discover their missing memories, and learn who the narrator really is.

Here is a document that outlines the mechanics in Act 1.
game cheat sheet

Graphics

The canvas size is 480 x 270 px with animations at 30 frames per second. It is a 2D pixel game.

There will be a dialog box, and the menu will use a simple system for selecting options.



Audio and Sound

Most of our audio has been sourced from a royalty free website known as pixaby and anything not taken from the website has been recorded by myself as sound implementation and optimisation Developer.

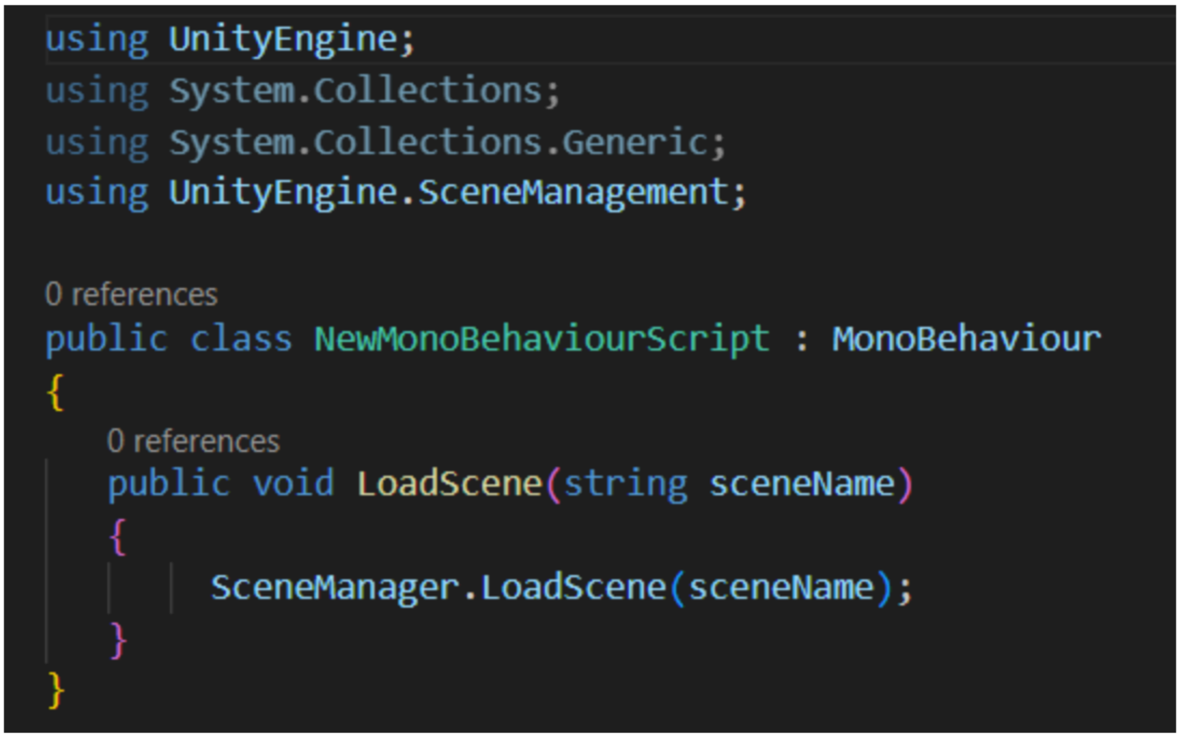
For each of the scenes I found music that would create a good atmosphere that fits the scene , whether it be the eerie lonely tower or the lively ballroom. The “main music” I’ve used refers to the background music of each scene; it pulls everything together and draws the players attention as though they are actually in the game themselves.

To add the cherry on top I have also added sound effects for most things such as , screwing the chandelier in the ballroom, footsteps or even just a howling wind in the tower.

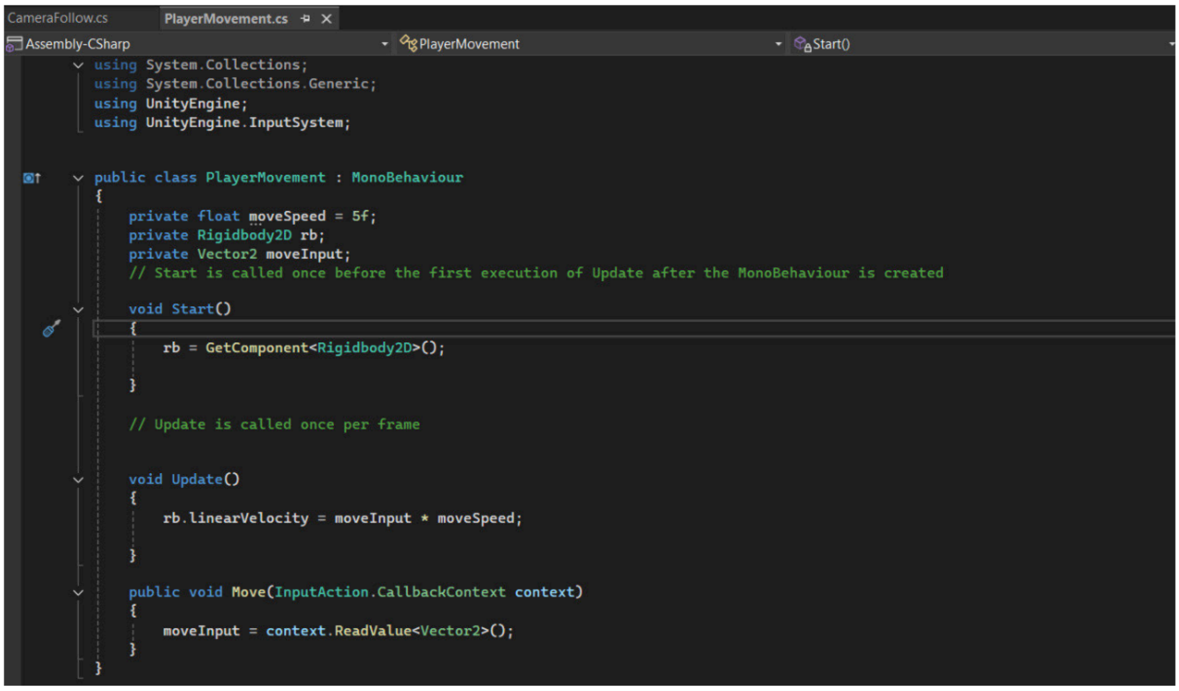
Everything has been balanced together to create an optimal player experience as they immerse themselves in the game.

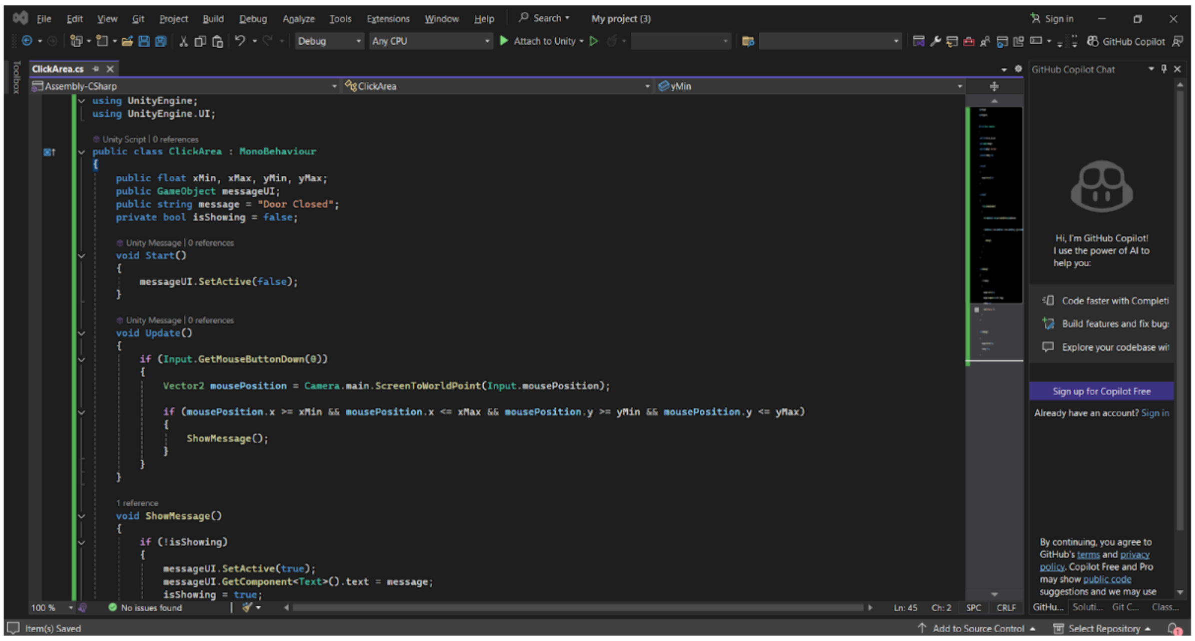
Technical Specifications

The game was developed using the Unity engine, with C# as the primary programming language. For learning and reference, the team utilized online resources like W3Schools’ C# courses. Visual assets, including sprite sheets for character animations, were created using the online photo editor Photopea. While working on the game, we also considered optimization techniques to improve performance, reduce loading times, and manage memory usage efficiently. Minimum and recommended system specifications for each platform are still being determined to ensure smooth gameplay across different devices.



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

0 references
public class NewMonoBehaviourScript : MonoBehaviour
{
    0 references
    public void LoadScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
}
```


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

public class PlayerMovement : MonoBehaviour
{
    private float moveSpeed = 5f;
    private Rigidbody2D rb;
    private Vector2 moveInput;
    // Start is called once before the first execution of Update after the MonoBehaviour is created

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();

    }

    // Update is called once per frame

    void Update()
    {
        rb.linearVelocity = moveInput * moveSpeed;

    }

    public void Move(InputAction.CallbackContext context)
    {
        moveInput = context.ReadValue<Vector2>();
    }
}
```


```
using UnityEngine;
using UnityEngine.UI;

Unity Script | 0 references
public class ClickArea : MonoBehaviour
{
    public float xMin, xMax, yMin, yMax;
    public GameObject messageUI;
    public string message = "Door Closed";
    private bool isShowing = false;

    Unity Message | 0 references
    void Start()
    {
        messageUI.SetActive(false);
    }

    Unity Message | 0 references
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

            if (mousePosition.x >= xMin && mousePosition.x <= xMax && mousePosition.y >= yMin && mousePosition.y <= yMax)
            {
                ShowMessage();
            }
        }
    }

    1 reference
    void ShowMessage()
    {
        if (!isShowing)
        {
            messageUI.SetActive(true);
            messageUI.GetComponent<Text>().text = message;
            isShowing = true;
```

Game Systems and features

Inventory System: Detailed description of the inventory system and how it is integrated into the gameplay (e.g., cipher pieces, restoration tools).

Progression and Unlockables: How players progress through the game, including achievements, unlockables, or new abilities.

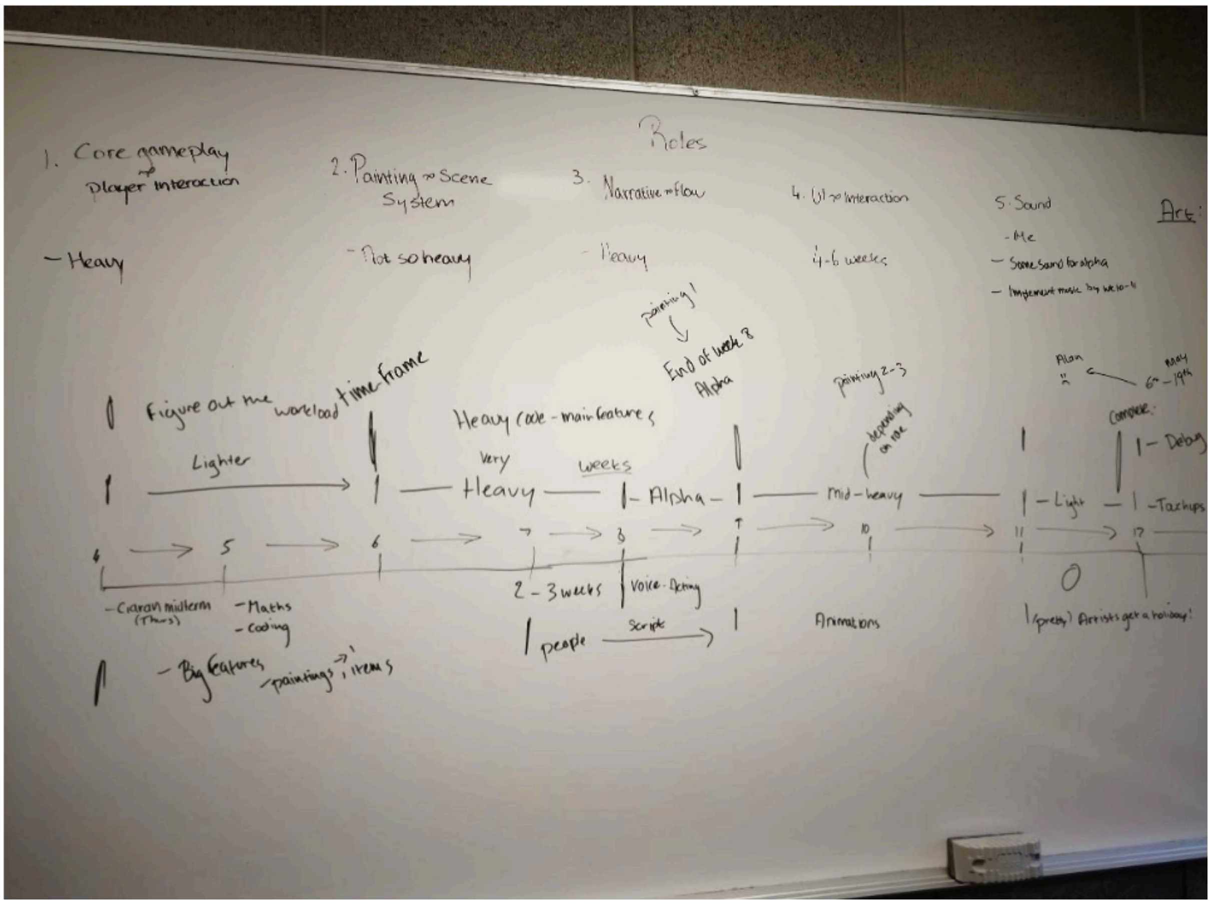
Difficulty: How the game adjusts its difficulty or challenge based on player actions.

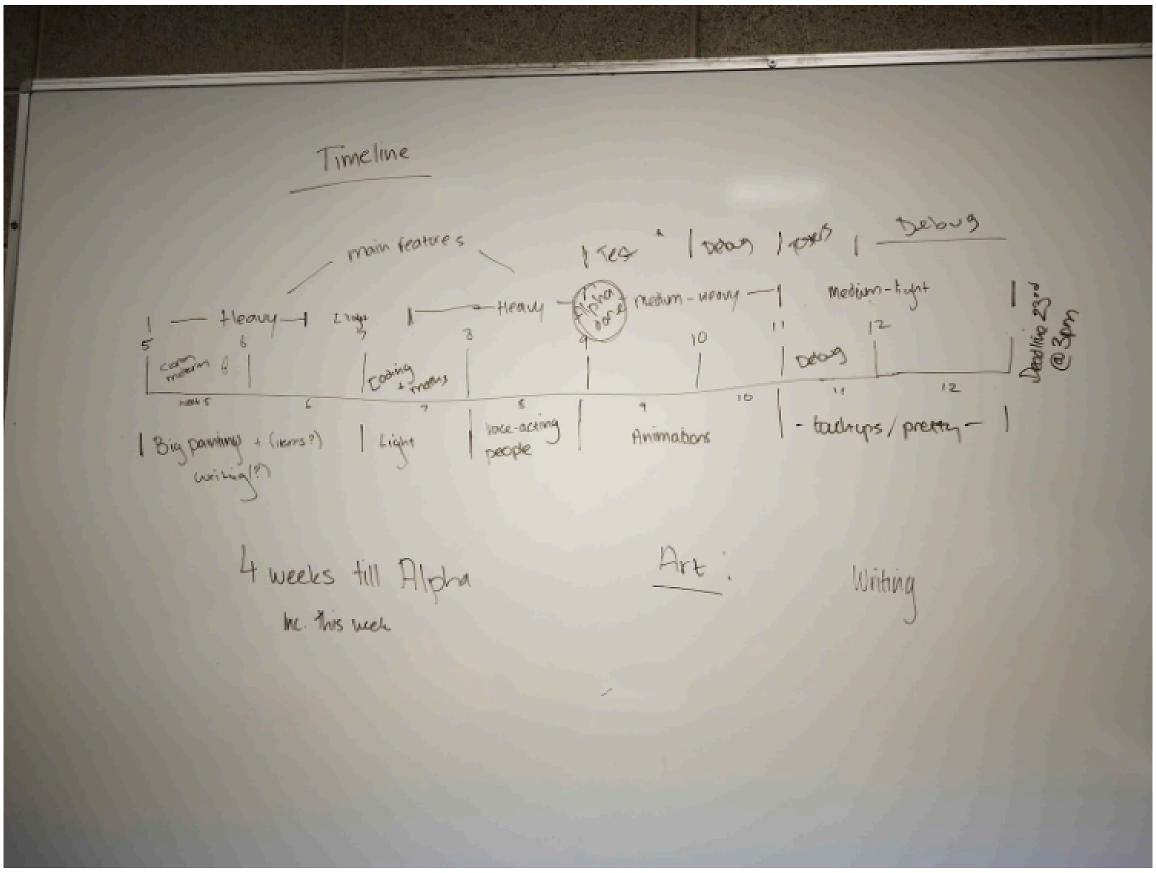
Multiplayer (if applicable): If the game includes multiplayer features, describe the mode(s)and how they function.

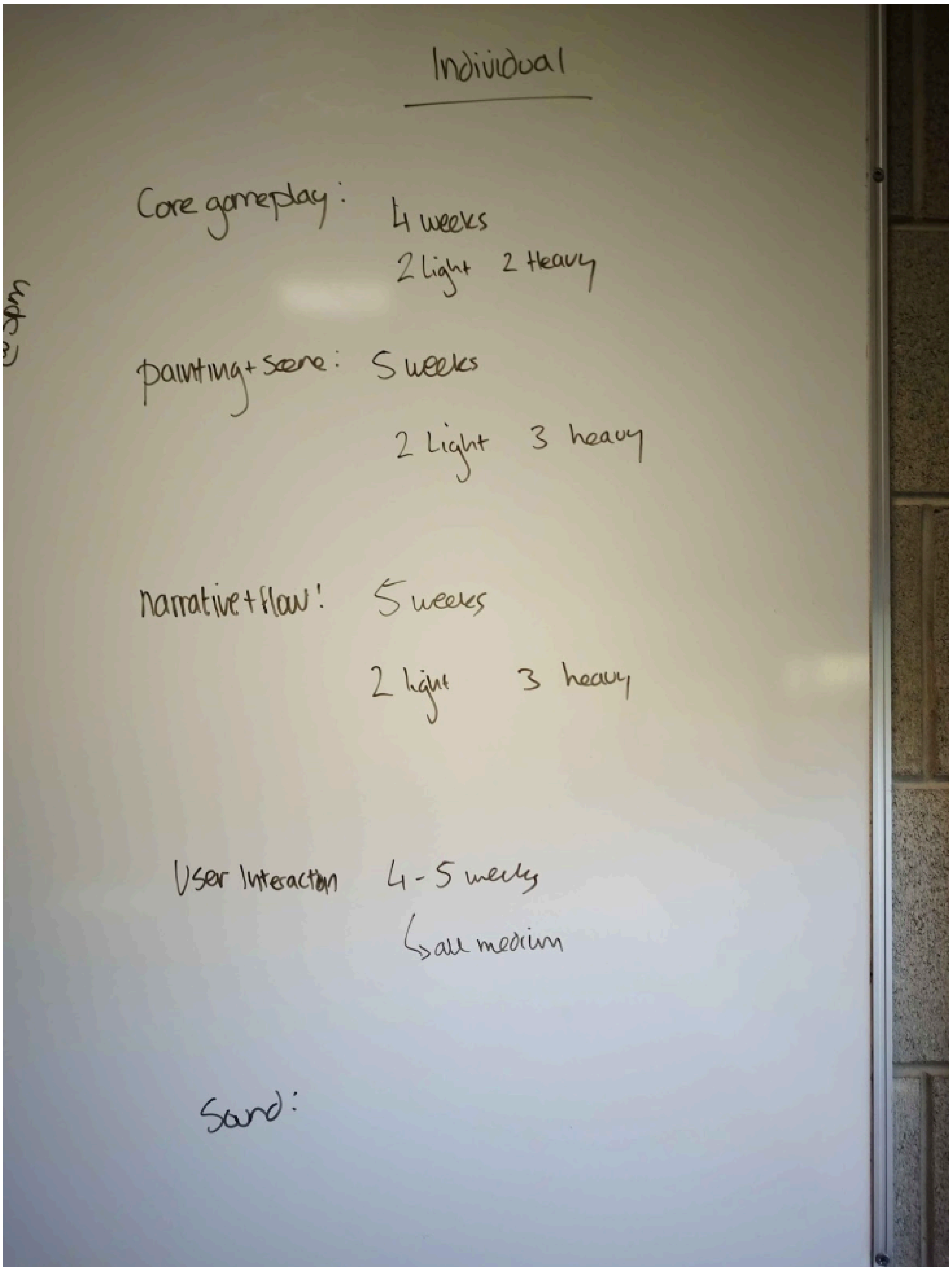
Development Timeline

Our initial timeline was created around Week 4. At the time, we believed it would guide us well, but as the project unfolded, we realized it wasn't as practical as we had hoped. Many tasks were too vague, and we hadn't fully accounted for our individual workloads, especially with a heavier academic load and exams approaching. As a result, the original plan lacked the structure we needed to balance complexity and capacity.

By Week 6, we re-evaluated and created a more realistic and structured timeline. We reassigned work based on difficulty, with some team members taking on two light and two heavy tasks, while others managed three heavy and two light tasks. This redistribution helped balance the workload more fairly across the team. We also built in buffer time for testing, debugging, and polishing, ensuring we wouldn't be scrambling at the last minute. With this improved timeline, we aimed to have our alpha build completed by the end of Week 9, setting a clearer path toward our beta and final release.




Timeline
main features
Test
Debug
fixes
Debug
Heavy
Light
Heavy
Alpha done
medium - heavy
medium - light
5
6
7
8
9
10
11
12
cam midterm
Coding + menus
Debug
Deadline 23rd @ 3pm
week 5
6
7
8
9
10
11
12
Big paintings + (items?)
writing(?)
Light
Voice-acting people
Animations
- touchups / pretty -
4 weeks till Alpha
inc. this week
Art:
Writing


Individual

Core gameplay: 4 weeks
2 Light 2 Heavy

painting + scene: 5 weeks
2 Light 3 heavy

narrative + flow: 5 weeks
2 light 3 heavy

User Interaction 4 - 5 weeks
↳ all medium

Sound:

Development Tools

Unity Game Engine is being used.
Here is our GitHub Repository: https://github.com/MulletWolf/redactedProject
Here is our google drive for the art assets: Game Project Folder