Mohammad Ibrahim Salman, December 2024

Final Year Project Report

Full Unit – Interim Report

Building a Game

Mohammad Ibrahim Salman

Interim Report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Susnas Sourjah

Department of Computer Science Royal Holloway,
University of London

Egham, December 2024

Table of Content

Abstract

This interim report outlines the development and progress of a project centered on creating a video game that prioritizes decision-making, interaction, and innovative design patterns. Beginning with an introduction to games as structured activities requiring skill, strategy, and creativity, the report delves into the motivation behind the project, which is inspired by the evolving gaming industry and the opportunity to explore innovative problem-solving methods in game development.

The project leverages the Godot engine, an open-source platform chosen for its flexibility in 2D game development and its Python-like scripting language, GDScript, which simplifies learning and implementation. The aim is to address challenges in accidental complexity by incorporating design patterns as templates for game mechanics. These patterns will reduce implementation challenges, promote code quality, and ensure future extensibility. Supporting this initiative is the GAME-DP repository, which documents the patterns and provides real-world examples to aid developers.

The report covers essential aspects, including background theory, literature review, and the technology stack, highlighting Godot's advantages over alternatives like Pygame. Progress has been made on core mechanics, software design, and initial testing, with milestones aligning with the project timeline. The testing approach ensures code quality, while reflections on achievements and next steps offer a roadmap for completing the project. Ultimately, the project seeks to combine theoretical insights and practical implementation, serving as both a learning exercise and a stepping stone in the creator’s journey into game development and computer science.

Introduction

Aims & objectives of the project

The aim of this project is to create a game that combines the mechanical and artistic elements of gaming's fourth generation, resulting in a seamless blend of classical gaming components with modern design sensibilities. This game, *Wimbledon's Lot*, captures the nostalgic charm of 16-32-bit graphics in a top-down 2D retro adventure set in the medieval town of Castle Rock. The gameplay includes exploration, combat, inventory management, and interactions with NPCs through dialogue trees and quests, presenting players with a compelling narrative and gameplay experience.

The story revolves around St. Dufresne, a knight with a secret mission. Tasked by a wealthy merchant under the guise of estate management, Dufresne embarks on a perilous journey to uncover the mysteries surrounding the Black Lodge. Along the way, he encounters eerie environments, gathers relics and artefacts, and unravels dark secrets about the town's sinister past. The climax pits him against "The Man of the Black Lodge," whose ambitions threaten global economic freedom and individuality. Players must navigate this narrative tapestry through exploration, decision-making, and combat, weaving their choices into the story's progression.

For designing *Wimbledon's Lot*, the integration of key theoretical frameworks is pivotal to creating an engaging and meaningful experience for players. Flow Theory, developed by Mihaly Csikszentmihalyi, highlights the importance of maintaining a balance between challenge and skill to immerse players in a state of "flow." This theory informs the game's difficulty progression, ensuring that tasks gradually become more challenging as players develop their skills. By keeping the gameplay neither too easy nor overly frustrating, the game maintains player engagement and fosters a sense of accomplishment as they overcome obstacles and complete objectives [1].

Bartle's Taxonomy of Player Types categorizes players based on their motivations—Achievers, Explorers, Socializers, and Killers—offering insights into designing mechanics that cater to diverse preferences. *Wimbledon's Lot* incorporates these principles by providing elements for each type: Achievers can focus on completing quests and collecting rewards, Explorers can delve into the richly detailed environments and uncover hidden lore, Socializers engage with NPCs through branching dialogue trees, and Killers are challenged by combat mechanics [2]. This approach ensures a broad appeal, allowing players to engage with the game in ways that align with their individual playstyles.

Environmental Storytelling Theory, emphasizes the use of physical spaces to convey narrative. This concept is a cornerstone of the game's design, with levels like Castle Rock and the Black Lodge incorporating visual and interactive elements that reveal the story organically. Players encounter relics, journal entries, and environmental cues that hint at the town's dark history and the overarching narrative. By embedding the story directly into the environment, the game deepens player immersion, encouraging exploration and discovery [3].

These frameworks collectively provide a robust theoretical foundation for *Wimbledon's Lot*, ensuring that its gameplay, narrative, and design foster a rich and rewarding experience for players.

Motivation for the project

The motivation for this project is rooted in the dynamic and rapidly evolving nature of the video game industry, a sector that has become a cornerstone of global entertainment. With annual revenues surpassing those of film and music combined, the industry is characterized by both immense opportunities and significant challenges. One of the key drivers of growth has been the shift towards digital distribution, which has transformed how games are marketed, sold, and consumed. Platforms like Steam, Epic Games Store, and mobile app marketplaces have created new pathways for independent developers to reach global audiences. However, this democratization of game development has also led to market saturation, with countless low-quality, repetitive, and overly simplistic free-to-play games flooding the market [4].

This project, *Wimbledon's Lot*, seeks to stand apart by delivering a free-to-play game that avoids the common pitfalls of many mobile and indie games, such as shallow mechanics and repetitive gameplay. As a first-time programmer, my goal is to prove that even with limited resources, it is possible to create an engaging and meaningful experience. By focusing on strategic decision-making, rich exploration, and a compelling narrative, this game is designed to entertain and challenge players, countering the notion that free games must be simplistic or uninspired. Unlike many free-to-play titles that rely heavily on in-app purchases or ad-supported revenue models, this project emphasizes player agency and thoughtful design, offering a rewarding experience without the distractions of monetization strategies [5].

Another key motivation is the exploration of economic and technological drivers shaping the video game industry. The project examines how innovations like social media integration and in-game advertising influence consumer engagement, analyzing their effectiveness in creating long-term value. Emerging business models, such as freemium pricing and subscription services, are also evaluated for their viability in sustaining independent game development. By studying advancements like cloud gaming and social trends like gamification, this project seeks to understand their broader implications, particularly how they shape consumer behavior and developer strategies [4].

Additionally, *Wimbledon's Lot* is an ambitious effort to challenge the perception that first-time developers must start small or limit themselves to uninspired concepts. This project demonstrates that with careful planning and an understanding of core design principles, it is possible to produce a game that rivals commercial titles in depth and engagement. By leveraging open-source tools like Godot and integrating theories such as Flow Theory and Environmental Storytelling, this game aims to deliver an experience that is intellectually stimulating, strategically challenging, and emotionally rewarding—qualities often lacking in many free-to-play offerings.

Technology Choices

The technology for this project is built upon the Godot engine, an open-source game development platform renowned for its versatility and accessibility, particularly in 2D game creation. Godot's architecture is node-based, allowing developers to construct game elements as hierarchical trees where each node represents a distinct functionality, such as sprites for visuals, collision nodes for detecting interactions, or animation nodes for movement. This modular system enhances scalability and reusability, enabling complex mechanics to be built by combining smaller, simpler components. For instance, an enemy character can be constructed by combining a sprite node for its appearance, an animation node for movement, and a collision node to handle interactions with the player.

To enrich the game's visual and auditory experience, assets such as textures, sound effects, and NPC models are sourced from open platforms like OpenGameArt.org and Itch.io's game assets section. These repositories offer high-quality, free assets tailored to indie developers, ensuring a professional look and feel for Wimbledon's Lot without exceeding the budget. Textures are utilized for environmental elements like grasslands and dungeon walls, sound effects provide feedback for combat and exploration, and pre-designed NPC models streamline the development of interactive characters [6 & 7].

Additionally, the game incorporates top-down design patterns and techniques sourced from GDQuest, an educational platform specializing in Godot tutorials and game development resources. GDQuest's guides provide practical insights into implementing mechanics like AI for NPCs, level design, and inventory systems, enabling a polished and efficient development process [8]. These patterns are particularly valuable in maintaining structure and consistency, ensuring that Wimbledon's Lot achieves a professional standard while adhering to its timeline.

By leveraging the Godot engine's node system, open-source assets, and expert design patterns, the technology stack for Wimbledon's Lot supports a

robust, scalable, and visually engaging game, optimized for both development efficiency and player experience.

List of Key Milestones (Timeline)

Here are the key milestones extracted from your project diary and project plan:

- **30/10/2024**: Project initiation, with a clear outline of objectives and goals for developing a 2D retro-style game.

- **05/11/2024**: Began creating the Game Design Document (GDD) and importing assets from open-source platforms like Itch.io and OpenGameArt. Initiated the development process by learning Godot's node and scene structures.

- **12/11/2024**: Detailed study and completion of several GDD sections, including Introduction, Game Overview, Gameplay, Story and Narrative, Levels and Environments, and User Interface (UI). Focused on understanding GDD as a blueprint for the game.

- **13/11/2024**: Completed the remaining GDD sections, including Characters, Art and Visuals, Audio, Technical Details, Marketing and Promotion, and Budget and Schedule, finalizing the document for future reference and team alignment.

- **17/11/2024**: Created the "player1.0" branch for focusing on developing the main playable character. Successfully implemented and debugged basic player movement with vertical and horizontal controls, while addressing initial animation issues.

- **26/11/2024**: Improved player movement mechanics, including proper input mapping and Zelda-style movement normalization. Integrated animations for idle and running states, fixing issues with collision nodes and node hierarchy.

- **03/12/2024**: Implemented smooth animation switching for the player, ensuring idle and running animations function properly. Debugged animation-related issues and finalized basic player movement mechanics.

- **Current Development**: Began world-building and terrain setup for the main hub area, laying the groundwork for creating levels and enhancing the game environment with additional assets and designs.

Background Theory

The development of *Wimbledon's Lot* is grounded in both theoretical and practical concepts from game development and IT project management. The integration of these disciplines provides a framework for designing and managing the game effectively while exploring its value in the broader context of the video game industry.

Analysing value creation in the video game industry provides a critical perspective on the economic and consumer-driven forces that shape game development. The value in games emerges from the interplay between game content, platform features, and user interaction [9]. In *Wimbledon's Lot*, this interplay is reflected in the seamless integration of retro-inspired gameplay mechanics, engaging storytelling, and accessible platform support through Godot. The focus on free-to-play mechanics further aligns with the industry's shift toward alternative monetization models like freemium pricing, which enhance consumer engagement while maintaining economic feasibility.

From the perspective of IT project management, concepts like Project Risk Management, Project Planning: Scope and Time, and Project Controlling Processes: Progress and Performance are integral to the successful development of Wimbledon's Lot. Developing a video game, particularly as a first-time programmer, involves inherent risks such as resource limitations, scope creep, and potential delays. Identifying these risks early and planning mitigation strategies ensures smoother development. For example, using Godot's modular node-based system reduces technical risks by enabling iterative development and adaptability. Similarly, Project Planning: Scope and Time ensures the game is developed within a manageable scope and realistic timeline. By establishing key milestones—such as completing the Game Design Document, implementing core mechanics, and designing the main hub environment—the project remains aligned with its objectives. Tools like Gantt charts and progress tracking help monitor development, ensuring deliverables are achieved on schedule. Furthermore, Project Controlling Processes: Progress and Performance allow for ongoing evaluation of the game's development phases. Regular assessments, such as refining player animations or debugging mechanics, ensure quality standards are met while providing flexibility to address unforeseen challenges [10].

Literature Review relevant to your project

Design patterns for top-down 2D games offer a methodical approach to overcoming the complexities of game design by creating modular, reusable, and scalable solutions. These patterns allow developers to focus on delivering engaging gameplay while ensuring that systems such as AI, collision detection, and inventory management are robust and adaptable. They also resonate with foundational principles of algorithms, which define structured sequences of operations aimed at achieving specific results. Drawing from computational theories, such as Turing's exploration of machine computability and Schrödinger's insights into biological systems, these patterns embody a balance of order and adaptability that drives meaningful gameplay [13-14].

The Entity-Component-System (ECS) pattern is a fundamental framework that separates game entities, their properties (components), and their behaviors (systems). This separation allows for high levels of concurrency, ensuring that entities like enemies, NPCs, and environmental objects can act independently while being influenced by shared systems. In a game like Wimbledon's Lot, this structure enables dynamic interactions between gameplay elements, such as NPCs reacting to the player's actions or enemies adapting their strategies during combat. The modularity of ECS aligns with the computational ideal of breaking down processes into smaller, manageable parts, much like how algorithms optimize tasks through decomposition. Research in ECS highlights its flexibility in supporting concurrency, making it an ideal choice for top-down 2D games that feature multiple simultaneous interactions [11].

Finite-State Machines (FSMs) complement ECS by providing a clear framework for managing state-based behaviours in characters and objects. In Wimbledon's Lot, enemies may transition between states such as "idle," "patrolling," "alert," and "attacking," with transitions triggered by player proximity or specific actions. This state-driven behaviour adds depth and predictability to interactions while keeping the code-base manageable. FSMs mirror algorithmic principles by creating a deterministic system where inputs

lead to defined outputs, maintaining coherence and reducing ambiguity in gameplay mechanics [12].

Observer Patterns further enhance the interactivity of the game world by enabling real-time updates to systems in response to player actions. For instance, when the player completes a quest or picks up a critical item, the UI, inventory system, and related NPCs can react dynamically without direct dependencies. This decoupling of systems aligns with algorithmic efficiency, ensuring that changes in one part of the game do not ripple unnecessarily across others, thereby maintaining a high degree of scalability and maintainability [11].

The application of design patterns in Wimbledon's Lot also reflects the broader theoretical implications of computation and adaptability. Turing's exploration of recursive functions emphasized the importance of structured operations, a concept that translates into game design as the need for predictable yet dynamic systems [14]. Schrödinger's observations about entropy reduction in biological systems offer a compelling parallel: just as living organisms maintain order against the natural progression of entropy, well-designed games maintain structured complexity amidst the dynamic chaos of player interactions [13]. ECS, FSMs, and Observer Patterns exemplify this balance, ensuring that the game evolves naturally while remaining grounded in clear, rule-based interactions.

Additionally, Wimbledon's Lot draws inspiration from biological systems and their ability to adapt and respond to environmental stimuli, akin to neural networks in modern computational models. While these networks aim to reduce human intervention in algorithmic processes, the underlying necessity for structured logic persists. Similarly, the design patterns employed in this game enable a framework where player actions drive the narrative and interactions, but the underlying structure ensures coherence and responsiveness [12]. By embedding these patterns within Godot's node-based architecture, the game achieves a balance of modularity and fluidity, ensuring that new mechanics or story elements can be integrated seamlessly.

Through the integration of ECS, FSMs, Observer Patterns, and algorithmic principles, Wimbledon's Lot delivers a robust and immersive gameplay experience. These patterns not only optimize the technical aspects of

development but also create a game world that feels alive and responsive to player actions. The project underscores the synergy between theoretical computing foundations and practical game design, ensuring that every element contributes to a cohesive and engaging whole.

Implementation So far

The implementation of *Wimbledon's Lot* so far has focused on creating the core functionality of the game, beginning with player input and movement systems. The character's movement was developed using Godot's scripting capabilities, enabling fluid control for both horizontal and vertical directions. Input handling ensures that player actions translate seamlessly into gameplay, allowing for smooth and responsive controls.

Collision detection was implemented by configuring CollisionShape2D nodes, ensuring that the player interacts correctly with the game world and other entities. Adjustments to physics properties and collision masks resolved initial issues, creating a stable framework for further interactions.

Character animations were integrated through AnimatedSprite2D nodes, supporting transitions between idle and running states in all directions. The animation system retains the last movement direction for idle states, enhancing visual continuity and player immersion. Debugging the animation and movement systems involved refining input vector normalization and addressing node hierarchy issues, resulting in a polished and dynamic experience.

Other aspects of the implementation include organizing assets into well-structured folders for efficiency and laying the groundwork for the world design. These efforts have established a strong foundation for the game, balancing technical functionality with player-centric design principles.

Software Engineering

The development of *Wimbledon's Lot* incorporates key design patterns and algorithms to ensure robust and maintainable systems. The Entity-Component-System (ECS) pattern is employed to modularize game elements such as NPCs, enemies, and interactive objects, allowing for scalable behaviours across different components. Finite-State Machines (FSMs) manage state transitions for entities, such as enemies moving between idle, patrol, and attack states, ensuring predictable and dynamic interactions. The Observer Pattern handles real-time updates in systems like UI, inventory, and quest tracking, decoupling dependencies to maintain scalability. Algorithms like A* are used for NPC pathfinding, enabling efficient navigation through complex environments. Together, these design patterns and algorithms create a responsive, dynamic game environment while ensuring the code remains manageable and adaptable.

Testing

Debugging and testing for *Wimbledon's Lot* were conducted through Godot's built-in debugging terminal. The terminal provided detailed error messages and warnings, allowing issues to be identified and resolved effectively. Debugging examples include resolving movement inconsistencies caused by redundant nodes and addressing collisions by refining physics layers and masks. Testing was performed manually throughout the development process to ensure smooth player movement, proper animation transitions, and accurate collision handling. By iterating on feedback from these tests, each feature was refined to ensure quality and consistency.

Code Quality

The code-base is well-documented and structured to facilitate readability and maintainability. Functions are clearly named and modularized, ensuring that each performs a distinct purpose, such as player movement or animation handling. For example, the script for character movement utilizes intuitive variable names like ‘input_vector’ and constants like SPEED for clarity. Debugging tools, such as ‘print()’ statements, were integrated during development to trace issues and verify functionality. Design patterns like FSMs were explicitly implemented within the animation handling logic, ensuring coherent and extensible code for dynamic state management.

Version Control

Version control for the project was managed using Git, chosen for its reliability and ability to track changes effectively. By using branches like "player1.0," features such as character movement and animations were isolated during development, allowing experimentation without affecting the main codebase. This approach ensured a clear history of changes, facilitated debugging, and provided the ability to revert to earlier states if necessary. Git’s collaborative capabilities also positioned the project for future scalability, should additional developers join the team.

Reflections

Reflection on project plan

Looking back at the project plan, it is clear that while it provided a general framework for the game's development, there were several aspects I did not anticipate. Initially, I struggled with defining a clear direction for the game, wavering between creating an experimental study, a 2D side scroller, or a simple arcade-style game like Pac-Man. This lack of clarity led to some inefficiencies during the early stages. Although the project plan outlined the general steps to create the game, it did not account for the challenges I faced in asset importing or the iterative process of writing the Game Design Document (GDD). To stay on track, I've had to go into crunch mode, working intensely to meet the deliverables for the first half of the term. This approach aims to create smoother development time for the second half of the project. While the project plan is fundamentally sound, it will need adjustments to better align with the current pace and requirements of the development process.

Milestones Achieved

Despite the challenges, significant progress has been made in key areas of the project. The GDD has been fully completed, providing a comprehensive blueprint for the game. The main character's movement mechanics, including input handling, collision detection, and animations, have been successfully implemented and refined. Debugging issues, such as node hierarchy problems and animation switching, has enhanced the stability of these mechanics. Additionally, asset organization and early terrain setup have laid the groundwork for world-building, ensuring the project remains on track toward its goals.

Next Steps

The next phase of development will focus on fleshing out the levels, creating dialogue tree-style quests, and implementing NPC behaviours to enhance interactivity. However, to meet the deadlines, I will need to scale back some aspects of the original plan, such as the final boss fight. Instead, the emphasis will be on delivering polished levels and quests that provide a complete gameplay experience within the current scope. Balancing development efficiency with quality will remain a priority as I approach the second half of the project timeline.

Bibliography

[1.0]: Csikszentmihalyi, M. (1990). Flow: The Psychology of Optimal Experience. Harper & Row.

[2.0]: Bartle, R. (1996). "Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs." Journal of MUD Research.

[3.0]: Jenkins, H. (2004). "Game Design as Narrative Architecture." First Person: New Media as Story, Performance, and Game. MIT Press

[4.0]: Marchand, A., and Hennig-Thurau, T., 2013. Value Creation in the Video Game Industry: Industry Economics, Consumer Benefits, and Research Opportunities. Journal of Interactive Marketing, 27(3), pp.141-157.

[5.0]: Griffith, Christopher. (2012) Real-World Flash Game Development; Routledge, 2nd Edition.

[6.0]: OpenGameArt.org, 2024. Open Game Art Repository. Available at: https://opengameart.org/

[7.0]: Itch.io, 2024. Game Assets. Available at: https://itch.io/game-assets

[8.0]: GDQuest, 2024. Godot Tutorials and Resources. Available at: https://www.gdquest.com

[9.0]:Marchand, A., and Hennig-Thurau, T., 2013. Value Creation in the Video Game Industry: Industry Economics, Consumer Benefits, and Research Opportunities. Journal of Interactive Marketing, 27(3), pp.141-157.

[10.0]:Schwalbe, K., 2020. Information Technology Project Management. Cengage Learning.

[11.0]: Redmond, P., Castello, J., Calderón Trilla, J. M., & Kuper, L., 2024. Exploring the Theory and Practice of Concurrency in the Entity-Component-System Pattern

[12.0]: Hadizadeha, M., Koocharia, A., & Sharifia, A., 2024. Algorithm-Free Programming with a Bio-Inspired Computing Machine.

[13.0]: Schrödinger, E., 1944. What is Life? Cambridge University Press.

[14.0]: Turing, A. M., 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society.

[15.0]: Zechner, Mario. Green, Robert. (2012) Beginning Android Games; Apress Second Edition.

[16.0]: Nystrom, Robert. (2014) Game Programming Patterns, Genever Benning; 1st edition. Available at: http://www.gameprogrammingpatterns.com/

Appendices

Project Diary

Entry: 05/11/2024

For my initial stage of development, I will create a Game Design Document (GDD), which will serve as a blueprint for the development team and help ensure everyone involved has a clear understanding of the project. My GDD will include: Game Overview, Story and Setting, Gameplay Mechanics, Level Design, Characters and Enemies, and User Interface (UI).

I will be writing my GDD while simultaneously working on the initial phase of the game, which involves importing the assets. More importantly, I will start preparing right away. Having an idea of what the game will look like, I will proceed with creating nodes, as this engine accommodates developers even if there are changes in vision, such as moving from a 2D game to a 3D one. For this project, I will focus on developing a 2D game as a first-time developer.

As for the GDD, I will create a Word document and store it in the Documents folder with the Project Plan, while importing the gaming assets from open-source websites. I will make notes on new information I read, tasks I perform, and where I need to go.

I imported the assets.

The assets are from the following open-source websites:

- Itch.io
- OpenGameArt

In Godot, a node is a fundamental building block with a specific function (e.g., Sprite, Audio, Script) that represents an element of game logic or behaviour. A scene is a collection of nodes arranged in a tree hierarchy, allowing complex objects or environments to be created and reused

throughout the game. This structure promotes modularity and simplifies the development process.

I resolved an issue with sound asset imports in Godot that were causing errors on Windows. The problem stemmed from macOS-specific metadata files (__MACOSX folders and ._ files) embedded in the sound pack, which Windows couldn't interpret correctly. These extra files were purely metadata that macOS adds, and they aren't actually needed for the game. I deleted the unnecessary files and re-imported the sound assets directly, clearing up the import errors. It's a good reminder of the platform differences that can sneak into asset management!

Entry: 12/11/2024

Today, I delved into what a Game Design Document (GDD) truly is and the essential role it plays in game development. Essentially, a GDD is a comprehensive blueprint for a game, detailing everything from the big-picture vision to the smallest mechanics, ensuring that all team members are aligned and that ideas are consistently documented.

The GDD's outline usually includes several sections:

1. Introduction, which sets the tone and goals of the project.
2. Game Overview, where the game concept and target audience are outlined.
3. Gameplay, explaining the core mechanics and controls.
4. Story and Narrative, describing the storyline and key themes.
5. Characters, detailing the main protagonists, antagonists, and supporting NPCs.
6. Levels and Environments, breaking down each game location and its unique features.
7. User Interface (UI), outlining the menus and HUD elements.
8. Art and Visuals, discussing the style, colors, and overall aesthetic.

9. Audio, which covers the music, sound effects, and ambient sounds.
10. Technical Details, specifying the game engine, platform requirements, and technical constraints.
11. Marketing and Promotion, where target audiences and marketing strategies are planned.
12. Budget and Schedule, estimating costs and setting deadlines.

So far, I've drafted sections on the Introduction, Game Overview, Gameplay, Story and Narrative, Levels and Environments, and User Interface (UI). Each of these sections is shaping up well, helping to solidify the core design and establish the game's atmosphere. Yet, there's still more to cover, like Characters—which will bring depth to our protagonist and NPCs—and Art and Visuals, where I'll outline the game's stylistic direction. The Audio section also needs fleshing out to set the right ambiance. Lastly, the Technical Details, Marketing and Promotion, and Budget and Schedule will round out the GDD, giving it the structure needed to take Wimbledon's Lot from concept to reality.

Entry: 13/11/2024

Today marks a milestone: I finally completed the remaining sections of the Game Design Document for Wimbledon's Lot. Adding depth to the Characters section brought our protagonist and NPCs to life, giving each a unique story and purpose in the game world. The Art and Visuals section now clearly defines the game's stylistic direction, inspired by vibrant, retro 32-bit graphics, ensuring a consistent visual atmosphere. Audio is carefully outlined to set the perfect ambiance, from sound effects to atmospheric music. I finished with the Technical Details, Marketing and Promotion, and Budget and Schedule sections, giving the GDD a robust, well-rounded structure. Now, it's a complete document, ready for others to read and envision Wimbledon's Lot as a reality.

Entry: 17/11/2024

I created a new branch called "player1.0" to focus on developing the main playable character. I began by designing the protagonist that the player will control. First, I imported the character's idle animation, consisting of seven frames, and increased its resolution for better visual quality. Then, I set up a scene named Player (Idle) using a Node, where I incorporated the idle animation. Additionally, I added an oval-shaped Collision Node to the scene to enable the character to interact with and collide with objects in the game world.

In the Game node, which includes Player (Idle) and the newly created camera, I successfully ran the character's animation.

To handle the player's movement, I wrote an initial script. However, the movement required adjustment to accommodate both vertical and horizontal directions as envisioned for the game. My refined code:

```
extends CharacterBody2D

const SPEED = 300.0

func _physics_process(delta: float) -> void:
    var input_vector = Vector2(
        Input.get_axis("ui_left", "ui_right"),
        Input.get_axis("ui_up", "ui_down")
    )
    if input_vector != Vector2.ZERO:
        input_vector = input_vector.normalized()
    velocity = input_vector * SPEED
    move_and_slide()
```

Initially, the script failed to respond to keyboard input. After troubleshooting, I discovered an extra CharacterBody2D node within the player node hierarchy. This redundancy confused the script. Removing the extra node and properly attaching the script resolved the issue. This was a valuable lesson on the importance of reviewing node hierarchies for clarity.

Later, I updated the character assets with a new set of animation frames for both horizontal and vertical movements, replacing the previous set limited to horizontal motion. This change resulted in smoother and more dynamic animations, enhancing the overall gameplay experience.

Entry: 26/11/2024

Today, I worked extensively on improving player movement mechanics. Initially, I encountered several issues where the character failed to move despite the absence of script errors. After detailed debugging, I adjusted the CollisionShape2D and StaticBody2D nodes, ensuring they had the correct configurations and physics layers/masks for interaction.

I also rewrote the player.gd script to implement Zelda-style movement, normalizing input vectors for smoother diagonal movement. By clarifying my understanding of input mapping in the project settings, I ensured proper control using ui_left, ui_right, ui_up, and ui_down.

Animations were further refined by adding an AnimationPlayer node to the character scene. I connected it to player movement states, ensuring seamless transitions between idle and running animations. Organizing assets into clearly labeled folders streamlined workflow and eliminated redundancy, ensuring efficiency moving forward.

Entry: 03/12/2024

Today, I successfully implemented animation switching for the player character. To achieve this, I set up an AnimatedSprite2D node within the Player node and loaded animations for all movement directions.

The script was expanded to handle animation switching:

- "Run" animations play while moving in any direction.
- "Idle" animations play when no input is provided, retaining the last movement direction.

I encountered and resolved minor issues with mixed indentation, which could have caused script errors.

The updated system ensures smooth transitions between animations, providing a polished user experience. With this milestone completed, I began setting up the terrain for the main hub, laying the foundation for world-building and level design.

Future work includes integrating NPCs, quests, and environmental interactions. For now, the character's movement and animation systems are functioning as intended, providing a solid foundation for the next stages of development.

Link to Video Demo

https://youtu.be/3K51mseGieg

Link to Project

https://gitlab.cim.rhul.ac.uk/wlis107/PROJECT.git