Software Engineering Group Project

COMP2002 / G52GRP: Final Report

Project Information

- **Project title:** VR Game for Lower Limb Rehab
- **Sponsor company:** Motus VR
- **Academic supervisor:** Joe Marshall
- **Industry supervisor:** Tom Baker & Jamie Pierce

Team Information

- **Team number:** 44
- **Team members:**
  - Jirui Zhang, 20513930, ssyjz21
  - Brendan Cheng, 20376052, eeybc3
  - Samuel Toth, 20531730, psyst8
  - Gang Yang, 20513291, scygy2
  - Dylan Aldridge Gonzalez, 20424348, pmyda4
  - Chi Hang Ung, 20621654, psycu1
  - Phileas Tang, 20564757, psypt2
  - Miles Butt, 20366523, psmb18
- **Repository:** `https://projects.cs.nott.ac.uk/comp2002/2024-2025/team44_project`

Contents

Part I
Final Report

1 Outline/Introduction

This report presents the final outcomes of the *'VR Game for Lower Limb Rehabilitation'* project, conducted in collaboration with MotusVR. The primary objective was to develop an engaging, accessible, and configurable virtual reality (VR) game that transforms traditional lower-limb rehabilitation into an enjoyable experience.

MotusVR, a company specialising in VR experiences for healthcare and rehabilitation settings, provided the context and initial brief. A key requirement was the integration of the Motus Explore seated treadmill, which captures one-dimensional lower limb motion and combines it with head and hand tracking inputs from a standalone VR headset (Meta Quest 2). The project had to cater to users with varying degrees of mobility by adjusting the level of lower limb exercise while maintaining a dynamic and fast-paced gameplay experience.

The subsequent sections of this report present a comprehensive overview of project's background, development process, and critical analysis, with particular emphasis on the technical implementation of the mini-games. Furthermore, the report examines the challenges and technical difficulties encountered, and solutions devised throughout the project. It also details the project management methodologies adopted, team collaboration, testing processes, and reflections on the team's overall progress. Finally, the contributions of individual team members and their specific roles in the successful delivery of the project are highlighted.

2 Project background and understanding

2.1 Project Overview

The game is centred around the Motus Explore treadmill device, which enables users to simulate walking movements while seated, thereby providing accessible mobility input when used in conjunction with a VR headset. The primary goal is to transform traditional lower-limb physiotherapy exercises into immersive and enjoyable interactive VR gameplay experiences. By gamifying repetitive rehabilitation tasks, the project seeks to enhance patient engagement, reduce therapy monotony, and ultimately support improved recovery outcomes.

This project features a series of carnival-themed mini games that combine physical movements with cognitive challenges requiring users to actively participate and engage their motor and mental skills. Each mini-game offers multiple difficulty options, allowing users to tailor the experience according to their physical abilities and familiarity with VR environments. Throughout the development of this game, both clinical and technical requirements were carefully considered, including hardware constraints of the Motus Explore treadmill and critical VR design challenges such as minimising motion sickness. The resulting product offers a user-friendly VR game that balances enjoyment with meaningful rehabilitation benefits.

2.2 Background of the Project Sponsor: Motus VR

Company Overview Motus VR specialises in developing accessible virtual reality experiences designed to significantly improve the quality of life for individuals facing challenges due to disability, frailty, or chronic conditions. The company has gained recognition for its innovative approaches, particularly in providing immersive virtual reality solutions tailored to the unique needs of patients. These solutions focus on delivering socially engaging experiences that not only enhance cognitive stimulation but also contribute positively to emotional well-being, thereby addressing aspects of isolation and emotional distress [19].

Company Objectives Motus VR's primary objectives are centred around supporting healthcare providers by delivering advanced technological solutions that enhance the standard of pa-

tient care. The company's VR applications specifically aim to facilitate recovery of mobility and physical function in patients experiencing physical deconditioning. Additionally, Motus VR addresses critical psychosocial needs, actively working to reduce feelings of isolation, boredom, and helplessness among patients. By promoting spontaneous social interactions, fostering meaningful companionship, and providing activities that encourage a sense of purpose, Motus VR contributes substantially to both the physical and emotional recovery processes of its users[20].

Existing Products [18]

1. **Motus Relieve:** Motus Relieve is utilised in NHS and private care settings to alleviate boredom, strengthen social connections, promote relaxation, ease anxiety, and provide distraction from pain.
2. **Motus Explore Treadmill:** Motus Explore provides seated, active social VR walking experiences. Incorporating leg movement enables small groups to explore various environments, broadening their horizons and fostering a sense of independence.

2.3 Market Research

2.3.1 Lower Limb Rehabilitation

An initial analysis was conducted to better understand the common conditions requiring lower limb rehabilitation and the standard therapeutic techniques used in recovery.

Pain & Recovery: To design a VR rehabilitation game that is clinically effective, it is important to understand the common injuries and conditions that require lower limb rehabilitation, as well as their respective recovery strategies. Table 1 summarises typical lower limb conditions and the recommended therapeutic focuses based on physiotherapy guidelines.

Condition	**Rehabilitation Focus**
Plantar heel pain	Emphasis on continuous and consistent foot exercise.
Calf strain	Rehabilitation focused on gradual movement reintroduction.
Osteoarthritis of the hip and knee	Physiotherapy exercises to enhance strength and mobility.
Lateral hip pain	Balancing weight distribution between limbs.
Achilles tendon pain	Activity intensity reduction strategies.
Ankle sprain	Use of compression techniques to aid recovery.
Ankle fracture	Assisted walking devices (e.g., Zimmer frames) during early rehabilitation.

Table 1: Common Conditions and Rehabilitation Focus [16]

Common Challenges in the recovery: Research carried out by Bosse et al. [3] investigates the common challenges faced by patients in recovery from severe lower limb injuries:

1. **Low Self-Efficacy**:
   - **Impact**
     - Feeling frustrated during the recovery process
     - Abandon the exercise or treatment plan ahead of time
     - Reacts apathetically to small advances and does not feel satisfied easily
   - **Findings**: For every 1-point decrease in self-efficacy, the Sickness Impact Profile (SIP) score rises by 0.6-0.7 points (representing an increase in dysfunction).
2. **Poor Social Support**
   - **Impact**

  - Feelings of loneliness and helplessness
  - Lack of practical help needed during recovery (e.g., travel, rehabilitation exercises)
  - Depressed mood, leading to decreased adherence to treatment
- **Findings**: Social support per unit decreased, SIP scores increased by 0.5-0.8 points, and rehabilitation effects deteriorated significantly.

These psychosocial barriers highlight the importance of designing a VR rehabilitation experience that maintains patient motivation, provides positive feedback loops, and fosters a sense of achievement.

2.3.2 Exercise Games

To develop a clearer understanding of how physical activity can be effectively integrated into interactive gameplay, research into several well-established exercise-based video games has been conducted. They illustrate how gaming can motivate users to engage in physical movement while maintaining an enjoyable experience.

Wii Fit Developed by Nintendo for the Wii platform, Wii Fit incorporates the Wii Balance Board to facilitate activities such as yoga, strength training, cardio, and balance exercises. The Balance Board's role in engaging lower limb movements is particularly relevant to our project, as it demonstrates how real-world physical activity can be translated into meaningful game interaction[6]; thus sharing a common goal of promoting lower limb activity through gameplay.

Ring Fit In Ring Fit Adventure, players use the "Ring-Con" accessory, equipped with squeeze sensors, and a leg strap to track physical movement via the Nintendo Switch's Joy-Con controllers. Players perform exercises to navigate the virtual world and defeat enemies, making physical exertion an integral part of the gameplay experience.

Holofit Holofit [11] is a virtual reality fitness platform that aligns closely with the objectives of this project, as it integrates VR with external physical equipment. Rather than relying on proprietary hardware, the system is designed to be compatible with a wide range of existing fitness machines, including rowing machines, elliptical trainers, and treadmills. This approach enhances accessibility and user convenience, allowing individuals to incorporate familiar equipment into their VR exercise routines. Unlike Motus VR Holofit positions itself more to the home fitness market, and less so towards medical professionals.

These games illustrate the potential of exercise games to improve user motivation, especially when physical activity is integrated seamlessly with gameplay mechanics. They also highlight the importance of clear feedback, adjustable difficulty, and accessible design, which are key principles in the development of VR rehabilitation games.

2.3.3 VR Games

Superhot Although *Superhot* is not explicitly designed as a fitness-oriented game, it stands out among other virtual reality titles due to its highly engaging gameplay. The core mechanics revolve around a distinctive concept: time advances exclusively when the player moves. Players are required to physically evade incoming projectiles and attacks, utilising direct and intuitive interactions with nearby objects to counter threats.

A significant issue commonly associated with virtual reality games is motion sickness, the susceptibility to which varies significantly between individuals and between games. Unlike many virtual reality experiences that employ joystick-controlled or indirect character movement, gameplay in *Superhot* typically confines the player to a fixed location within each level. The simplicity and effectiveness of these physical interactions provide valuable inspiration for similar implementations in future projects.

2.3.4 Virtual Reality Therapy (VRT)

Virtual Reality Therapy (VRT) is an emerging technology that integrates immersive virtual environments into therapeutic practices to enhance patient outcomes.

- **Benefits of VRT [9]:**
  1. Enhanced Patient Engagement & Motivation
  2. Faster Recovery & Improved Outcomes
  3. Pain Reduction Through Distraction
  4. Real-World Application & Functional Training
  5. Remote Therapy & Accessibility
- **Existing products**
  1. **XR Therapy System [10]:**It is one of the flagship products in Neuro Rehab VR. It creates engaging, interactive virtual environments tailored for patients recovering from neurological conditions such as stroke and spinal cord injuries.
  2. **MindMotion PRO [15]**: MindMotion PRO utilises MindMaze technology to deliver intensive therapeutic training, incorporating evidence-based motor rehabilitation techniques such as action observation, mirror therapy, motor imagery, and constraint-induced movement therapy. Its primary goal is to stimulate neural activity within the affected regions of the brain.

2.3.5 Industry Growth and Unmet Need

The global virtual reality market in health care is forecast to exceed **$28 billion by 2029**, driven by rapid advances in telemedicine and wearable sensor ecosystems [5]. Since the COVID-19 pandemic, tele-health usage has stabilized at 38 times its pre-pandemic baseline, creating strong demand for home rehabilitation platforms [14]. Despite this momentum, systematic reviews reveal that most commercial VR systems prioritize cognitive training, pain distraction, or upper-limb therapy; purpose-built solutions for lower limb gait, balance, and strength remain scarce [4]. This supply–demand mismatch underpins the opportunity addressed by the Motus VR project.

2.3.6 Competitive Gaps in Current VRT Offerings

Leading vendors such as XRHealth, AppliedVR and Neuro Rehab VR demonstrate good engagement metrics, yet three gaps persist:

1. **Sensor Fusion**: limited integration of wearable IMUs, EMG or foot-pressure analytics constrains objective outcome measurement;
2. **Adherence Mechanics**: few platforms employ long-term gamification (adaptive goals, progression tracking) proven to sustain motivation;
3. **Lower-Limb Focus**: the vast majority of commercial catalogues concentrate on upper-limb or cognitive tasks, leaving lower-limb rehabilitation comparatively underserved [7].

Addressing these gaps is central to Motus VR's value proposition.

2.4 Product Positioning

Based on our research into Motus VR and broader market analysis, it is evident that combining virtual reality (VR) gaming with rehabilitative exercise offers significant benefits for patient recovery. VR therapy enhances engagement, motivation, and adherence to physiotherapy programs, leading to improved outcomes across various patient groups.

While there are a growing number of exercise-focused games and VR-based rehabilitation systems currently available, such as Neuro Rehab VR and MindMotion PRO, our research revealed

a notable gap in the market: the absence of VR games specifically dedicated to lower limb rehabilitation. Existing solutions either focus on general fitness, cognitive therapy, with few options tailored to the unique requirements of seated, lower limb exercise in rehabilitation contexts.

Our project positions itself to address this gap by integrating structured physical exercises with interactive mini-games and adjustable difficulty settings, the product bridges the divide between clinical rehabilitation needs and the entertainment value necessary to sustain long-term patient motivation. Furthermore, by utilising consumer-grade VR hardware, the system remains cost-effective and scalable for deployment across various healthcare settings, including rehabilitation centres, hospitals, and home-based therapy programs. Through this focused positioning, our project delivers a novel rehabilitation tool that supports effective lower limb recovery while maintaining high levels of user engagement, thereby contributing meaningfully to the evolving field of VR-based therapeutic interventions.

3 Requirements and Critical Analysis

In this section various sources of requirements have been analysed and collected to create a complete requirements table.

3.1 Sponsor Brief Requirements

- **Functional Requirement**

  1. **Integration with the Motus Explore Treadmill:** The game must utilise the Motus Explore seated treadmill device, which provides a simple one-dimensional input based on the user's walking speed.
  2. **Configurable Exercise Intensity:** The system must allow the exercise level, particularly lower limb involvement, to be adjustable.

- **Non-Functional Requirement**

  1. **Standalone VR Compatibility:** The game must be developed to run on a standalone VR headset, specifically targeting the Meta Quest 2 platform owned by the University.
  2. **Maintained Fast-Paced Gameplay:** Even at lower exercise levels, the game must retain a sense of pace and urgency.
  3. **Open-Ended Game and Aesthetic Design:** The game concept, mechanics, and visual design are left open for the development team to decide.
  4. **Emphasis on Gaming Experience:** Teams with prior gaming experience and those who can reference relevant game design principles are encouraged.

3.2 Hardware Facility

The development is based on the two aforementioned hardware platforms, and compatibility with the hardware must be considered throughout the development process.

Motus Explore Treadmill [18]

- **Advantages:**

  1. Safe lower body movement - building strength and stamina in all major muscles
  2. Improves flexion and extension of leg and ankle joints
  3. Works anywhere – wireless treadmills no Wi-Fi, internet, mobile (cell) required
  4. Portable/lightweight easy set up
  5. extending and amplifying reach of clinicians

- **Limitations:**

1. Only a simple 1-directional input which responds to how quickly the user is walking on the treadmill.
2. older age of clients and, thus, lack of interest and familiarity with technology and video games.
3. uncertainty about how older clients would react to the games and safety concerns.
4. inability to correct posture
5. lack of familiarity of therapists with the device.

Meta Quest 2

- **Advantages:**
  1. Fully wireless operation, allowing users complete freedom of movement without the need for external cables.
  2. Lightweight and ergonomically designed, reducing physical strain during extended use.
  3. High-quality display with fast refresh rates, enhancing immersion and minimising visual fatigue.
- **Challenges:**
  1. **motion sickness:** Users, particularly those unaccustomed to VR environments, may experience symptoms such as nausea, dizziness, and disorientation. This requires careful design of game movement mechanics and user interface to mitigate adverse effects.
  2. **Game Interactions:** VR interaction models, especially involving motion tracking and controller input, can be unintuitive for users unfamiliar with video games or digital interfaces
  3. **User Interface:** Designing intuitive and accessible user interfaces in VR presents unique challenges. Menus, settings, and interaction prompts must be embedded naturally within the 3D environment, avoiding excessive complexity that may confuse users, particularly those with limited VR or gaming experience. Traditional 2D interfaces do not translate directly into VR, necessitating innovative UI solutions that maintain immersion while ensuring ease of navigation.

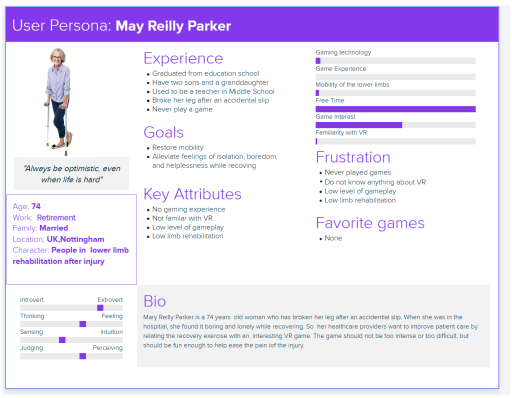
Analysis Given the above analysis of the hardware involved in the project, some additional more detailed project requirements have been laid out:

- **Functional Requirements**
  1. The VR game must be capable of interpreting the one-dimensional walking speed input provided by the Motus Explore treadmill to control in-game movement.
  2. The system must allow configuration of exercise intensity to accommodate users with varying lower limb mobility levels.
  3. The VR game must feature simplified and intuitive interaction models that are accessible to users unfamiliar with VR and video games.
- **Non-Functional Requirements**
  1. The game must implement motion design principles aimed at minimising VR-induced motion sickness, such as slow camera transitions, fixed viewpoints, and smooth navigation mechanics.
  2. All menus, tutorials, and settings interfaces must be naturally embedded within the 3D VR environment.

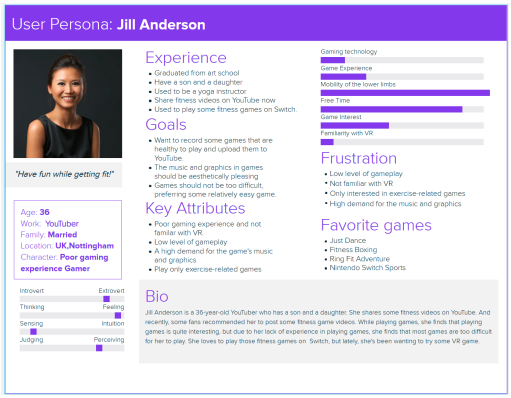
3.3 Target Audience of the Game

The game must be accessible to users undergoing lower limb rehabilitation following injury, with the required level of lower limb activity being configurable to accommodate players with varying degrees of mobility. The primary target audience for this game comprises individuals engaged in lower limb rehabilitation programs. Accordingly, the development process prioritises the user experience from their perspective, implementing a range of difficulty levels tailored to the severity of the users' physical impairments. To support the design process, a set of representative user personas has been developed, illustrating the diversity of potential users and their rehabilitation needs (Figure 1).

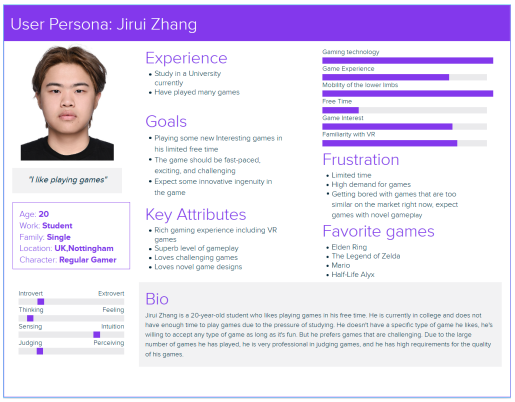
3.3.1 Persona



(a) Older rehab client



(b) Low experience gamer with Lateral hip pain



(c) High experience gamer with Ankle sprain

(d) Project Manager in Motus VR

Figure 1: User Personas for Lower Limb Rehabilitation VR Game

3.3.2 User Stories

- Regular gamer
  - As a regular gamer, I want the game to be fast-paced, exciting, and challenging so that it provides a stimulating and enjoyable rehabilitation experience.
  - As a regular gamer, I want the game to have some innovative ingenuity in the game, instead of copying ideas from other games so that the game will give me a new and exciting experience.
  - As a regular gamer, I want the game to not be too long per level so that I can play the game in my fragmented time.
- Inexperienced gamer
  - As an inexperienced gamer, I want the game to be exercise-related, so that I can exercise while playing the game.
  - As an inexperienced gamer, I want the game to be simple and have a slim newbie tutorial so that I don't get overwhelmed while playing the game.
  - As an inexperienced gamer, I want the game to have its own unique style of graphics and music so that it will attract me to play the game.
- Person in lower limb rehabilitation after injury
  - As a person who is rehabilitating a lower limb injury, I want the game to help me with my rehab so that I can recover from my injury a little faster.
  - As a person in lower limb rehabilitation after injury, I want the game to be fun enough so that it eases the loneliness and frustration I feel during normal rehabilitation exercises.
  - As a person in lower limb rehabilitation after injury, I want the game to not require a lot of lower limb movement so that I can play this game with a lower limb injury.
  - As a person in lower limb rehabilitation after injury, I want the game to be simple and have a slim newbie tutorial so that I don't get overwhelmed while playing the game.

3.3.3 Analysis

- **Functional Requirement**
  1. To accommodate the varying levels of gaming proficiency among users, and their differing preferences regarding challenge, the game must offer configurable difficulty settings.
  2. In response to both user needs and sponsor requirements, the game should maintain a fast-paced and engaging gameplay experience, with a coherent and consistent style across music and visual design.
  3. Considering the specific needs of individuals undergoing lower limb rehabilitation, the level of lower limb activity required during gameplay must be configurable to match the user's physical capabilities.
  4. Given the unfamiliarity of some users with gaming and VR technologies, the game must include detailed tutorials and beginner-friendly guidance to ensure accessibility.
  5. To support rehabilitation goals, the game must allow players to select varying intensities of lower body exercise based on their individual settings and therapeutic requirements.

3.4 Existing research relating to usage of games in rehab

All data in this part are from *a clinical survey about commercial games in lower limb prosthetic rehabilitation* [12]. Overall, 38/82 (46.3%) reported that they use Smart devices in prosthetic lower limb rehabilitation.

Using commercial games in prosthetic lower limb rehabilitation: The majority ($n = 21/38$, 55.3%) indicated that their clients typically spend < 1 hour per week practising these games. Seven (18.4%) reported that their clients spend between one and three hours per week using these games, and the remaining respondents ($n = 11/38$, 28.9%) indicated that the amount of use varies considerably depending on the client. The majority ($n = 29/38$, 76.3%) reported that these games are used *"close to discharge"* (Figure 2).

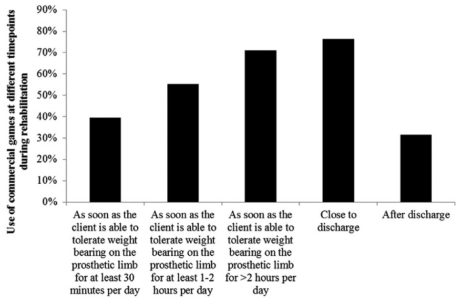


Figure 2: Summary of the responses for time points at which commercial games are used for prosthetic rehabilitation. Respondents were asked to check all that apply.

Not using commercial games in practice: The most selected reason ($n = 15/44$, 34.1%) was that *"they are not familiar with the games"* (Figure 3). In all 9 of the 44 (20.4%) selected *"other"* as the reason for not using the games, only one participant elaborated on this point and explained that their clients are too young to use the games.

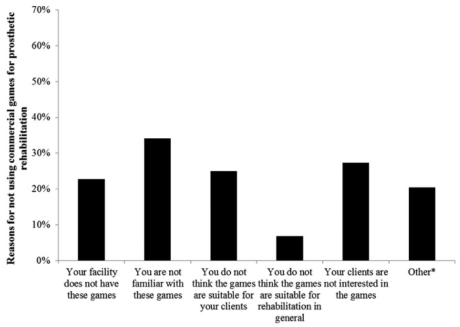


Figure 3: Summary of responses of reasons why respondents do not use commercial games for prosthetic rehabilitation. Respondents were asked to check all that apply.

Analysis

- **Functional Requirements**
  1. The game must be effective in delivering rehabilitation benefits even within short session durations (less than one hour per week).
  2. The game must focus on functional goals relevant to the late stage of rehabilitation, such as independent mobility, balance training, and confidence building.
- **Non-Functional Requirements**
  1. The system must feature intuitive interfaces, guided tutorials, and beginner-friendly onboarding to accommodate users and clinicians unfamiliar with VR gaming.
  2. Supplementary educational materials should be provided to help clinicians integrate the game into rehabilitation protocols confidently.

3.5 Clinical Considerations

Lower limb rehabilitation is a structured, multi-stage process designed to restore mobility, strength, flexibility, and function following injury, surgery, or chronic conditions. A thorough understanding of established rehabilitation practices was critical in informing the functional and non-functional requirements of the project.

Phases of Rehabilitation

1. **Immediate Post-operative Phase (0–2 weeks)**: Focus is on pain control, swelling reduction, and protection of the surgical site. Use of braces, crutches, and elevation is common. Early passive and assisted range-of-motion (ROM) exercises are introduced to prevent stiffness [1].
2. **Intermediate Phase (2–6 weeks):** Patients begin weight-bearing as appropriate, regain mobility, and initiate active strengthening. Gait training and balance exercises become integral [8].
3. **Late Phase (6–12+ weeks):** Emphasis shifts to advanced strengthening, endurance, and functional training such as stair climbing and squats. For athletes, sport-specific drills and plyometrics may be introduced [13].
4. **Return to Activity:** Patients continue exercises independently and may undergo return-to-sport testing. Functional assessments ensure readiness to resume daily or athletic activities [17].

Key components essential to successful rehabilitation [2]:

- **Exercise Therapy:** A personalised programme focusing on lower limb strengthening, joint mobility, and range of motion restoration through exercises such as squats, lunges, balance training, and controlled functional movements.
- **Manual Therapy:** Techniques such as massage, joint mobilisation, and soft tissue manipulation are used to alleviate muscle tension, enhance circulation, and address mobility restrictions, often employed alongside exercise therapy to optimise outcomes.
- **Patient Education and Self-Management:** Educating patients about their condition, rehabilitation goals, and home management strategies—such as pain control techniques, posture optimisation, and progressive loading principles—is critical to ensuring long-term success.

Analysis

- **Functional Requirements**
    1. The VR rehabilitation game must support gradual progression across different rehabilitation phases — starting from gentle, low-intensity movements to advanced strength and functional exercises.
    2. The game must include exercises or tasks that promote balance, stability, and gait training, especially in the intermediate phase of rehabilitation.
    3. The VR game must include tasks that promote the restoration of joint range of motion (ROM) safely and progressively (e.g, Walking).

3.6 Requirements Summary

3.6.1 Functional Requirements

Requirement	Description
Integration with Motus Explore Treadmill	Interpret one-dimensional walking speed input from the Motus Explore treadmill to control in-game movement.
Configurable Exercise Intensity	Allow exercise level, especially lower limb involvement, to be adjustable.
Configurable Lower Limb Activity Level	Adjust required lower limb activity intensity based on mobility capabilities.
Configurable Difficulty Settings	Offer adjustable difficulty levels for varying gaming skills and rehabilitation stages.
Simplified and Intuitive Interaction Models	Feature intuitive, beginner-friendly controls and interactions.
Embedded 3D User Interfaces	Embed menus, tutorials, and settings naturally within the 3D VR environment.
Support for Progressive Rehabilitation Stages	Support gradual progression from gentle movements to advanced exercises.
Balance, Stability, and Gait Training	Include tasks that promote balance, stability, and gait training.
Range of Motion (ROM) Restoration	Include activities that progressively improve lower limb joint ROM.
Effective Rehabilitation in Short Sessions	Deliver rehabilitation benefits within short sessions (less than 1 hour per week).
Support for Lower Body Exercise Selection	Allow players to select and vary lower body exercise types and intensities.

Table 2: Functional Requirements Summary

3.6.2 Non-Functional Requirements

Requirement	Description
Standalone VR Compatibility	Develop for standalone VR, specifically Meta Quest 2.
Motion Sickness Minimisation	Implement VR motion design principles to minimise motion sickness.
Maintained Fast-Paced Gameplay	Maintain a sense of pace and urgency at lower physical exercise intensities.
Coherent Aesthetic Design	Ensure consistent visual and musical style for user experience.
Emphasis on Enjoyable Gaming Experience	Prioritise an enjoyable gaming experience aligned with game design principles.
Onboarding and Tutorials	Provide guided tutorials and onboarding for new users.
Supplementary Educational Materials for Clinicians	Offer educational resources to help clinicians integrate the game into rehabilitation protocols.

Table 3: Non-Functional Requirements Summary

3.7 Overall Reflection

A clear understanding of the sponsor's functional and non-functional requirements has been essential. The explicit requirement to integrate the game with the Motus Explore Treadmill ne-

cessitated careful consideration of user interaction and gameplay mechanics, ensuring that the one-dimensional walking input translated intuitively into engaging and effective VR experiences. The adjustable intensity of lower limb exercise within the game was another critical factor, making the game accessible and beneficial for users with varying mobility levels, thus aligning closely with the rehabilitation goals outlined by clinical standards.

Moreover, a comprehensive assessment of the hardware facilities provided crucial insights into their capabilities and limitations. While the treadmill offered portability and effective lower limb exercise opportunities, its simplistic input mechanism and users' varying technological familiarity posed distinct design challenges. Likewise, the VR platform necessitates careful motion and interface design to mitigate common issues such as motion sickness and user interface confusion, particularly among inexperienced or elderly users.

Clinical considerations were another cornerstone of the project, shaping the design strategy to accommodate multi-phase rehabilitation processes. Through detailed analysis of standard clinical practices, the game was structured to support progressive stages of rehabilitation, ranging from initial gentle movements to more advanced balance and stability training. Incorporating exercises aimed at restoring joint range of motion (ROM) was essential to enhance the game's therapeutic value, directly addressing clinical recommendations for successful rehabilitation outcomes.

User perspectives and experiences were critically important in refining the game's accessibility and engagement. By developing targeted user personas and user stories, the team identified diverse user needs, ranging from experienced gamers seeking engaging, fast-paced content, to novice users requiring simpler interactions and clear, embedded tutorials. Furthermore, the integration of user testing highlighted additional considerations such as the need for naturalistic interaction models within the VR environment and provided insights into improving the overall user experience, further aligning the design with both entertainment and therapeutic effectiveness.

The project's reflection highlights the importance of balancing innovation with usability and therapeutic efficacy. Despite clear benefits and promising potential, limitations remain, particularly concerning user familiarity with technology and potential motion sickness. Continuous user feedback and iterative testing throughout future development phases are strongly recommended. Additional educational resources aimed at clinicians could also enhance adoption and utilisation within clinical practice, addressing concerns raised in the conducted surveys about clinician unfamiliarity with VR gaming applications in rehabilitation contexts.

In summary, the iterative process of combining user-centred design principles with clinical requirements and technological constraints has resulted in a well-rounded, engaging, and effective VR rehabilitation solution. However, ongoing efforts to refine user interfaces, enhance user comfort, and continuously adapt to clinical feedback will be crucial in achieving sustained success and adoption.

4 Game design

4.1 Core Gameplay

The game is designed around an amusement park theme, where four teleportation portals are distributed throughout the virtual park environment (Figure 4). Each portal serves as an entry point to a distinct mini-game. Players can freely explore the park and initiate different game modules by walking into these portals. This structure allows for a cohesive yet modular experience, offering users both freedom of navigation and thematic consistency across gameplay.

Figure 4: Amusement Park and Portal

The four core mini-games (Table 4) are each designed to target lower limb rehabilitation while maintaining engaging gameplay and offering adjustable difficulty settings to accommodate a wide range of physical capabilities and VR experience levels.

Mini-Game	**Inspiration Source**	**Theme**	**Core Gameplay**
Unicycle	Temple Run	Cave system	Ride a unicycle through an endless cave system moving fast enough to avoid being caught and catching coins on the way.
Maze Adventure	Classic Maze Exploration	Haunted House	Find a hidden key in a complex maze and reach the exit to escape. Players can view a map for planning routine.
Chase Escape	Dark Deception	Haunted House	Collect a specified number of coins scattered across the map while evading AI enemies. Escape after collecting enough coins.
Red Light, Green Light	Red Light, Green Light Game in Squid Game	Huge Mascot	Move towards the finish line without being attacked.

Table 4: Summary of Mini-Games: Inspiration and Core Gameplay

4.2 System Architecture

The system architecture of the VR rehabilitation game is structured into five major layers: Input Layer, Game Logic Layer, UI/UX Layer, Hardware Abstraction Layer, and Data Management Layer. This modularized design ensures a robust and scalable system that efficiently integrates hardware interaction, immersive gameplay, and user-friendly interfaces.

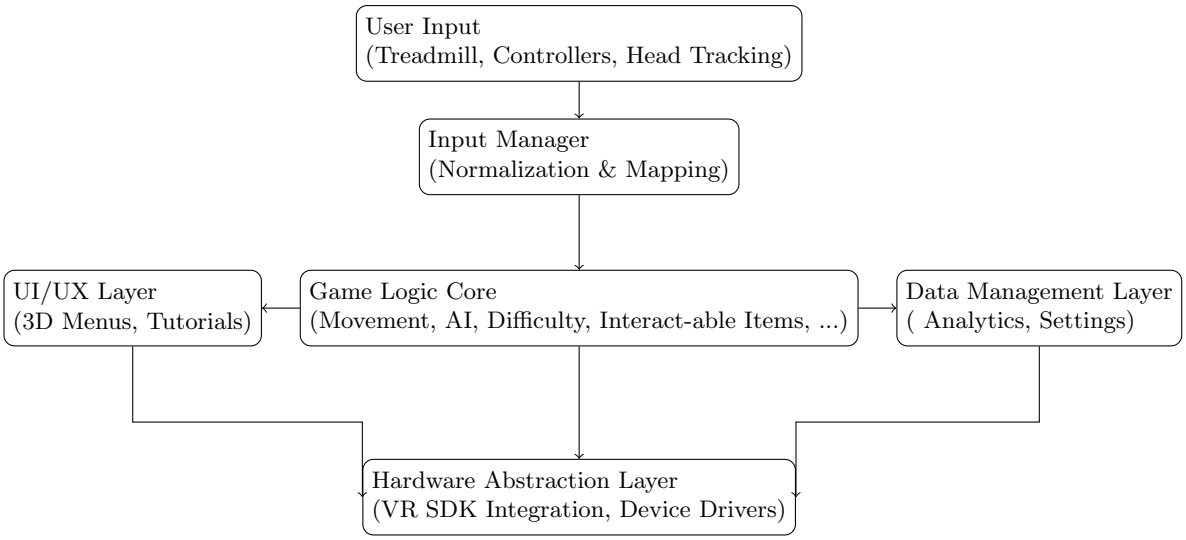


Figure 5: System Architecture of the VR Rehabilitation Game

4.3 Component Design

4.3.1 Game Logic Core

Movement: Three distinct locomotion models are employed to accommodate both gameplay variation and rehabilitation needs:

1. **Gridlocked Movement (Maze & Chase):**
   - Movement direction is independent of user orientation and fixed relative to initial pawn rotation.
   - Directional changes are performed via turn left, turn right and turn around.
2. **Head-Following Movement (Red Light, Green Light):**
   - Walking direction dynamically follows the orientation of the user's head.
   - This approach allows a more natural and immersive navigation style, particularly beneficial for players unfamiliar with traditional controls.
3. **Unicycle-Based Movement (Unicycle):**
   - Inspired by real unicycling dynamics, directional input is based on lateral head tilt.
   - The player tilts their head left or right to initiate corresponding directional turns, thereby simulating balance-based movement control.
   - This mechanic eliminates the disconnect between user orientation and pawn motion, enhancing embodiment and engagement in VR.

AI Enemy: The AI system in the game is built on a finite-state machine architecture, designed to simulate intelligent, responsive enemy behaviour during gameplay. It operates based on the line of sight of AI and player proximity, transitioning dynamically between four primary states (Figure 6):

1. **Idle (Waiting):** In the absence of player detection or patrol conditions, the AI enters a passive idle state. During this state, it consumes minimal resources and remains stationary, awaiting further input or conditions.
2. **Patrol:** When the AI is not engaged with the player, it executes a patrol routine. This involves navigating randomly selected waypoints across the map, simulating natural roaming behaviour. This state improves immersion and encourages player interaction.
3. **Chase:** Upon detecting the player within a defined visual range, the AI transitions into an active chase state. It moves directly toward the player, attempting to close the distance within a limited time window. If the player escapes or exceeds the threshold duration, the AI reverts to patrol.
4. **Attack:** When the player enters the AI's predefined attack range, the AI halts movement and initiates an attack sequence. This include visual effects, animations, or damage infliction depending on the game context. Once the player leaves the attack range, the AI may return to chase state.

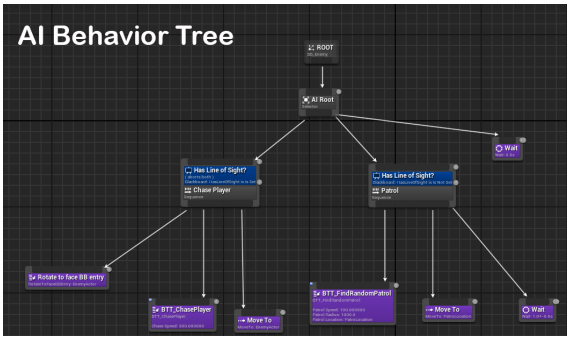


Figure 6: AI Behaviour Tree

Interact-able Items: During gameplay, various interactive objects are placed throughout the environment to enhance engagement and provide strategic opportunities.

1. **Coins (Maze & Unicycle):** Coins are scattered across various maps to incentivize movement and exploration. Players are rewarded with coins upon collection, which contribute to level completion goals.

2. **Health (Maze):** Hearts serve as health restoration items, players can collect hearts to recover lost health during gameplay. The health system is visually represented using heart icons, where each heart symbolizes a portion of the player's vitality.

3. **Light (Maze & Chase):** Sensor lights react to the player's presence, dynamically illuminating areas when players approach and turning off when they depart, guiding player movement.

4. **Boosters (Unicycle):** Within the Unicycle game, boosters temporality speed up your movement input, making the game feel faster paced.

Figure 7: Coins, Health and Light Viewport

Difficulty: To accommodate a diverse user base and support various stages of lower limb rehabilitation, each mini-game adopts a tailored difficulty design to align with its unique gameplay mechanics and rehabilitation goals.

1. **Maze Game:**
   (a) **Easy:** 3 Lives, 1.2x movement speed
   (b) **Medium:** 2 Lives, 1.0x movement speed
   (c) **Hard:** 1 Life, 0.8x movement speed

2. **Chase Game:**

(a) **Easy:** 3 Lives, 1.2x movement speed, Least Coins Necessary

(b) **Medium:** 2 Lives, 1.0x movement speed, More Coins Necessary

(c) **Hard:** 1 Life, 0.8x movement speed, Most Coins Necessary

3. **Red Light, Green Light Game:**

   (a) **Easy:** 1.2x movement speed

   (b) **Medium:** 1.0x movement speed

   (c) **Hard:** 0.8x movement speed

4. **Unicycle:** The game naturally gets harder as the player continue to stay alive due to the chaser speeding up - alongside the unicycle is speeding up, making it harder to turn.

Clearance conditions: To ensure structured gameplay objectives and measurable rehabilitation milestones, each mini-game features clearly defined clearance conditions. These conditions are tailored to the specific mechanics of each game and encourage players to complete targeted tasks, thereby enhancing engagement and promoting therapeutic outcomes.

Game	Clearance Condition
Maze	Find the key, and then reach the exit door.
Chase	Collect a specified number of gold coins while avoiding the AI enemy.
Red Light, Green Light	Reach the finish line without being attacked.
Unicycle	Endless mode.

Table 5: Clearance Conditions of Mini-Games

4.3.2 User Interface (UI)

The User Interface (UI) in the VR rehabilitation game is designed with accessibility, clarity, and immersion as core principles. Considering the needs of users undergoing lower limb rehabilitation, the UI aims to provide intuitive guidance without disrupting the gameplay experience.

1. **Menu**: The user can toggle the menu using the Y button. When the controller is pointed toward the UI, an auxiliary ray is displayed to assist the player in interacting with the interface. When the ray hovers over a button, the button changes colour to indicate that it is selectable. The user can then press the trigger on the controller to confirm the selection.

2. **Map**: By pressing the X button on the right controller, a map appears in front of the player. The map is mainly used in the Maze and Chase games to assist with navigation. A red triangle on the map indicates the player's real-time position, while yellow exclamation marks show enemy locations, both of which update dynamically. Additionally, the locations of in-game items such as health packs and coins are also displayed on the map.

3. **Back View**: When the X button on the left controller is pressed, a back view is shown in front of the player. This allows players to quickly check the situation behind them, such as whether enemies are in pursuit or if the area behind is safe.

4. **Health**: The player's health is displayed in real time in the upper-left corner of the screen. The health bar updates based on game difficulty and the player's current condition. When the player is attacked, the blood colour changes from red to gray. If the player picks up a health pack, the health bar is restored and returns to red.

5. **Plain Text**: Short plain text messages are displayed in front of the player to indicate game events. For example, "Dead" appears when the player fails, "Escape" when the level is cleared, and "Coin Collected" when a coin is collected. These prompts help the player stay informed of the game's progress.

6. **Score**: In the Chase and Unicycle game modes, a score display is fixed prominently at the top-left corner of the player's field of view. It consists of two numbers: the first indicates the number of coins the player has currently collected, updating dynamically as gameplay progresses. The second number shows the total coins required, clearly informing players how many coins remain to achieve the game's objective.

Figure 8: Menu, Map, Back View, Health, Plain Text, Score UI

4.4 Design Justification

This Table 6 demonstrates the traceability between the functional requirements identified and the corresponding game features designed and implemented in Section Game Design. By maintaining this mapping, all design elements are based on our research and contribute to the rehabilitation goals outlined in the project.

Functional Requirement	**Related Feature or Component**	**Section Reference**
Integration with Motus Explore Treadmill	User Input: Treadmill Speed → Movement Speed	4.3.1 User Input
Configurable Exercise Intensity	Difficulty Settings: Easy/Medium/Hard	4.3.3 Game Logic Core
Configurable Lower Limb Activity Level	Movement Models: Gridlocked / Head-following / Unicycle	4.3.3 Game Logic Core
Configurable Difficulty Settings	Difficulty Settings per Minigame	4.3.3 Game Logic Core
Simplified and Intuitive Interaction Models	Input Mapping: VR Controller mappings	4.3.2 Input Manager
Embedded 3D User Interfaces	UI Elements: Menu, Map, Health, Text	4.3.4 User Interface (UI)
Support for Progressive Rehabilitation Stages	Increasing Game Complexity in Unicycle Mode	4.3.3 Game Logic Core
Balance, Stability, and Gait Training	Maze Navigation, Unicycle Tilt Control	4.3.3 Game Logic Core
Range of Motion (ROM) Restoration	Continuous forward walking in Maze, Chase, Red Light games	4.3.3 Game Logic Core
Effective Rehabilitation in Short Sessions	Short-loop Mini-Game Structure	4.1 Core Gameplay
Support for Lower Body Exercise Selection	Game Modules Targeting Different Motion Intensity	4.1.1 Mini Games

Table 6: Requirements Traceability Matrix

4.5 Accessibility and Adaptivity

Accessibility and adaptivity were integral to the development of the VR rehabilitation game, ensuring it accommodates users with varying physical abilities and gaming experience levels. The implemented features specifically focused on facilitating ease of use, promoting user comfort, and supporting diverse rehabilitation needs.

Adjustable Difficulty and Exercise Intensity The game provides multiple difficulty levels across all mini-games, directly influencing factors such as player movement speed, lives available, and the complexity of objectives. By adjusting these settings, the game effectively caters to users with different physical capabilities, ensuring accessibility for individuals at varying stages of rehabilitation. For example, the Maze and Chase mini-games offer reduced player speed and additional lives at lower difficulties, accommodating users with limited mobility or those in early recovery stages.

Intuitive and Simplified Interaction Models Recognising that many users may have limited experience with gaming or virtual reality, the control schemes were designed to be intuitive and straightforward. Movement is primarily controlled by the user's walking speed via the Motus Explore treadmill, while directional inputs and game actions rely on minimal, clear, and accessible interactions with the Meta Quest 2 controllers. This simplified approach lowers the barrier to entry and ensures that the gameplay remains accessible, even to inexperienced or elderly users.

Natural Integration of User Interfaces within VR Menus, maps, and user interface elements were embedded naturally into the 3D VR environment. Essential gameplay information, such as player health status, remaining objectives, and real-time progress indicators, is presented within the player's field of view, minimising disruption to immersion. Menus and maps appear and disappear intuitively through straightforward button presses, enhancing accessibility without overwhelming the user with complex interactions.

Motion Sickness Mitigation The design actively incorporates techniques to minimise VR-induced motion sickness, a common concern among VR users. The game employs stable viewpoints, avoids rapid camera movements, and uses consistent locomotion mechanisms that closely match user expectations based on physical input from the treadmill. This design significantly reduces sensory conflicts, enhancing comfort for all players, particularly those new to VR or susceptible to motion sickness.

Clearance Objectives and Progressive Challenge Each mini-game includes clearly defined and straightforward clearance conditions to guide user interaction effectively. Objectives such as collecting coins, locating keys, or navigating towards an exit were designed to be immediately understandable and easily remembered. Additionally, the Unicycle game progressively increases difficulty over time, adapting naturally to player performance. This adaptive challenge mechanism provides users with motivation to continuously improve their physical abilities through incremental increases in gameplay demands.

Collectively, these implemented accessibility and adaptivity measures ensure that the game provides an engaging, comfortable, and therapeutically effective rehabilitation experience suitable for a broad range of users

4.6 Non-Functional Design Elements

4.6.1 Background Story

> Once upon a time, a legendary amusement park stood at the edge of reality and dreams where lost souls could rediscover strength through play. After a mysterious event sealed the park in silence, it became accessible only to those on a special journey: a path of recovery and rediscovery.
>
> As the player awakens in the centre of this forgotten wonderland, they find themselves surrounded by four glowing portals, each radiating a unique energy. These are not ordinary games, they are trials designed to help the player regain balance, confidence, and control.
>
> Through the *Maze*, the player learns to navigate fear and confusion, chased by shadows that test their courage and spatial awareness. In the *Chase*, they must pursue hope in the form of golden coins, constantly on the move, outpacing the darkness that follows. *Green Light, Red Light* puts them face-to-face with stillness and motion, demanding patience and precision as a watchful zombie turns to catch the unready. Finally, the *Unicycle* path leads them across narrow tracks and broken roads, where balance is not only physical, but also a test of mental resilience.
>
> In this magical realm, progress is not only measured in points or wins, but in the small victories of every step taken. The amusement park mirrors the player's journey, blending healing with adventure.

4.6.2 Music

Initially, music was planned to take on a circus-style sound as the project in its starting stages would take on the theme of a circus (in line with unicycle mini-games as the initial idea), as the project progressed and moved towards Amusement Park games as the theme, the music was rewritten to adapt to the new theme.

Tracks Two tracks were written for the game:

1. **Main Theme** - This theme plays in the Hub-World
2. **Action Theme** - This theme plays in the mini-games

The Main Theme is an upbeat track that brings a happy go lucky feel which affirms the safety and comfortability of the Hub-World for the User. The Action Theme however takes on a completely different sound as it is written to be intense, get users in the mood to move and draw out adrenaline, engaging the user and enhancing the experience by adding another layer of immersion to the game. Ableton Live 10 Suite was used to produce the tracks for this game.

4.6.3 Map Design

Maze Game The Maze map is designed as a haunted house environment featuring a procedurally generated labyrinth. The layout is moderately complex, requiring players to navigate through longer and more intricate corridors.

- **Clearly marked spawn point and exit gate:** The player always knows the start and end location.
- **Mini-map support:** A mini-map is shown in the game to help players plan their route effectively.
- **Strategic key placement:** Keys are located at key junctions to encourage deeper exploration.
- **AI enemy patrols:** Enemies roam main paths to increase tension and motivate directional decisions.

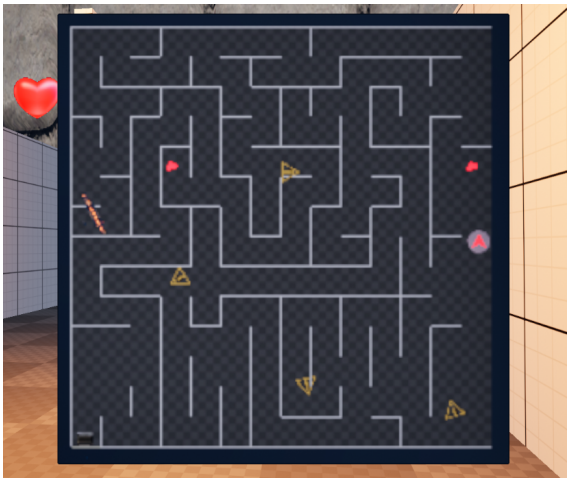


Figure 9: Maze Game Map

Chase Game The Chase Game map is optimized for dynamic pursuit and evasion mechanics. It ensures a balanced experience between therapeutic movement and immersive gameplay.

- **Moderate size:** Designed to avoid fatigue and confusion while offering enough space for meaningful movement.
- **Fully connected paths:** No dead ends; all paths are interconnected to support continuous forward motion.
- **Looped design:** The layout avoids one-way or trap paths to provide consistent escape opportunities.
- **Item placement for motivation:** Coins or crystals are distributed along all active paths to reward exploration.
- **Chase-optimized structure:** A balance of corners, corridors, and open areas ensures dynamic pacing.

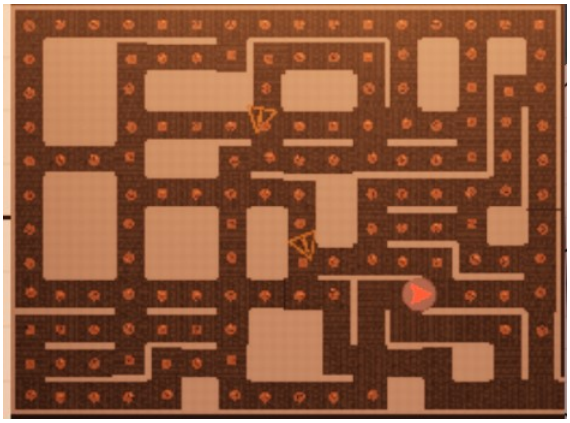


Figure 10: Chase Game Map

Red Light, Green Light Based on the "Squid Game" concept, this mode uses a large open map to simulate start-stop walking exercises.

- **Straight track layout:** Simple path from start to finish with no branching to reduce cognitive load.
- **Safe zone indication:** Ground colour or markers show when to move or stop.
- **Large-scale field:** Offers enough space for multiple players or varying movement speeds.

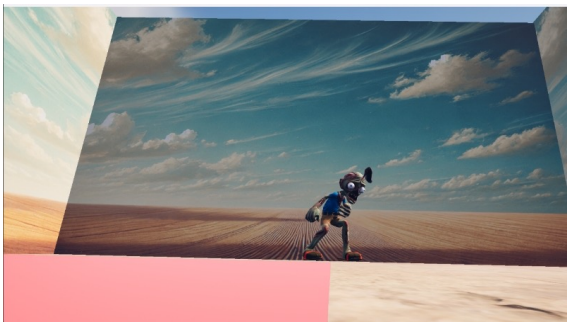


Figure 11: Red light Green light Game View

Figure 12: Unicycle Game View

Unicycle The unicycle game takes place in a procedurally generated cave system. As the player explores, they will discover new tiles and new challenges.

- **Coins as a scoring system:** Coins are distributed across the environment, giving the player a goal and a providing a score at the end of the run.
- **Tiles with unique challenges:** Because each tile can be customized, it is possible to give tiles unique challenges, some of which include: Squeezing through a tight rockfall gap, balancing over a narrow plank, and slaloming through a series of tight turns.
- **Power-ups:** Throughout the map there will be randomly placed power-ups, such as big coin drops and boosters. Some of these will require the player to reach up with their hands, or quickly turn to collect the boost.
- **Chasing enemy:** There will be an enemy AI chasing the player giving a sense of urgency. As the level progresses they will speed up, allowing the game to suite a variety of different ability levels.

4.6.4 Gameplay Design

Several gameplay elements have been carefully implemented, ensuring that each component directly contributes to the rehabilitation objectives while providing a compelling user experience.

- **Feedback Mechanisms:** In the Maze game, the AI chasing the user provides real-time feedback by alerting the player to move faster as it gets closer. This dynamic feedback encourages more frequent lower limb use, adding an element of urgency and improving the player's rehabilitation as they progress through the game.
- **Dynamic Difficulty and Rehabilitation Progression:** Each mini-game incorporates clearly defined, scalable difficulty settings to cater to users with varying levels of mobility and gaming proficiency. For example, in the Maze and Chase games, the difficulty adjustments modify the player's movement speed, available lives, and the complexity of tasks. This flexibility ensures users remain consistently challenged without becoming overwhelmed, effectively supporting gradual physical rehabilitation progression.
- **Strategically Designed Objectives:** Gameplay objectives across the mini-games are specifically structured to foster therapeutic outcomes. Objectives such as navigating a maze to locate keys and exits, evading AI enemies, or reaching a finish line effectively combine cognitive challenge with physical activity. These clear, achievable targets encourage sustained physical effort and enhance cognitive engagement, essential elements for effective rehabilitation therapy.
- **Immersion and Presence** An essential component of the gameplay design is the deliberate cultivation of immersion, which plays a vital role in maintaining motivation throughout repetitive rehabilitation exercises. The use of first-person perspective, head tracking,

and controller-based interactions helps to closely align in-game actions with users' real-world physical movements. Environmental audio cues, such as the approach of AI enemies or the sound of coins being collected, contribute to situational awareness and deepen the player's sense of presence within the game world. Each mini-game also adopts distinct visual themes—such as the eerie, enclosed layout of the Maze or the dynamic cave setting of the Unicycle game — these provide a unique atmosphere that supports gameplay variety while maintaining psychological engagement. Furthermore, UI elements are embedded within the 3D environment rather than presented as separate 2D overlays, maintaining the illusion of a continuous world. Interactions are designed to feel natural and responsive—for example, collecting coins with hand gestures or reacting to visual light cues in Red Light, Green Light.

Through these thoughtfully implemented gameplay features, the VR rehabilitation game successfully balances therapeutic effectiveness, player engagement, and accessibility, delivering a rewarding and inclusive rehabilitation experience.

5 Implementation

5.1 Environment and Tools

One of the first tasks that was undertaken in this project was to research and understand the technology that is commonly used in the VR game development field, and evaluate which choices would work best for our group.

1. **Game Engine**: Research was conducted into the most commonly used game engines for VR development. Both Unreal Engine and Unity are widely used by the community both hobbyist and professional; both have strong support for VR and ship with a number of subsystems and components that allow you to ship features at an accelerated rate. Ultimately Unreal Engine was chosen due to it's reputation for industry leading VR support as well as existing familiarity of some of the team members.

2. **Programming Languages**: Within Unreal Engine itself, there are two main ways through which to add your own custom functionality: either with C++, or Blueprints. The advantages to using C++ include complete access and control to all engine features, increased maintainability of the codebase as well as the ability to use tradition version control in teams; on the other hand, blueprints offer a rapid implementation-testing loop and a shallower learning curve. Whilst initially we had planned to use these two systems in tandem so that their benefits were in complement, we found that the speed of development of blueprints was too valuable. In addition the team would have to learn C++ alongside the already complicated Unreal Engine, and so we opted for a blueprint only solution.

3. **Other Tools**: A variety of other tools were used in the development of this game:

    - **Version Control Systems**: *Git-Lab.*
    - **Modelling Tools**: Blender
    - **IDE**: Visual Studio 2022
    - **Music Producing Tools**: Ableton Live
    - **Maze generation plugin:**[1] Maze Generator plugin
    - **Free Models & Resources from third party sites:** *Fab* and *Mixamo.*

[1] `https://www.mediafire.com/file/0wau3k9d8cj48d7/Mazes_plugin_5.0.3.7z/file`

5.2 Game System Architecture and Modular Design

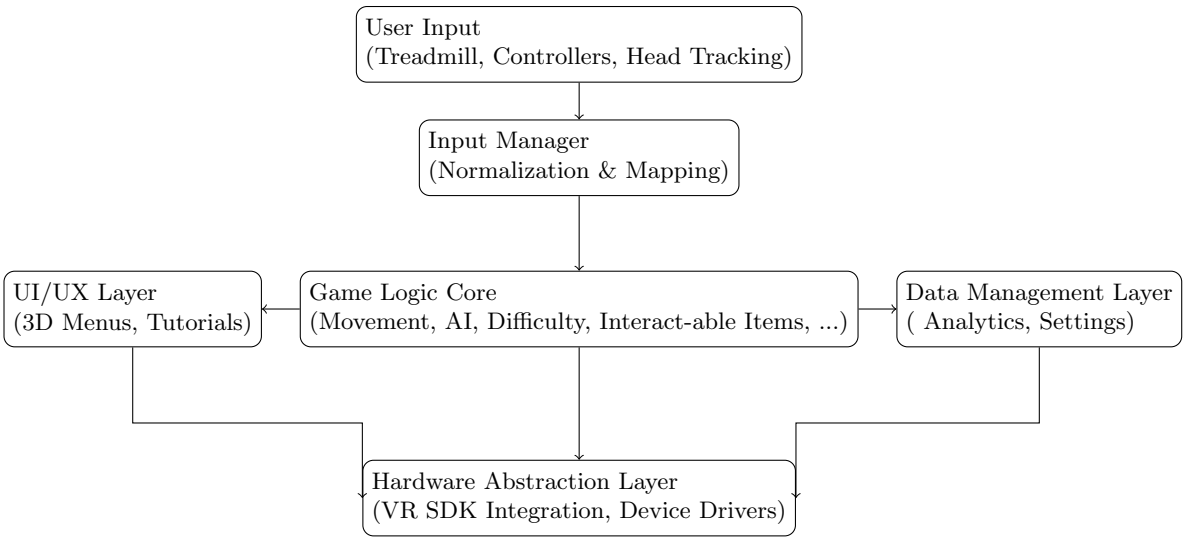


Figure 13: System Architecture of the VR Rehabilitation Game

VRPawn Control Module: Each mini-game in the project uses a customized VRPawn Blueprint derived from Unreal Engine's default VRPawn setup. These VRPawns are responsible for player movement control, UI integration, and temporary data storage relevant to gameplay. The separation of VRPawn logic per game ensures modular design and allows each gameplay mode to define its own interaction and control scheme.

UI Binding and Interaction: The UI system is organized modularly, with each mini-game assigned its own Widget Blueprint to manage its unique interface layout and logic. These UI elements are integrated into the VRPawn and are visually anchored to the player's view to maintain immersive interaction. Each UI module handles basic in-game navigation, task-related prompts, and dynamic feedback during gameplay.

Game Level/Scene Transition Module: The level transition system enables players to enter mini-games through portals in the central Game Hub and return to the hub at any time via the in-game menu. Each game also supports difficulty selection through the same menu system. Transitions are managed through Blueprint-controlled logic and Unreal Engine 5's level streaming framework, ensuring seamless navigation between different scenes.

User Input Management: The project employs a modular input configuration system using Input Actions and Input Mapping Contexts (IMCs). High-level actions such as movement and interaction are abstracted through Input Actions and then mapped to specific controller inputs via IMCs. This design provides hardware flexibility and maintains input consistency across all mini-games.

5.3 Code Quality and Maintenance

Organising code in blueprints: Although blueprints make development faster, as a project becomes bigger, it can become a challenge to keep things well organised. Some of the issues we encountered were:

- When a blueprint becomes large it can become difficult to discover and quickly navigate to specific bits of code that can be located far apart on the 2-dimensional grid.

- It can be more difficult to see the nesting structure of the code (i.e which bits are inside loops or if statements) compared to textual representation that make it immediately clear over via indentation.
- If a single event or function becomes large, then it can be more challenging to see at a glance the overall goal of the function or event. (N.B. this is also an issue in traditional textual programming, however, we argue that it is exacerbated by the visual paradigm)
- It is easy for the choice of position of nodes to cause many wires to overlap and the *code* to become difficult to read. One can truly see the meaning of spaghetti code after reading some blueprints.

Over the period of the project, a number a number of strategies were discovered to to overcome these blueprint related issues as well as enforce the kind of good coding practices that are essential to developing maintainable software. Some of our informal guidelines include:

- Keeping events and functions short, splitting out into a custom event whenever necessary.
- Making sure to use the 'pin' nodes to avoid as much wire *spaghetti* as possible.
- Making use of `ActorComponent` classes whenever possible to split up and group related functionality. This also has the benefit of allowing other classes to easily compose this functionality with their own.
- Making use of the code commenting and boxing functionalities to explain and group together content.

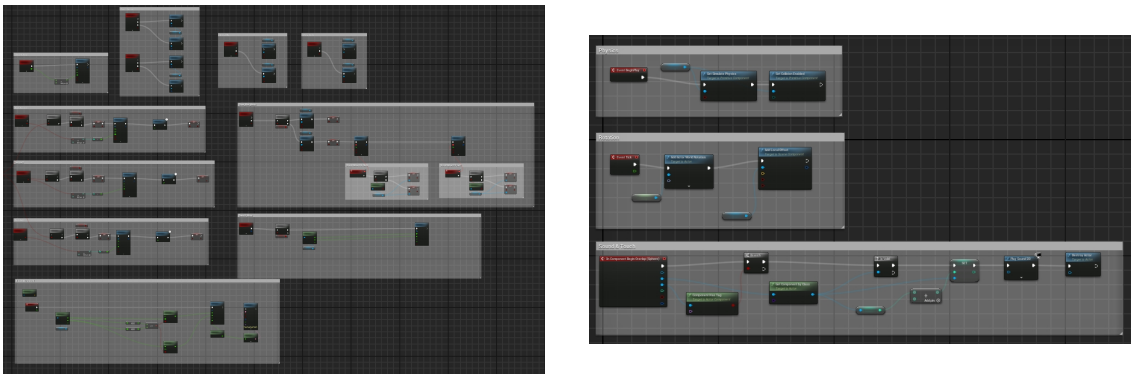


Figure 14: BluePrint Boxing

5.4 Key Feature Implementation

VRPawn-Based UI Attachment: Each mini-game creates a dedicated Widget Blueprint to design its user interface, including buttons, text, and indicators. These UI elements are instantiated in the VRPawn using the `Create Widget` node and assigned to a `Widget Component`. The component is then attached to the player's camera within the VRPawn, ensuring that the UI remains directly in front of the user and follows their head movement, enhancing immersion.

UI Interaction via VR Controller: UI interaction is mapped to the VR controller's input actions. For example, directional navigation within menus is controlled via the joystick axes, while selections are confirmed by pressing down on the joystick. Button behavior is implemented using event bindings within the Widget Blueprint, typically through `OnClicked` delegates for each interactive element.

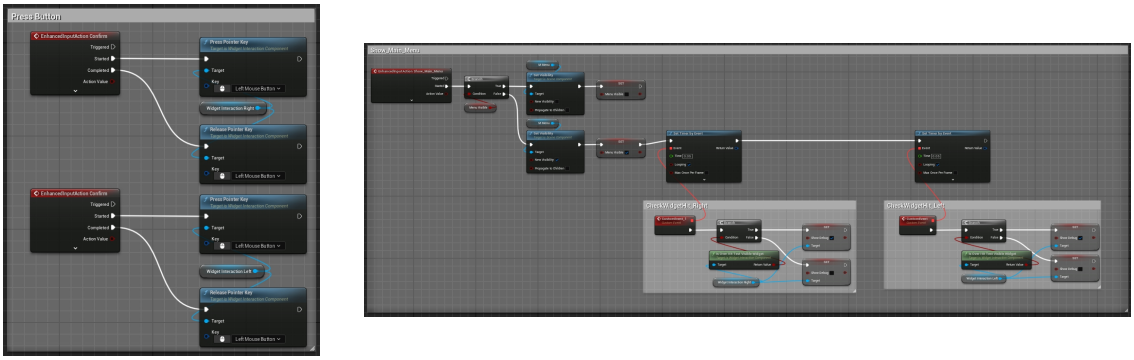


Figure 15: Button Press and Main Menu

Scene and Difficulty Switching: Level transitions are handled using UE5's `Open Level` or `Load Stream Level` nodes, depending on whether a full load or streamed transition is required. Menu widgets provide navigation options to switch between mini-games or return to the Game Hub. Additionally, each mini-game supports dynamic difficulty selection (Easy, Medium, Hard), controlled by setting a difficulty variable via the menu prior to level loading.

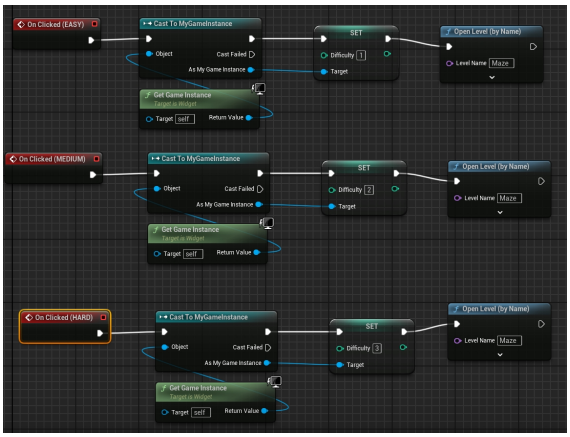


Figure 16: Difficulty Switching

Enemy AI The enemy AI system in our game leverages Unreal Engine 5's Behavior Tree and Blackboard components in combination with blueprint-based logic. Enemies operate in two main states: patrolling and chasing. The behavior tree first checks for line-of-sight; if the player is detected, the AI transitions into a pursuit sequence where the enemy rotates to face the player, moves toward them, and performs an attack upon collision. If no player is in sight, the AI patrols by selecting random locations using `BTT_FindRandomPatrol`, navigating to those points, and waiting for a short random delay before continuing. When the player overlaps the enemy's attack sphere, a blueprint sequence is triggered: the system casts to `VRPawn_Maze` and verifies validity, delays 1 second to simulate wind-up, plays a zombie attack montage via the skeletal mesh's animation instance, and invokes the `TakeDamage` function on the player. A looping timer ensures continuous attacks while the player remains in range. Once the player exits the area, the timer is cleared and the animation is stopped. This system allows for dynamic transitions between idle and combat states based on proximity and visibility, and is modular enough to support further extensions such as varied attack types or smarter behaviors.

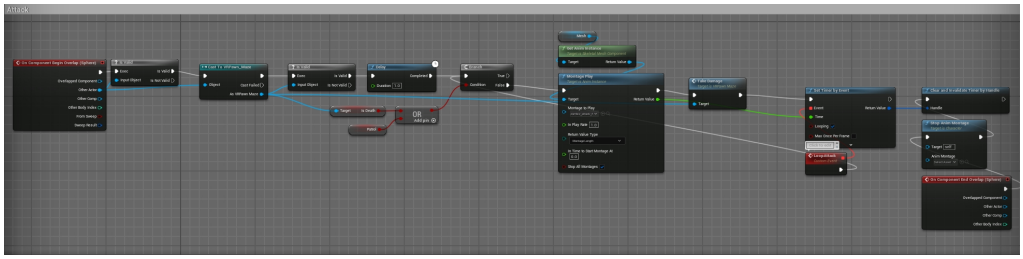


Figure 17: Enemy AI Blueprint

Custom Input System Configuration: The input system uses Unreal Engine's Enhanced Input framework. Input Actions such as `MoveForward`, `TurnLeft`, and `Confirm` are defined as data assets. These actions are linked to physical inputs through Input Mapping Contexts. Within Blueprints, the `Enhanced Input Action` events are used to bind these actions to gameplay behaviour, such as movement or UI control. This setup allows the input configuration to remain abstract and device-agnostic.

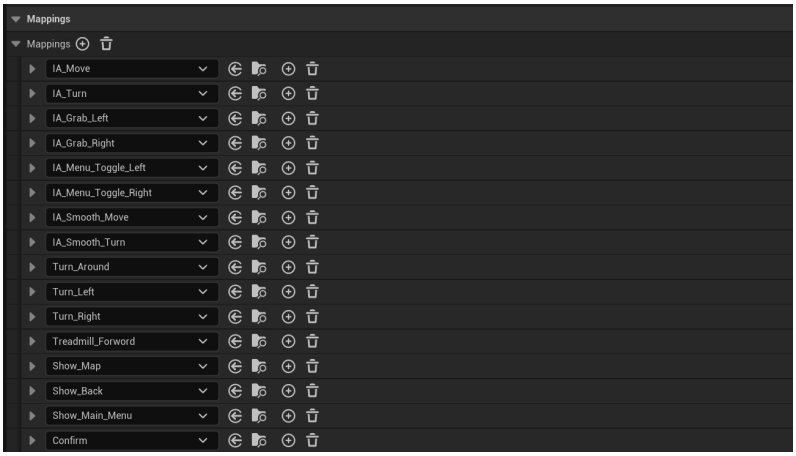


Figure 18: Enhanced Input

Procedural Level Generation: The Unicycle mini-game is designed around an infinite random cave environment through which the player must ride. Technically, this requires the level and world to be generated dynamically as the player progresses. Furthermore, care must be taken to ensure that parts of the level no longer in use are dynamically unloaded, particularly due to the performance constraints typical of VR systems.

To implement this, a structure similar to a doubly linked tree is used. In this setup, each node is a subclass of a base tile blueprint, storing references to its preceding and succeeding nodes. Collision boxes are used to trigger messages that prompt future tiles to spawn their children, or if already spawned, to pass the message further forward. Simultaneously, a similar message is propagated backwards to delete older, unused tiles at the opposite end of the structure. Unvisited directions are also tracked to prevent the creation of orphaned tiles—i.e., tiles that are never de-spawned.

This system enables continuous traversal through a procedurally generated environment while ensuring that only a limited number of tiles remain loaded at any given time. Additionally, because each tile is implemented as a subclass of a default tile class, it retains full control over its visual appearance and spawning behavior, including how many child tiles may be created and from which locations. This modular design not only supports a wide variety of tile styles, but also allows future developers to easily extend the system with custom content without modifying the core framework.

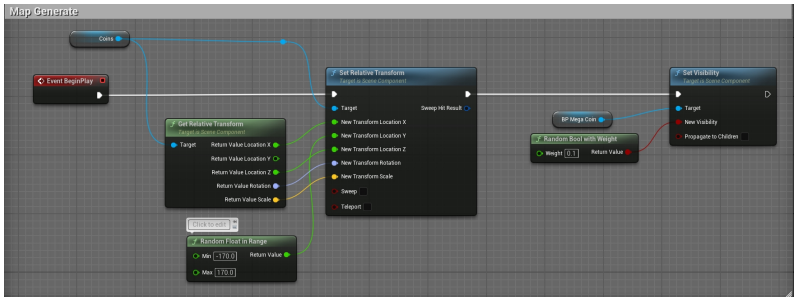


Figure 19: Generation

Unicycle Pawn and movement controller: By default, Unreal Engine provides a subclass of `UPawn` called a character. This character contains a pre-implemented bipedal movement system, which for the standard first person controls used in the maze, chase and red light green light was a suitable base to build their respective pawns over; however, because the movement model for the Unicycle is very different, it was not possible to use the `UCharacter` as a base. To implement the equivalent functionality It was necessary to create a child class derived from the base `UPawn` class instead. The simple floating movement controller built into the engine was utilized to handle acceleration and deceleration; however, custom implementations were still required for gravity and steering based on head movement.

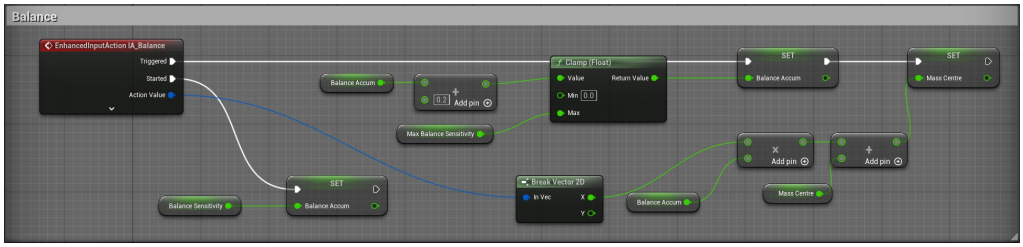


Figure 20: Unicycle Balance

5.5 Performance Optimization

Due to the performance limitations of typical VR devices and the importance of retaining a smooth frame rate, it was extremely important to limit the amount of computationally expensive effects or functionality. There were a number of occasions where the initial design idea ended up causing performance issues and so we had to tweak our design or try and optimise the functionality. Some examples of this were:

Avoid using direct Cast Using `Cast To ...` in Blueprints, especially inside frequently triggered events like `Tick` or `OnComponentBeginOverlap`, can lead to performance issues due to runtime type checking. In our project, direct casts such as `Cast to VRPawn_Maze` were originally used to access player variables, but this proved inefficient in VR. We optimized these by caching references in `BeginPlay`, using Blueprint Interfaces for decoupling, and leveraging Event Dispatchers for communication between actors. These changes helped reduce CPU load and improved frame rate stability.

Dynamic lighting On mobile devices, the use of dynamic lighting can cause dramatic slowdowns and is generally advised against, particularly for the underpowered *Meta Quest 2* devices.

Avoiding particle effects Portals in the game hub that featured particle effects and a glowing texture. These caused the system to drop to an unacceptable frame rate when the project was built to VR, so we had to redesign them to be a more simplistic dark portal without the particle

effects. The redesign is intended to retain the mystique of the original, with a much smaller performance cost.

Postprocessing textures Postprocessing effects like bloom, ambient occlusion, and color grading were reduced or disabled, as they caused frame rate drops on mobile VR devices. Only essential effects were kept to balance visual quality and performance.

Complicated geometry High-poly meshes and complex collision shapes increased GPU and physics load. We optimized these by using simplified meshes, basic collision shapes, and LOD systems to maintain smooth rendering while preserving visual fidelity.

5.6 Technical Challenges

How to Achieve Intuitive UI Interaction: During development, we encountered a key interaction challenge: enabling players to select and click UI buttons using VR controllers. The initial approach involved simulating mouse movement with the joystick, combined with a separate button for clicking. However, this method proved unintuitive and broke the immersive nature of VR interactions. After further research, we discovered that projecting a laser pointer from the player's hand to interact with UI elements provided a more natural VR experience. Fortunately, Unreal Engine 5's default `VRPawn` includes a built-in `Widget Interaction` component that supports laser-based UI interaction. We enabled hit testing and configured the laser to be visible only when hovering over a UI element, improving immersion and clarity.

This led to a new issue: determining which controller button should perform the click action. The triggers, ideal for selection, were already used for turning left and right. Other buttons lacked ergonomic placement for precise UI control. To resolve this, we implemented a context-sensitive input logic—when the UI is active, the trigger is temporarily repurposed for clicking UI elements. Once the UI is closed, the trigger reverts to its original turning function. This approach successfully balanced intuitive UI interaction with limited controller input options.

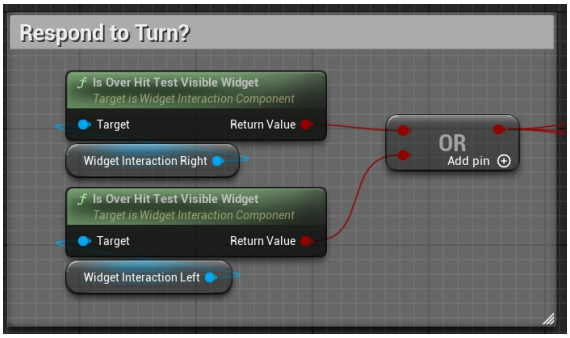


Figure 21: Turn Respond

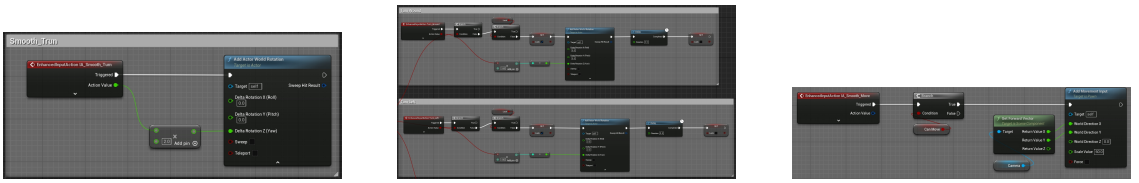
Achieving Comfortable Movement Control with Limited Hardware Input: One of the major technical challenges was determining how to control the player's movement direction under hardware constraints. The MotusVR treadmill only detects forward movement and does not allow directional input. As a result, we experimented with additional input mechanisms to control the player's turning and movement behaviour.

The initial approach involved using the thumbstick on the VR controller to control turning. However, this caused motion sickness due to a mismatch between visual motion and the vestibular system's perception. To mitigate this, we experimented with using the player's head rotation to determine movement direction. Unfortunately, since users were seated, maintaining a rotated head position (e.g., looking left to move left) led to poor usability, especially in maze-like environments where frequent turning is required.

The ultimate solution was to use the left and right triggers on the controller to perform discrete 90-degree turns. When the trigger is pressed, the player's view instantly rotates by 90°, effectively

simulating a quick turn. This approach proved especially effective in environments with right-angle turns, such as mazes. It successfully addressed both the direction control issue and significantly reduced motion sickness, resulting in a more comfortable and immersive experience.

Many in-house testing sessions were undertaken to evaluate these different movement strategies, and so the feedback of many team members - all with differing VR gaming experience - was taken into account before the final controls were settled upon.



(a) Joystick Turn (b) Trigger Turn (c) Head Direction

Figure 22: Different Movement Control

Designing and Implementing 3D Spatial UI for VR Environments: Unlike traditional 2D games, VR experiences require user interfaces that are spatially present and often utilize interactive—floating panels that feel embedded in the 3D world instead of static elements overlaid on the screen. To achieve this, we explored Unreal Engine's `Widget Component`, which allows 2D UI elements to be rendered in 3D space.

However, a limitation quickly became apparent: widgets placed directly in the scene remain fixed in position and do not move with the player, which breaks immersion and usability. To solve this, we created a `Widget Component` within the player's `VRPawn` and attached it to the camera. This ensured the UI stays consistently in front of the player and follows their head movement, maintaining a natural and immersive spatial UI experience.

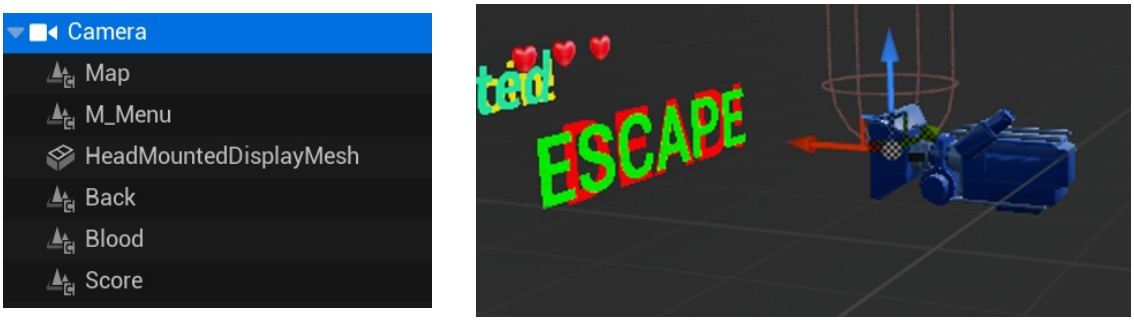


Figure 23: UI and Camera Setting

Version control: Due to the nature of Unreal Engine, where projects consist largely of binary files, using version control effectively proposed a significant challenge to our team. Firstly, it is a common issue with game development projects that the repository size can balloon to unmanageable levels as you start to add content and assets such as textures, 3d-models and other Unreal Engine Asset files. In order to mitigate this, we enabled Git LFS[2], which is a extension to git developed specifically to handle these kind of issues. Because git is not able to effectively diff binary files, "any change to a binary file requires Git to completely replace the file in the repository", the solution is to store these large files elsewhere and just store a plaintext pointer in the repository. This solved the problem of an oversized repository causing the common commands such `clone`, `fetch`, and `pull` to slow down; but it also introduced a new problem: merging to conflicting files is now extremely difficult. Because of the binary nature of the Unreal Engine files, the merging process was both slow and entirely manual - even when changes were made to non-conflicting parts of the blueprint. As a result we ended up adopting a solution by *Social Contract*. This

[2] `https://docs.gitlab.com/topics/git/lfs/`

means that we would agree to try and work on separate parts of the project and agree to merge very frequently back in to the main branch - even if this meant breaking the convention of only merging feature-complete changes into main. This worked effectively for us as a small team with good and frequent communication; however, in a larger organisation or team, a software solution would be essential - for example using the locking feature of gitLFS, or even using a different game development tailored version control system.

Because we started the project with comparatively little experience in working on software in a team, there were some *teething* issues in our approaches to git. At one point, we had two branches that had diverged significantly that both contained important features that needed to be merged. Because manual merging process can take a long period of time, we had to arrange a *feature-freeze* where the group agreed that we would stop developing new features for a week and focus our efforts on merging the existing branches together. This experience taught us the importance of frequent merging and we were able to avoid any similar slowdowns for the remaining duration of the project.

5.7 User Testing

Towards the end of the project, user testing was carried out to evaluate the effectiveness of our prototype with respect to both rehabilitation concerns and the level of engagement and fun that the game provided. Ten participants took part in a short playtest followed by an interview to obtain feedback on their experience. The participants were selected based on their experience with rehab and lower leg injuries in the past and the group was made up of a variety of different levels of experience when it came to VR gaming.

5.7.1 Ethics and Consent

Ethical considerations and informed consent were paramount in carrying out this user study. We aligned our study with the University of Nottingham's research ethics guidelines and the specific requirements in outlined in the Data Management Plan (DMP). Participants were clearly informed about the purpose of the study, nature of their participation, and their rights, including the right to withdraw at any time without the need of providing a reason.

Prior to their participation, all participants were required to provide consent, confirming their voluntary engagement and understanding of the activities involved, including audio and video recordings, questionnaires, and direct observational data. Consent procedures ensured transparency regarding the collection and usage of potentially identifiable information such as names, ages, and video recordings.

All data collected that would be able to identify participants, including names, age and audio recordings, were anonymised through numerical identifiers (e.g., P1, P2). Audio recordings were transcribed and subsequently destroyed. Due to the impracticality of anonymisation, video recordings which were essential for capturing participants VR interaction were securely stored on University servers via Teams which is only accessible to authorised project members outlined in the DMP.

All data was securely stored on University servers using OneDrive, accessible only to authorised team members listed in the DMP and any changes in team members would result in immediate revocation of access for individuals no longer part of the team.

Additionally, data sharing with third-party entities, in our case specifically Motus VR, will be restricted to anonymised datasets only, maintaining adherence to ethical standards and data protection laws.

Throughout the study continuous oversight was maintained to ensure compliance with ethical guidelines. These measures collectively ensured ethical integrity, participant rights, and ethical data management throughout the research lifecycle.

5.7.2 Structure of test sessions

Firstly, the purpose and aims of project would be explained to the participant, after which their level of familiarity with VR was assessed before being giving a brief introduction to the hardware

- both the Motus Explore Treadmill and the Meta Quest 2 device. Once the participants were comfortable with the hardware, we allowed them to explore the hub world, getting used to the controls and sensation of *walking* on the treadmill in a virtual world. The participants were then guided through the minigames in a particular order: starting with the most simple - "red light green light" - to get further accustomed to the controls, and then moving on to the more complicated maze and unicycle games. Video footage was recorded throughout the play sessions to allow for further analysis down the line. Finally, after the play testing was complete, a short interview was conducted (this was also recorded). The participants were asked the following questions:

1. What did you enjoy or think went well during the experience?
2. Was there anything confusing or difficult to use
3. How did you feel while playing the game - for example: engaged, bored or motivated?
4. Have you had any rehab experience. If so do you feel the game would be relevant or helpful for rehab purposes?
5. If you could improve one thing, what would it be?
6. Would you use this game regularly if it was part of your recovery plan? Why or why not?
7. Do you have any final thoughts or comments?

5.7.3 Summary of results

Play-testing observations Simply watching players who had never played our system before provided an invaluable experience to assess where the pain points in the game were - some of which were obvious even before the interview. Overall we felt that players were quick to pick up the controls in the hub world and simple *Red light green light* games, but took longer to get used to the more complicated games. Users had difficulty at the start of the maze game due to the fact that the *zombie* was immediately chasing after you. We observed that all participants seemed engaged throughout the testing experience and only one participant mentioned feeling slight motion sickness, and only one participant mentioning pain during the use of the device (due to an ongoing knee injury).

Interview Responses The responses to the interview questions laid out in section 5.7.2 have been summarised:

1. **Positives:** There were a variety of responses to this question. Some comments included:
   - Enjoying the interactivity and realism that the system provided.
   - Enjoyed the experience of being in the hub world
   - Some users enjoyed the controls of the unicycle game - citing that it felt more realistic to control
   - Other users mentioned the maze game and the excitement of zombies chasing and the puzzle aspect
   - One user mentioned they enjoyed that the game made them "work up a sweat".
2. **Confusing or difficult aspects:** The responses to this question were more unified. Many users felt that the menus were difficult to use and understand, and a number of users also felt that the 90°turn controls in the maze game felt confusing or unintuitive. Despite this, many of these users also found the controls to be much more manageable after being advised to not move their head as much.

3. **Engagement:** All users of the study said that they were engaged during the play test, with a variety of reasoning, including that they found it generally fun, that they enjoyed the interactive and realistic feelings, and that they enjoyed the excitement and jump scares in the maze game. One user mentioned that they would have preferred the unicycle to have a more clear objective as opposed to just being endless mode.

4. **Suitability for rehab:** A number of participants were very positive, for example citing that "the sliding movement worked my calf muscles" and thought that the stretch and release pattern of this movement was helpful for recovery. Some participants felt like the device didn't offer a wide enough range of motion, for example they noted that it didn't provide opportunity for flexion in the same way that a normal walking motion on a treadmill would.

5. **Improvements:** There were many specific and useful comments on this question, some of which include:

   - Improvement in graphics and textures
   - Improvement in the feel of the menu
   - Adding more levels of difficulty
   - Adding more fast paced gameplay (e.g. more boosters in Unicycle game)
   - Creating better responsiveness to the speed of the feet movement

6. **Would you use in your rehab program:** This question gained the largest amount of positive response. A large number of participants cited regular rehabilitation programs as "boring" and that "it can be a chore", and said that in comparison our system was more fun and engaging. Only one participant said that they would only be likely to use it if it was developed to have more levels and more scaleability.

Conclusions The feedback from the user study was overwhelmingly positive, especially towards the gaming experience itself. The study also gave us crucial insights into which areas aren't working well at the moment, in order to better understand these requests, the complaints have been grouped into three categories: The polish of the game, referencing comments on the graphics as well as the slightly buggy menus; hardware issues, mainly pertaining to the questions on the suitability of the device for rehab purposes; and game play issues. Of these categories the one which we are most able to put into action are the game play issues, we were able to make simple tweaks to the game in response to common complaints. Some examples of simple tweaks were adding more powerups in the unicycle game and giving the player a grace period in the maze game.

5.8 Future Development

The current version of the project presents a functional prototype built in Unreal Engine 5 using Blueprints. While the core gameplay and rehabilitation mechanisms have been validated through user testing, several architectural improvements and feature expansions are planned but not yet implemented due to limited development time and the priority of delivering a stable demo for evaluation.

Additionally, user testing was completed relatively late in the development cycle. As a result, some of the valuable user feedback could not be incorporated into the current version; despite this, the feedback has allowed to compile a clear user-driven roadmap for potential future developers of this project. Furthermore, due to limited experience in 3D modelling and design optimization, certain visual and performance aspects of the game remain unpolished. Finally, hardware limitations — such as the lack of high-performance VR devices and more sophisticated treadmill hardware for example with footstep recognition and gait analysis — restricted exploration of advanced locomotion methods such as natural walking.

Based on our experience developing the game and the feedback from our users, we imagine future development will focus on two main areas which will support long-term extensibility for research and clinical applications.:

1. Architectural refactoring and modularization to improve maintainability and scalability
2. The addition of new features to enhance immersion, usability, and rehabilitation tracking.

5.8.1 Architectural Refactoring

The project is currently developed using Unreal Engine 5 Blueprints, which facilitates rapid prototyping but requires structure for long-term scalability. The following architectural practices are suggested to support future extension:

- **Dedicated Folder Structure per Mini-game:** Each mini-game should reside in its own folder under the `/Games` directory. Inside each game folder, include:
  - `/Blueprints` – all logic-specific Blueprints (e.g., GameMode, PlayerPawn).
  - `/Assets` – static meshes, textures, and audio used specifically for this mini-game.
  - `/Animations` – animation sequences and blendspaces.
  - `/UI` – widgets, HUD elements, and interface Blueprints.
  - `/Data` – any DataTables, configuration files, or tuning data.
- **Class Inheritance Structure:** Introduce a base class hierarchy to simplify behavior reuse. For instance:
  - All mini-game `VRPawn`s inherit from a generic `BaseVRPawn`.
  - Shared functionality (movement, camera control, tracking input) is handled in the base class.
- **Modular Blueprint Structure:** Refactor gameplay logic into reusable modules (movement, tracking, UI, etc.) to ensure separation of concerns and ease of updates.
- **Resource Cleanup:** Periodically audit and remove unused assets from the project to reduce clutter, save disk space, and improve loading performance.
- **Consistent Naming Conventions:** Apply standard naming schemes across folders, Blueprints, and variables to improve clarity and support collaborative development.
- **Data-Driven Design:** Externalize tunable values (e.g., speed, scoring thresholds) via Blueprint Data Assets or DataTables to allow non-programmers to tweak gameplay.
- **Event-driven Component Interaction:** Use Blueprint Interfaces and Event Dispatchers to create decoupled communication between actors and components.

5.8.2 Potential New Features

Feature	Description
Reward Systems	Include motivational incentives such as progress badges, trophies, and performance milestones to encourage consistent engagement during rehabilitation exercises.
Rest and Pause Mechanisms	Incorporate clear and user-friendly pause and rest functions, enabling users to take breaks comfortably. This is especially valuable for those undergoing physical therapy.
Customisable Visuals	Provide options to adjust text size, color contrast, and interface simplicity to accommodate users with visual impairments or cognitive processing difficulties.
More Natural and Flexible Controls	Replace or complement snap 90° turning with joystick or smooth-turn options. Allow adjustable sensitivity for foot or head movement.
Expanded Gameplay Mechanics	Add more in-game mechanics and layers of interaction to prevent repetition and maintain user interest, such as stretch-release or heel-lift movements.
Difficulty Scaling	Introduce levels of difficulty or progression systems that adapt to users' rehabilitation stages and physical capabilities.
Immersive Tracking Features	Enhance hand and head tracking fidelity to increase immersion and physical responsiveness.
Time-based Goals and Leaderboards	Include timers, speed-based rewards, and high score boards to introduce friendly competition and performance tracking.
Boost Elements and Game Flow Enhancers	Implement dynamic speed boosters (e.g., auto-triggered accelerators) to intensify pacing and engagement.
Immersive Audio Experience	Incorporate vivid and adaptive background music to enhance user immersion and emotional engagement.
Rehabilitation Analytics Integration	Work with medical advisors to track step frequency and movement data to generate progress records and insights.
Bug Fixes and UX Improvements	Address issues such as confusing layouts (e.g., maze printers) and reward coin glitches to improve satisfaction.
Directional Treadmill Integration	Integrate hardware that can detect footstep direction to improve locomotion control in future iterations.
More 3D Models	Multiple 3D models have already been created but due to time constraints and some changes during development, they weren't used. Many of these models could be re-used at a later date.

Table 7: Proposed Features and Enhancements for Rehabilitation-focused VR Gameplay

6 Project Management

6.1 Project Planning & Methodology

6.1.1 Methodology

An Agile Scrum methodology underpinned all planning activities, with one-week to two-week sprints enabling rapid iteration and continuous supervisor feedback. A high-level project timeline was first established in a Gantt chart (Figure 24), setting key milestones such as requirements sign-off, prototype completion, and final delivery. This visual roadmap ensured that the team and supervisor maintained a shared understanding of overarching objectives and deadlines.

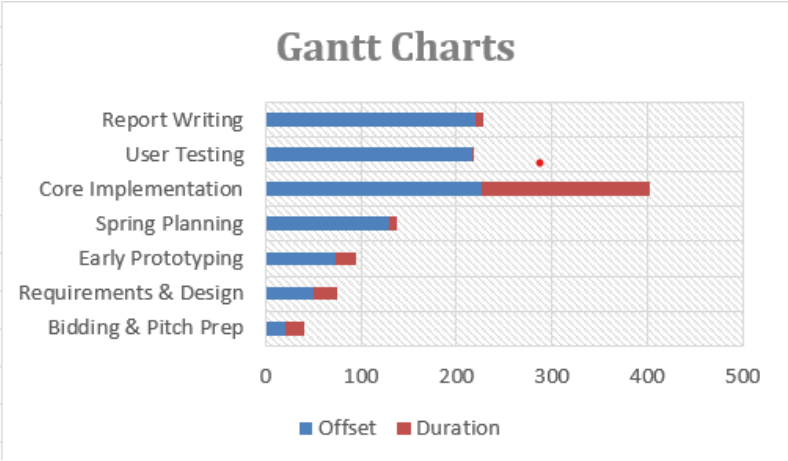


Figure 24: Gantt Charts

At the beginning of each sprint, the Mini-Game Group Leader refined the product backlog in Jira—prioritising tasks according to the INVEST criteria—and circulated the updated backlog via Discord before the Sprint Planning meeting (Figure 25). During the Sprint Planning meeting, each member reported completed work, upcoming tasks, and any impediments.

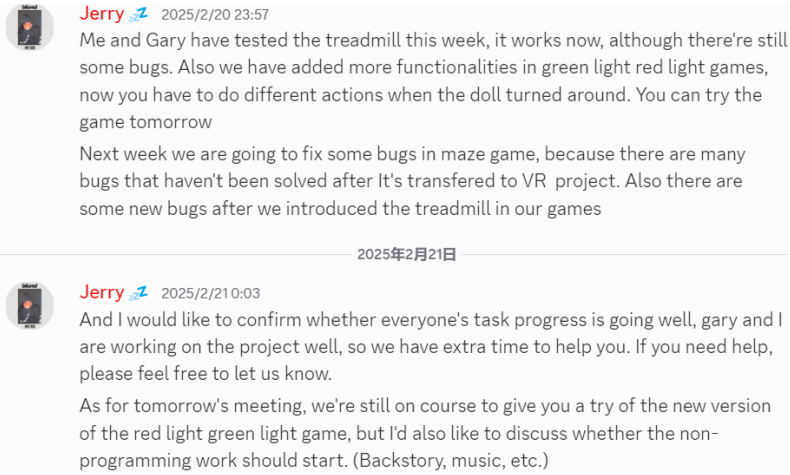


Figure 25: Backlog

At the end of the weekday, a Sprint Review demonstrated implemented features to the academic supervisor, gathering direct feedback to refine subsequent sprints. This was immediately followed by a Sprint Retrospective, documented in the team's GitLab Report (Figure 26). Action items were recorded and carried into the next planning cycle.

The adoption of Agile Scrum—with its time-boxed sprints, structured ceremonies and real-time backlog management—provided a disciplined yet flexible framework for planning. By combining visual diagrams, clearly defined roles and ongoing feedback, the team maintained steady momentum, swiftly addressed challenges and established a solid foundation for the subsequent phases of implementation.

6.1.2 Project management tools

A suite of integrated tools was employed to support synchronous and asynchronous collaboration, version control, task tracking and continuous integration:

- **Microsoft Teams:** All formal online ceremonies—sprint planning, sprint reviews and retrospectives—were conducted via dedicated Teams channels. Meeting invitations, agendas

Meeting

Metting Order	Format	Date	Particapant	Summary	Result
1	In-Person	2024.09.27	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Meet whole team	assign team roles
2	In-Person	2024.9.30	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Discuss the optional projects	Determine the project choices and future plan
3	In-Person	2024.10.02	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Discuss some general ideas for pitches	Determine the division of labour
4	In-Person	2024.10.07	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Teamwork Activity	None
5	In-Person	2024.10.09	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang Dylan Aldridge Gonzalez	review and suggest any improvements to CV	Determine the division of labour in EOI and Pitches
6	In-Person	2024.10.12	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Review EOI together and start Pitch Videos	Finalized the EOI
7	In-Person	2024.10.14	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang	Record Pitch Videos	Record Pitch Videos
8	In-Person	2024.10.16	Brendan Cheng Phileas Tang Chi Ung Jirui Zhang Samuel Toth Gang Yang Miles Butt	Finalized Pitch Videos and EOI	Submit Final EOI and Pitch Videos
9	In-Person	2024.10.17	Brendan Cheng Phileas Tang Chi Ung Samuel Toth Gang Yang Miles Butt Dylan Aldridge Gonzalez	Discuss for the coming Pitch Day	Prepared for the Pitch Day
10	In-Person	2024.10.24	Phileas Tang Chi Ung Samuel Toth Miles Butt Dylan Aldridge Gonzalez	Discuss for the initial Task	Make an initial plan
11	In-Person	2024.10.30	Jirui Zhang Chi Ung Samuel Toth Gang Yang	Clarify the tasks	add some issues in GitLab
12	In-Person	2024.10.31	Jirui Zhang Samuel Toth Gang Yang Miles Butt Dylan Aldridge Gonzalez Brendan Cheng	Discussed game ideas	Set up Jira and assigned a few tasks
13	Online	2024.11.4	Jirui Zhang Samuel Toth Gang Yang Miles Butt Phileas Tang Chi Ung	Discussed Tasks	Determine the division of labour
14	In-Person	11.6	Jirui Zhang Samuel Toth Gang Yang	Discuss with the project	Decide to divide our group in sub

Figure 26: Meetings Record

and minutes were stored in Teams, ensuring a centralised record. All key project artefacts—including meeting minutes, requirements documents, design specifications and reports—are maintained in the "Shared Documents" library within Microsoft Teams, ensuring centralised access and version control.

- **Discord:** A private Discord server provided an informal workspace for off-hours communication and rapid technical discussions. Separate channels (e.g. #Unicycle, #maze, #report-stuff) helped organise queries, share code snippets or screenshots, and coordinate ad-hoc pair-programming sessions (Figure 27).

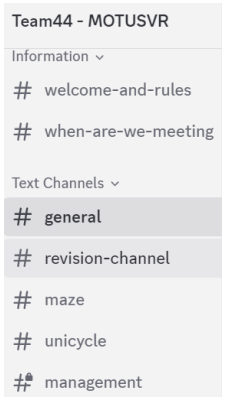


Figure 27: Discord Channel

- **GitLab:** Source code was managed in GitLab with a main branch supported by integration and feature branches. Merge Requests (MRs) were required for all changes, with at least one peer review before integration (Figure 28).

Managing an Unreal Engine project in GitLab presented two main challenges: repository bloat and binary merge conflicts. For a detailed account of the version control usage and issues we faced see our discussion in section 5.6.

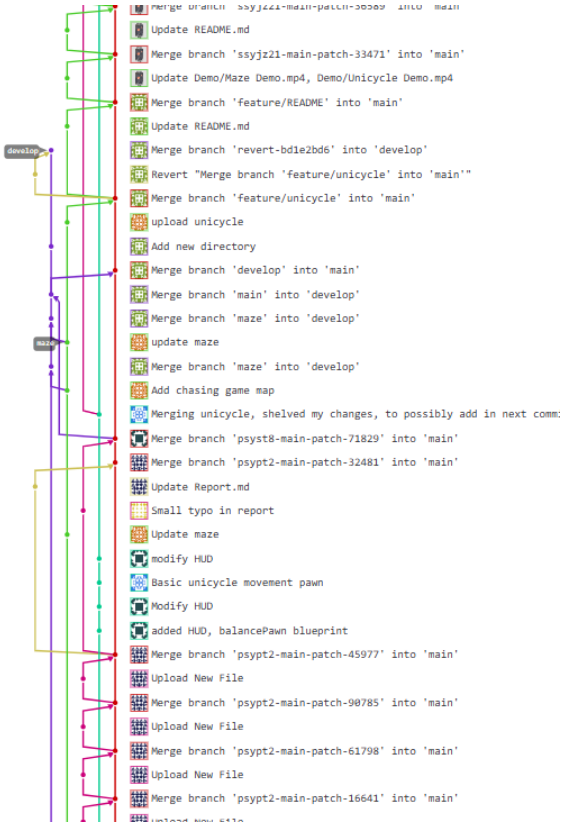


Figure 28: GitLab Repository Graph

- **Jira** At beginning, Tasks and bugs were tracked in Jira, organised into Epics corresponding to each mini-game module. Sprints were defined in a Scrum board, with columns for "To Do", "In Progress", "In Review" and "Done". Story points were estimated during backlog refinement sessions, and a burndown chart was used to monitor sprint progress. Jira issues were linked to GitLab commits and MRs, providing end-to-end traceability from requirement to code.

  However, we found Jira to have a sharp learning curve and to be better suited to larger teams in more traditional software engineering projects. Ultimately, we moved to a more ad-hoc strategy - especially due to the fact that the team of implementers had shrunk to three. This is one aspect that we would hope to improve given a similar task in the future.

- **Overleaf** The main group report and accompanying documentation were authored collaboratively on Overleaf, a cloud-based LaTeX editor. An Overleaf project was created and shared with all team members, granting edit access and enabling real-time co–editing. Section ownership was assigned—each member was responsible for drafting specific chapters directly within Overleaf—while the built-in version history and "Track Changes" comments provided a clear audit trail of edits. Overleaf's integrated PDF preview allowed immediate verification of formatting and layout, and its commenting tools facilitated inline discussion of revisions. The final PDF export from Overleaf was used for submission, ensuring that the report remained consistent with the LaTeX source.

Integrations GitLab–Jira: Commits that referenced Jira issue keys automatically transitioned the issue status in Jira, reducing manual updates.

6.2 Communication & Collaboration

Team Skills & Task Allocation: To support effective collaboration, tasks were allocated according to each member's expertise and tracked in Jira. Gang Yang, Jirui Zhang and Samuel Toth—our most experienced coders—were assigned core development duties; Chi Ung, as Git leader, managed version control and porting the project to standalone VR devices; Phileas Tang, with prior 3D-modelling experience, created essential asset models; and Brendan Cheng composed the game's music and soundscape. Dylan Aldridge Gonzalez and Miles Butt handled ad-hoc tasks, with Dylan focusing on background research and documentation and Miles specialising in ethics and risk analysis. This skill-driven assignment was confirmed during Sprint Planning and reviewed in daily stand-ups, ensuring that each responsibility aligned with individual strengths and that communication remained clear and focused. (For detailed task allocation, please see the Appendix A).

Knowledge Sharing: Within the first few weeks it became quite apparent that there were major limitation that needed to be overcome. Namely, within the team, experience with Unreal Engine, the engine that was chosen for this project, was scarce. Unreal Engine is a behemoth of complexity, and in order to make progress a sideways step was needed and individual attempts to create learning projects was necessary. Knowledge and techniques learnt during the week was shared. See Figure 29 for a table of these learning outcomes.

Unreal

name	Research	Relevant link
Jirui Zhang	How to implement a vehicle in Unreal	https://dev.epicgames.com/documentation/zh-cn/unreal-engine/how-to-set-up-vehicles-in-unreal-engine
Brendan Cheng	Inter-Component communication, collision boxes	
Phileas Tang	Hit registering, Progress bars, Widgets	
Chi Ung	Enemy Behaviour trees, Decision making, AI "Vision"	
Samuel Toth	Understanding the unreal C++ framework	research document
Gang Yang	1.Arrangement of objects on unreal, creat a romm. 2.Creat a player controler to control the movement of character including move forword, back, left, right, jump. And view control including look up, down, left, right.	
Dylan Aldridge Gonzalez		
Miles Butt	Colour schemes, user interface, general design	

Figure 29: Individual Learning projects

Team Organisation & Task Delegation: After this point, requirements felt far easier to decompose into manageable tasks, and this is when the first official sprint had begun. The team was broken down into two sub-teams in order to maximise efficiency. As the nature of the design is based on a collection of small mini-games, it was easy to organise this process. Two simple mini-games were agreed upon, a maze game and a unicycling experience was designed with the workforce split proportionally to work on these games. Within each team, further specialised roles were assigned. For example, in the unicycle team, Phileas Tang worked on asset development – including crucially the creation of a unicycle 3D model; Samuel Toth on the development of the unicycle controller blueprint, and Chi Ung developing a user interface. With these roles reflect the existing skillsets, delegation and the planning of tasks within these smaller groups was far smoother than before.

Pair-Programming Sessions Formal pair-programming sessions were introduced to accelerate skill transfer and collaboratively solve complex problems. Two such sessions were held each sprint conducted over the in-person meeting.

Sessions focused on tasks such as Blueprint construction, UI widget integration and merge-conflict resolution. The team adopted the driver-navigator pattern: the "driver" controlled the Unreal Editor, implementing code or Blueprints, while the "navigator" reviewed changes in real time, suggested optimisations and checked adherence to style conventions. At the end of each session, pair outcomes—including code snippets, workflow improvements and any emergent issues—were summarised in the GitLab Wiki, creating a living record of technical solutions and lessons learned.

This practice not only reduced the frequency and severity of merge conflicts but also improved overall code quality,and fostered shared ownership of modules.

6.3 Risk Analysis & Management

A risk analysis was carried out to understand the risks associated with team work and develop strategies to deal with this if problems ever arose.

Stage	**Risks**	**Mitigation Strategy and Role**
Forming	1st risk: Poor communication	As a facilitator, encourage open communication by creating a welcoming environment where everyone feels safe sharing their thoughts.
	2nd risk: Unclear character and advantages of each team member	As an initiator, initiate communication about team members' character, strengths and interests. Lead group members to share more about themselves.
Storming	1st risk: Keeping conflict	As a mediator, promote tolerance and patience to each other, analyse the problem rationally and actively facilitate conflict resolution by encouraging members to address issues respectfully.
	2nd risk: Not following team rules	As a rules advocate, take the lead in following the rules, emphasizing the contribution of the rules to the team's success. When someone breaks the rules, ask the reason and remind the person to follow the rules.
Norming	1st risk: Group thinking, lack of personal ideas	As a new ideas' supporter, put forward some new ideas to the project and remind the team that diverse ideas strengthen our solutions. Encourage the team to voice their opinions and express constructive ideas.
	2nd risk: Unequal distribution of responsibility	As a collaborator, help team members with overburdened tasks, discuss with team members for a secondary task allocation. Allocate tasks according to ability and amount of work and clarify the specific responsibilities of each member.
Performing	1st risk: Overfatigue	As a supporter, promote a healthy work-life balance, encouraging breaks and time-off when necessary. Propose group building activities (e.g., picnic, hiking) to relieve team members' stress.
	2nd risk: Difficulty adapting to changes	As an adaptation facilitator, take the lead in piloting new processes. Design flexible adaptation measures, communicate the reasons for change and expected effects to team members. Analyse the costs and possible risks before starting tasks.

Table 8: Stages of Group Development: Risks and Mitigation Strategies

7 Reflection

7.1 Successes of the Project

Individually, many members of the team have made great strides in their understanding of game development and VR technology; starting with either very little or no experience in game development, and by the ending of the project feeling familiar and confident with a complicated systems such as Unreal Engine or Blender. As a group, we have navigated the challenges that come with working in a large team. It has not always been easy, and conflicts have sometimes arisen but we were able to use foundational work such as our manifesto and risk management document to mitigate the potential negative outcomes.

The success of the project is reflected in the demo that we have created and also in the many positive comments received in user testing. In particular we are pleased to have to implemented a game that stays true to our original design aspirations, meeting many of the functional and non-functional requirements as laid out in the analysis section of the report.

7.2 Choices in technology

At the beginning of the project, many choices had to be made around which technology and techniques we would make use of. Now that the project has reached it's conclusion, and the team has collectively gained a much greater understanding of VR and 3D game development, we will reflect on our choices of technology.

Unreal Engine As discussed in section 5.1, we considered the two major 3D game engines, Unreal Engine (UE) and Unity, and ultimately settled on Unreal Engine. One of the major deciding factors was the suitability and capability to create VR games, and UE held up to it's reputation in this regard - with useful presets and inbuilt demos we could build on. Despite this success, we still encountered some difficulties with this choice of technology. Firstly, Unreal Engine is a very large and very complicated bit of software, with countless subsystems, editors and workflows, this meant that the learning curve was extremely large as many members of the team were completely new and others had just a small amount of experience with the software. A large part of the first term was spent getting comfortable with using the software, other game engines, for example, Unity are considered to be slightly more user friendly, as this may well have allowed us to move more quickly at the initial stages of the project. Another benefit of Unity may also have been that it's use of C# as a scripting language may have been a good compromise between the rapid prototyping available in Blueprints and the superior organisation and maintainability of C++.

Blueprints vs code Another choice that was made early on in the process was the decision to exclusively use Unreal Engine's proprietary visual scripting language *Blueprints* or a hybrid approach combining C++ and Blueprints. You can find the result of research into the tradeoff of both approaches in section 5.1, and after trying to take the hybrid approach initially, we ended up choosing to go exclusively Blueprints. There were many benefits initially, most importantly, allowing us to get more members up and running quickly and rapidly develop prototypes; despite this, as the year went on, we found that the Blueprints made designing a maintainable and well organised code base particularly challenging. In addition, the ability to have full version control integration, with automatic diffing and merging would have been extremely valuable as the pace of development necessarily sped up towards the end of the project.

7.3 Logistical Choices

Issues we faced One significant issue we faced was in the difference of performance capability between the VR stand-alone and running via using meta quest link via a computer. Whilst we knew that we needed to pay attention to performance, we didn't realise how limiting the VR headsets would be. The process of testing performance changes was quite slow because we needed

to package the game to VR each time to test the true performance on stand-alone VR. Additionally, optimisation was yet another area that we weren't experienced in.

Organisational Difficulties Due to the unique nature of the project, allocation of our tasks proved to be challenging. Having no prior experience developing games or using any sort of game engine led to difficulties in segmenting tasks into easily assignable subtasks, often finding that it was difficult to have enough tasks to assign or to have tasks that would feasibly be completable within a week. Additionally, a lack of understanding of UE5 and game development concepts led to difficulty completing certain tasks or trial and error development to find optimal designs.

Agile methodology While we had initially planned to make use of a SCRUM framework, with weekly meetings and sprints to be tracked on Jira, we eventually found that the variable lengths of time required to complete tasks would lead to a lot of rollover between sprints. As a result a more relaxed approach was taken, still adhering to weekly meetings but focusing on current progress and whether help was required/if there was anything new to do rather than trying to adhere to a strict weekly sprint model.

Finding Niches Despite the organisational difficulties, the group seemed to naturally take on different responsibilities that would allow for us to work semi-independently on tasks more suited to our personal strengths, thereby mitigating the issue of having too many hands per task. Because the project involved more than simply coding, there were many other aspects of the project that could be taken on by each member including graphic design, modelling, music and more.

8 Conclusion

This report documents the implementation processes and decision making procedures that have taken us from an initial design idea arrived at through a thorough requirements analysis all the way to a prototype who's success was highlighted by the extremely positive response in user testing, conducted with participants who have experienced the challenges involved in the traditional rehabilitation process. Feedback from these participants also provided valuable insight into potential directions for future development, based on which a roadmap for future development has been created. In conclusion, over the course of the project, the team has managed to create an engaging prototype for a recovery-focussed VR game with novel game mechanics and thoughtful human-centric design choices.

A Task Allocation

- **Brendan Cheng**
  - Food Preference App EOI and pitch video
  - Main point of communication with project supervisor
  - Organise and direct meetings, producing meeting summaries to track progress
  - Produced music for the game
  - Ran user testing, conducting post-test interviews and analysis
  - Wrote Component Design User Input, Game Logic Core Movement/Difficulty, Music, Logistical Choices in main report
  - Wrote Game Controls, Navigating the Game, Minigames Instructions, Hardware Setup, Contact Details for User Manual
  - Helping write script for the demo video
- **Phileas Tang**
  - MOTUS VR EOI and pitch video
  - Maintain a log of meetings
  - Research on lower limb rehabilitation
  - Helped write project background and understanding
  - Created the unicycle model and its animation
  - Created various different models that weren't used in the final product
  - Written the safety and comfort section of the user manual
  - Written the abstract
  - Recorded lines for the demo video
  - Wrote finalised script for the demo video
  - Did the bidding for the projects
  - Helped edit the demo video
- **Chi Ung**
  - Motus VR EOI and pitch video
  - Made the Unicycle demo for the pitch video
  - Wrote initial draft of the VR game research and device under Market research in the interim report
  - Optimisation and packaging for the android platform
  - Wrote system requirements and installation guide in the software manual and a few minor sections in the user manual
  - Helped write the finalised script for the demo video
  - Edited the demo video
- **Samuel Toth**
  - Augmented Telepresence project EOI and pitch video
  - Recording food preferences app pitch video
  - Developing the Unicycle prototype for the interim report
  - Writing the Intro, Project Management & progress and unicycle-related parts of the interim report
  - Maintaining the git repository and assisting members in use of git.

- Development of the final incarnation of the Unicycle mini-game in the second semester
- Helped with user testing - including writing the report section on user testing
- Writing and editing final report, in particular the implementation and reflection sections - and all sections pertaining to the unicycle game.
- Writing Unicycle parts of the software manual.
- Helping write the script for the demo video.

- **Jirui Zhang**
  - Augmented Telepresence Projet EOI and Pitch Video with Samuel Toth.
  - Edit EOI of the other two projects.
  - Make a second-version PPT of VR Game Project.
  - User Story and Persona.
  - Implementation of the maze and chasing game, including coins picking, AI enemies, player health system, automatic map generation.
  - Write a report of the group in Git, record the team meeting in first semester.
  - Rewrite and extend the Project background and understanding, Requirements and critical Analysis Project Part, and write the initial software implementation, Maze-Game related part of interim report.
  - Record Meeting in the first semester.
  - Allocate the task and organise the mini-group of Maze, Chase, red light green light games.
  - Implemented Red light, green light game with Gang Yang.
  - Implemented the Game Hub with Gang Yang
  - Write Introduction, Project background and understanding, Requirements and Critical Analysis, Game Design and Project Management part in main report.
  - Doing research for Project background and understanding, Requirements and Critical Analysis, draw the system architecture diagram for the report.
  - Write the Structure of the whole report, User manual, Software Manual

- **Gang Yang**
  - Writing EOI, creating PowerPoint, record video of Food Preference App
  - Writing the Preliminary Reflection part and the Chasing Game-related part of the interim report.
  - Implementation of the game hub, maze game, chasing game, red light green light game with Jirui Zhang, including sensor light, crystal picking, maze prototype construction.
  - Implementation of converting a first-person project to a VR project.
  - Realizing movement control.
  - Realizing UI design and UI event response.
  - Debugging of treadmill, getting it working on UE5.
  - Helping with user testing.
  - Helping with Demo Video.
  - Writing the implementation part of final report and software manual.

- **Miles Butt**
  - Research for the pitch into Motus VR and lower limb injuries
  - Presentation and research into the User Interface

- 3 Prototype map designs for the 'Chase Game'.
- Final Map design and Implementation for the 'Chase Game'.
- Writing the Requirements & Critical Analysis part of the interim report.
- Completed Standard Operating Procedure (SOP) Form
- Data Management Plan
- Oversight of the User Study, including collaborating in writing the User Study section of the report
- Sourced Individuals and Organised Timeslots for the User Study
- Created the survey and interview questions, ensuring they follow GDPR guidelines
- Wrote the Outline/ Introduction part of the Final Report
- Research and writing the Project background and Understanding Section of the Final Report, namely the following sections: 2.1, 2.2, 2.3
- Collaborated in writing part of the Requirements and Critical Analysis section,

- **Dylan Aldridge Gonzalez**
  - Food Preference App EOI and pitch preparation
  - Wrote the Conclusion of the Interim Report
  - Wrote a Research Report on Lower Limb Rehabilitation best practices
  - Wrote a Research Report on the growth and trends of the VR industry
  - Wrote a Market Analysis Report and Competitor Analysis
  - Reviewed and Edited Final Report
  - Created a Website for Demo Day

B Manifesto

Meetings

- Weekly meetings to ensure that tasks are on track and that risks are identified in a timely manner, discuss contingency plans, and avoidance strategies.
- Frequent Stand-ups when necessary to quickly check progress.
- Before/After each sprint, a retrospective will be held to summarize the previous sprint, reflect on how to become more effective and make a detailed plan for the next sprint.
- Team Members must attend all meetings. If unavailable, they must inform the coordinator at least 24 hours in advance (emergencies aside)

Communication

- Recognising conflict is inevitable, we shall transform relationship conflicts into task conflicts by refocusing on shared goals, creating safe spaces for discussion, and considering multiple perspectives.
- Communicate with Project supervisors frequently, to make sure that the project meets their requirements and to be the first to know when the requirements change.
- All important information, such as meeting summaries and project progress, should be reported in both Teams and GitLab.
- Use a consensus-driven approach to decision-making, ensuring all voices are valued.

Task

- Assigned work should respect a member's time and skillset alongside the relative contribution of other members.
- Adopting SMART principles to set tasks.
- Before starting tasks, understand the requirements, determine their goals and priorities, analyse the costs and possible risks.
- Tasks should be completed before the assigned deadline. If completion is not realistic then the coordinator should be informed prior to the deadline.

Code

- Ensure all code is concise and compliant with best practices to ensure readability, this includes proper formatting, comments and modular design.
- Adhere to a uniform coding standard to ensure consistency across all contributions, enabling better collaboration and maintenance.
- Adhere to modern best practices for software development, including good use of version control as well as continuous integration and testing.

Others

- All issues not covered by this manifesto should be resolved in a unified manner after discussion by all members and then added into this manifesto.

References

[1] D. Adams et al. "Current concepts for anterior cruciate ligament reconstruction: a criterion-based rehabilitation progression". In: *Journal of Orthopaedic & Sports Physical Therapy* 42.7 (July 2012). Epub 2012 Mar 8, pp. 601–614. DOI: 10.2519/jospt.2012.3871.

[2] Anthony Adesanmi. *Rebuilding mobility: A guide to lower limb rehabilitation*. 2024. URL: https://www.topdoctors.co.uk/medical-articles/rebuilding-mobility-a-guide-to-lower-limb-rehabilitation#:~:text=In%20the%20initial%20phase%2C%20the%20focus%20is%20on,ice%20therapy%20to%20alleviate%20discomfort%20and%20promote%20healing..

[3] Michael J. Bosse et al. "An Analysis of Outcomes of Reconstruction or Amputation of Leg-Threatening Injuries". In: *New England Journal of Medicine* 347.24 (Dec. 2002), pp. 1924–1931. DOI: 10.1056/NEJMoa012604. URL: https://www.nejm.org/doi/full/10.1056/NEJMoa012604.

[4] L. Cheng et al. *Effects of Virtual Reality Rehabilitation Training on Gait and Balance*. 2019.

[5] Data Bridge Market Research. *Global Virtual Reality (VR) Healthcare Market—Industry Trends and Forecast to 2029*. 2022.

[6] Vera Fung et al. "Use of Nintendo Wii Fit™ in the rehabilitation of outpatients following total knee replacement: a preliminary randomised controlled trial". In: *Physiotherapy* 98.3 (2012). Special Issue on Advancing Technology including papers from WCPT, pp. 183–188. ISSN: 0031-9406. DOI: https://doi.org/10.1016/j.physio.2012.04.001. URL: https://www.sciencedirect.com/science/article/pii/S003194061200048X.

[7] Georgiev, D. D., et al. *Virtual Reality for Neurorehabilitation and Cognitive Enhancement*. 2021.

[8] S. van Grinsven et al. "Evidence-based rehabilitation following anterior cruciate ligament reconstruction". In: *Knee Surgery, Sports Traumatology, Arthroscopy* 18.8 (Aug. 2010). Epub 2010 Jan 13, pp. 1128–1144. DOI: 10.1007/s00167-009-1027-2.

[9] HelloNote. *Virtual Reality Therapy: Transforming Rehabilitation and Patient Care*. 2024. URL: https://hellonote.com/virtual-reality-therapy-rehabilitation-patient-care/#:~:text=Patients%20wear%20VR%20headsets%2C%20immersing%20them%20in%20digital,promoting%20recovery%20in%20a%20safe%20and%20controlled%20setting..

[10] Brianna Hodge. *AI-Powered Virtual Reality:Next Frontier for Rehabilitation*. 2025. URL: https://neurorehabvr.com/blog/ai-integrated-with-vr#:~:text=Neuro%20Rehab%20VR%20stands%20as%20a%20leader%20in,automate%20and%20enhance%20the%20administrative%20side%20of%20therapy..

[11] HOLOFIT. *HOLOFIT VR FITNESS*. 2025. URL: https://www.holodia.com.

[12] Bita Imam et al. "A clinical survey about commercial games in lower limb prosthetic rehabilitation". In: *Prosthetics and Orthotics International* 42.3 (2018). PMID: 29126375, pp. 311–317. DOI: 10.1177/0309364617740238. eprint: https://doi.org/10.1177/0309364617740238. URL: https://doi.org/10.1177/0309364617740238.

[13] L. M. Kruse, B. Gray, and R. W. Wright. "Rehabilitation after anterior cruciate ligament reconstruction: a systematic review". In: *The Journal of Bone and Joint Surgery* 94.19 (Oct. 2012), pp. 1737–1748. DOI: 10.2106/JBJS.K.01246.

[14] McKinsey Company. *Telehealth: A quarter-trillion-dollar post-COVID-19 reality*. 2021.

[15] MindMaze. *MindMotion® PRO*. 2025. URL: https://mindmaze.com/mindmotion-pro/.

[16] UHDB NHS Foundation Trust. *Lower Limb Therapy conditions*. 2025. URL: https://www.uhdb.nhs.uk/lower-limb-therapy-conditions.

[17] Inge E.P.M. van Haren et al. “Return to sport after anterior cruciate ligament reconstruction - prognostic factors and prognostic models: A systematic review”. In: *Annals of Physical and Rehabilitation Medicine* 68.3 (2025), p. 101921. ISSN: 1877-0657. DOI: https://doi.org/10.1016/j.rehab.2024.101921. URL: https://www.sciencedirect.com/science/article/pii/S1877065724001040.

[18] Motus VR. *Our Products*. 2025. URL: https://motusvr.com/our-products/.

[19] Motus VR. *Then Virtual met Reality!* 2025. URL: https://motusvr.com/about/.

[20] Motus VR. *Virtual Reality Therapeutics*. 2025. URL: https://motusvr.com/.

University of Nottingham
UK | CHINA | MALAYSIA

Team 44

Software

Mannual

University of Nottingham

Part II
Software Manual

Contents

1 Overview

This software manual provides technical documentation and guidance for maintaining, extending, and understanding the VR game project developed in Unreal Engine 5. It is intended for future developers, collaborators, and maintainers who need to work with the existing codebase and system architecture. This manual covers key components including project structure, input configuration, gameplay modules, AI behaviour, and performance optimisation. Wherever possible, the document refers to industry-standard practices such as Blueprint interfaces, behaviour trees, and Git-based collaboration workflows. Diagrams, sample blueprints, and configuration references are included to aid implementation and troubleshooting.

2 System Requirements

2.1 Hardware requirements

- VR headset (Meta Quest/Pico 4)
- Motus Explore Treadmill

2.2 Software requirements

- Java SE Development Kit 17.0.10
- Android Studio Flamingo 2022.2.1 patch 2
- Unreal engine 5.4.4
- Meta XR Unreal Plugin V71
- Meta XR Platform Unreal Plugin V71
- Meta Quest Developer Hub

3 Installation Guide

3.1 Java SDK Set up

Download **Java SE Development Kit 17.0.10** From **Oracle**, extract **jdk-17.0.10** into an appropriate location.

3.2 Unreal Engine Setup

Download and install **Epic Games Launcher**, go to the **Unreal Engine** tab in the sidebar, and then go to **Library**. Press the plus sign next to **Engine Versions** until version **5.4.4** appears, install it. press the drop down menu next to the launch button, go too **Options**, tick **Android** under **Target Platform** then **Apply**

3.3 Android Studio Setup

Download and install **Android Studio Flamingo 2022.2.1 patch 2**

Open **Android Studio**, go into **More Actions**, **SDK Manager**, under **SDK Platforms** make sure **Android API 35**, **Android API 34** and **Android 12L** are installed.

In **SDK Tools** under Android SDK Build-Tools 36, install version **35.0.0-rc1**, **34.0.0** and **33.0.1**.

Under **NDK**, install version **25.1.8937393**.

install all of **Android SDK Command-line Tools** .

Under **CMake** install versions **3.22.1**, **3.10.2.4988404**.

Finally also install:

- Android Emulator
- Android Emulator hypervisor driver (installer)
- Android SDK Platform-Tools

Restart after everything is installed.

3.4 Cloning the repository

Use `git clone` to clone repository into "`Documents/Unreal Projects`" (preferably)

3.5 Installing dependencies

Download

- Meta XR Unreal Plugin V71
- Meta XR Platform Unreal Plugin V71

The two plugins need to be installed on [your Unreal Engine install path]/Engine/Plugins/Marketplace
In your uproject, go to **Settings**, **Plugins**, find and turn both plugins on and restart Unreal Engine.

3.6 Building the project

run SetupAndroid.bat on [your Unreal Engine install path]/Engine/Extras/Android In your uproject, go to **Settings**, **Project Settings Platforms**, **Android SDK** set Location of JAVA, Location of Android SDK, Location of Android NDK path

Back to the front page Navigate to **Platforms**, under **Android**, select **Package Project** then unreal engine will start the packaging process

3.7 Meta Quest

Download and install Meta Quest Developer Hub.
Ensure **Developer Mode** is turned on.

3.8 Deployment

Connect headset to PC Open **Meta Quest Developer Hub** Under the **App** tab, press **Add build**, Then open the packaged .apk

4 System Architecture

4.1 High-level architecture diagram

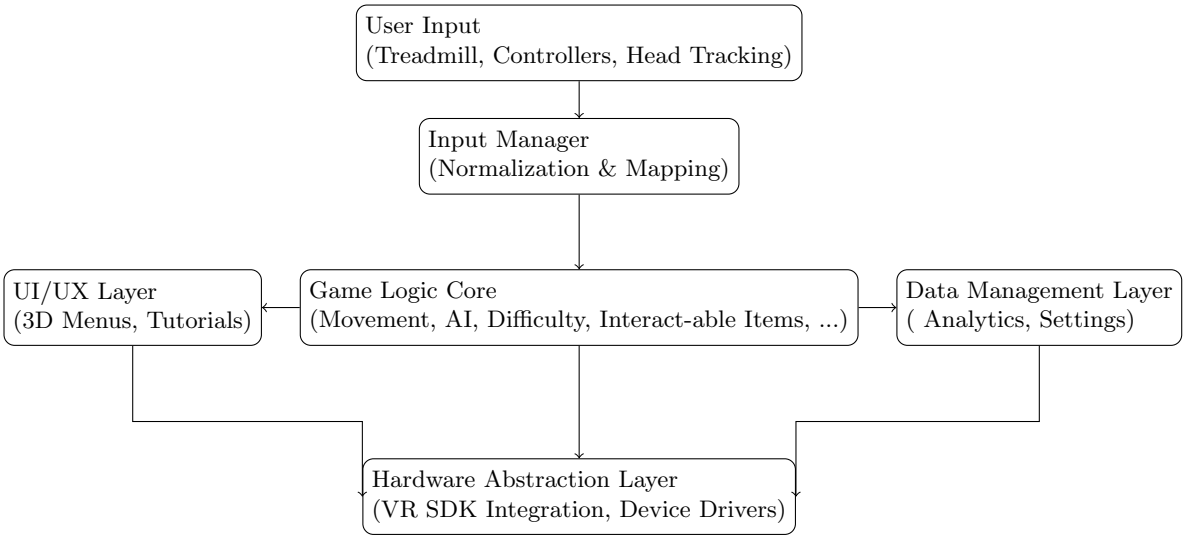


Figure 30: System Architecture of the VR Rehabilitation Game

4.2 Main modules/components overview

VRPawn Control Module: Each mini-game in the project uses a customized VRPawn Blueprint derived from Unreal Engine's default VRPawn setup. These VRPawns are responsible for player movement control, UI integration, and temporary data storage relevant to gameplay. The separation of VRPawn logic per game ensures modular design and allows each gameplay mode to define its own interaction and control scheme.

UI Binding and Interaction: The UI system is organized modularly, with each mini-game assigned its own Widget Blueprint to manage its unique interface layout and logic. These UI elements are integrated into the VRPawn and are visually anchored to the player's view to maintain immersive interaction. Each UI module handles basic in-game navigation, task-related prompts, and dynamic feedback during gameplay.

Game Level/Scene Transition Module: The level transition system enables players to enter mini-games through portals in the central Game Hub and return to the hub at any time via the in-game menu. Each game also supports difficulty selection through the same menu system. Transitions are managed through Blueprint-controlled logic and Unreal Engine 5's level streaming framework, ensuring seamless navigation between different scenes.

User Input Management: The project employs a modular input configuration system using Input Actions and Input Mapping Contexts (IMCs). High-level actions such as movement and interaction are abstracted through Input Actions and then mapped to specific controller inputs via IMCs. This design provides hardware flexibility and maintains input consistency across all mini-games.

4.3 Dependencies

1. **UExplore**: The plugin enabling Unreal Engine 5 to receive input from the VR treadmill was provided by Supervisor Joe Marshall and is available online.[3]

[3] `https://github.com/joemarshall/`

2. **Mazes**: This plugin is used to generate the maze in the Maze level.[4]

5 Module Descriptions

5.1 Chase Level Module

The `Chase` level is constructed using a variety of modular wall assets located in the `Walls` folder. These components are strategically assembled to form a closed, corner-free chase environment designed for smooth VR locomotion.

The player-controlled pawn for this level is `VRPawn_Chase`. This pawn implements all core movement functionalities internally, including forward motion, left and right turning, backward turning, toggling the map display, and switching to a rear view. These actions are bound to the VR controller using Unreal Engine's Enhanced Input System.

The pawn also contains a variable named `Total_Coins`, which is bound to the `CoinUI` interface component. When a player collects a coin in the level, the coin's blueprint logic updates the `Total_Coins` variable in `VRPawn_Chase`, and the new value is immediately reflected in the UI through the binding.

This module demonstrates a clear dependency chain between input mappings, player pawn behavior, collectible actors, and UI feedback, all of which are tightly integrated to support a responsive and immersive VR experience.

5.2 Maze Level Module

The `Maze` level is procedurally generated using the `Mazes` plugin, which creates a dynamic map layout tailored for VR exploration. The movement control in this level follows the same input logic as in the `Chase` level and is implemented within the `VRPawn_Maze` class.

The `VRPawn_Maze` pawn contains two key variables: `health` and `speed`. The `health` variable is dynamically updated based on game difficulty settings and in-game interactions. It is bound to a UI element called `Health_UI`, which reflects its value in real time. When the player collects an `aid` object, the `health` value increases by one. Conversely, when attacked by a `zombie`, the value decreases by one. Once `health` reaches zero, the `Health_UI` will display a "dead" status, indicating game failure.

The `speed` variable is also affected by game difficulty. Difficulty settings are configured through the `Maze_Menu`, which is bound to the `VRPawn_Maze`. The selected difficulty level updates the `difficulty` variable in the `MyGameInstance` class, which is subsequently accessed by `VRPawn_Maze` to dynamically adjust player speed.

Enemy behavior is handled using AI behavior trees. Specifically, the `zombie` actor uses vision detection logic to track the player: once the `VRPawn_Maze` is detected within its camera view, the zombie begins chasing the player based on its behavior tree logic.

This module showcases a tightly coupled system between procedural level generation, difficulty-aware player attributes, user interface bindings, and AI-driven enemy interactions.

5.3 Red Light Green Light Level Module

The `Red Light Green Light` level features a timing-based survival mechanic where visual cues and player motion determine success or failure. The player-controlled pawn in this level is `VRPawn_GLRL`, which implements all player movement logic internally. While the movement implementation follows the same structure as in the `Chase` level, the control scheme is customized to fit the unique mechanics of this mode.

In this level, the color of the ground dynamically changes in response to the `zombie`'s attack behavior. This functionality is orchestrated by the `BP_AttackRange` blueprint, which controls three critical aspects of the gameplay:

- Triggering and managing the zombie's attack animations.

[4] `https://www.mediafire.com/file/0wau3k9d8cj48d7/Mazes_plugin_5.0.3.7z/file`

- Changing the material or color of the ground to reflect safe (green) and danger (red) states.
- Performing real-time player status checks to determine if the player moved during a red-light phase, triggering the death condition if necessary.

The combination of visual cues, precise animation timing, and conditional player input handling creates a fast-paced, high-stakes experience. This module illustrates a tightly integrated system of blueprint-driven environmental feedback, AI animation synchronization, and user control gating, tailored specifically to the "Red Light Green Light" gameplay style.

5.4 Unicycle Level Module

Inside the Unicycle module, there are a number of subsystems that are implemented in order to realise the endless mode, temple-run inspired unicycling game as set out in the game design section (4) of the main report. The main components are laid out as follows:

- Custom unicycle pawn
- Procedural tile generation system
- Powerup system

The unicycle pawn blueprint, located at `Content\Unicycle\BP_UnicycleVRPawn`, contains the core implementation details for the unicycle character. Unlike in the other minigames, the blueprint inherits from the `UPawn` class not `UCharacter`, meaning that more functionality needs to be implemented within the blueprint.

Movement system In order to make use of the functionality that is built in to Unreal Engine, the movement system is implemented using the `FloatingPawnMovementController`, which implements acceleration/deceleration but not gravity. Gravity is simply implemented via a raycast that detects if the Unicycle is grounded, and if not applies a constant movement input downwards. The forward acceleration is applied per tick based off the `treadmillForward` value given by the *Enhanced Input Action* system, in the current forward direction. To extend the movement system, the function `addMovementInput` can be called both from within the blueprint or from an external component. Additionally there is a public parameter `boost`, which acts as a multiplier for the current input. By default it is set to 1, but can be overwritten, which is how the boost mechanic of the powerup is implemented.

Procedural tiles The procedural tile system is central to the Unicycle minigame. It is also designed to be easily extensible and customizable to allow for easy extensions to the game. The `DefaultTile` blueprint forms the basis of this system, and each further tile is designed to be a child blueprint of this class. All tiles contain a green arrow at their origin representing where they will connect to their parent tile, in addition, there may be multiple red arrows, within the exit folder of the blueprint that signal where the children tiles may be spawned (a more detailed description of this this feature can be found below). In order to extend the tile system, all you need to do is add a new subclass of `DefaultTile` and change the scenery or add mechanics, you can then either use the existing exit or add multiple copies of the exit to give the player a choice of direction. Once you add this blueprint to the list parameter `TileList` of the default tile, the rest is handled for you. The tile will spawn randomly as a child of other tiles and be de-spawned automatically when the user has cleared it. Currently, if you want your tile to be more common, you can add it multiple times to the tile list in default tile; however, it could be useful future work to implement a weighted list data structure to have finer control over this functionality.

Power up system The final noteworthy module of the Unicycle game, is the powerup system. Currently there are three types of powerups, regular coins, *mega*-coins, and boosters. You may inspect the existing blueprints to see their implementation, but it is ussually a simple box collider with an `onCollision` event. It is recommended that if you want to add functionality to the VRPawn via a powerup, you create an `ActorComponent` with the relevant data, and then on

collision, to change this data you can use a `getComponentByClass`. This architecture allows a decoupling between the powerups and the specific pawn implementation. In order to add your powerups to the world, you can either create a specific tile where they spawn, or in the default tile, give a random chance of them spawning on the `beginPlay` event. It could be valuable future work to create a system where the default tile contains list of potential powerups and chooses a random one to spawn to allow for ease of implementation - especially if there are many more types of powerups added.

6 Key Feature Implementation

VRPawn-Based UI Attachment: Each mini-game creates a dedicated Widget Blueprint to design its user interface, including buttons, text, and indicators. These UI elements are instantiated in the VRPawn using the `Create Widget` node and assigned to a `Widget Component`. The component is then attached to the player's camera within the VRPawn, ensuring that the UI remains directly in front of the user and follows their head movement, enhancing immersion.

UI Interaction via VR Controller: UI interaction is mapped to the VR controller's input actions. For example, directional navigation within menus is controlled via the joystick axes, while selections are confirmed by pressing down on the joystick. Button behavior is implemented using event bindings within the Widget Blueprint, typically through `OnClicked` delegates for each interactive element.

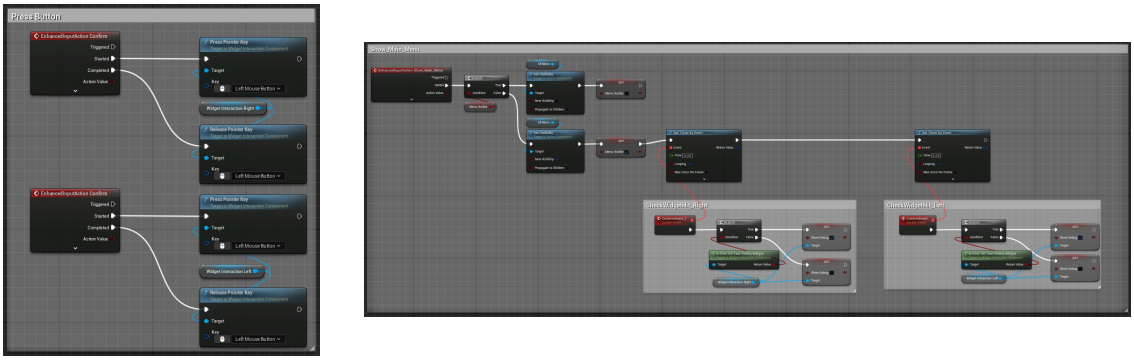


Figure 31: Button Press and Main Menu

Scene and Difficulty Switching: Level transitions are handled using UE5's `Open Level` or `Load Stream Level` nodes, depending on whether a full load or streamed transition is required. Menu widgets provide navigation options to switch between mini-games or return to the Game Hub. Additionally, each mini-game supports dynamic difficulty selection (Easy, Medium, Hard), controlled by setting a difficulty variable via the menu prior to level loading.

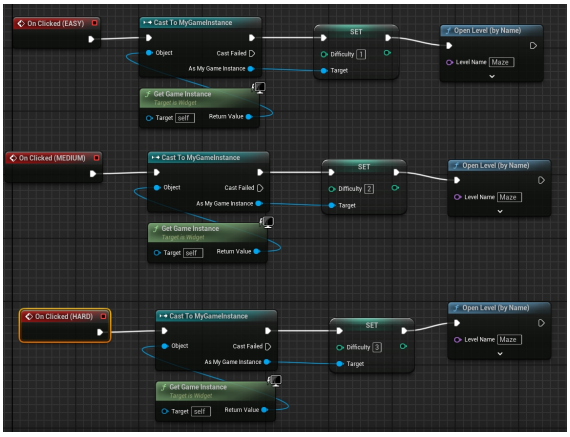


Figure 32: Difficulty Switching

Enemy AI The enemy AI system in our game leverages Unreal Engine 5's Behavior Tree and Blackboard components in combination with blueprint-based logic. Enemies operate in two main states: patrolling and chasing. The behavior tree first checks for line-of-sight; if the player is detected, the AI transitions into a pursuit sequence where the enemy rotates to face the player, moves toward them, and performs an attack upon collision. If no player is in sight, the AI patrols by selecting random locations using `BTT_FindRandomPatrol`, navigating to those points, and waiting for a short random delay before continuing. When the player overlaps the enemy's attack sphere, a blueprint sequence is triggered: the system casts to `VRPawn_Maze` and verifies validity, delays 1 second to simulate wind-up, plays a zombie attack montage via the skeletal mesh's animation instance, and invokes the `TakeDamage` function on the player. A looping timer ensures continuous attacks while the player remains in range. Once the player exits the area, the timer is cleared and the animation is stopped. This system allows for dynamic transitions between idle and combat states based on proximity and visibility, and is modular enough to support further extensions such as varied attack types or smarter behaviors.

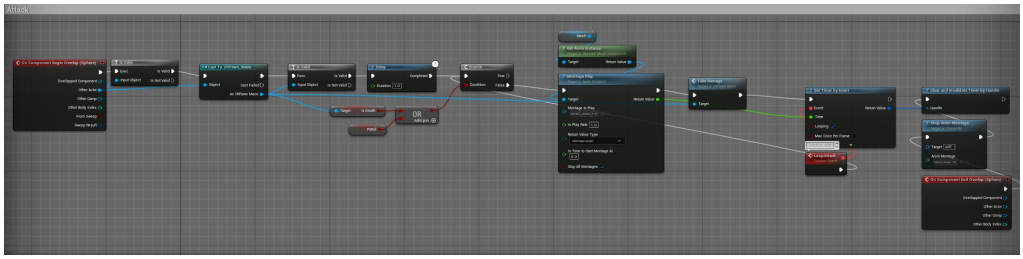


Figure 33: Enemy AI Blueprint

Custom Input System Configuration: The input system uses Unreal Engine's Enhanced Input framework. Input Actions such as `MoveForward`, `TurnLeft`, and `Confirm` are defined as data assets. These actions are linked to physical inputs through Input Mapping Contexts. Within Blueprints, the `Enhanced Input Action` events are used to bind these actions to gameplay behaviour, such as movement or UI control. This setup allows the input configuration to remain abstract and device-agnostic.

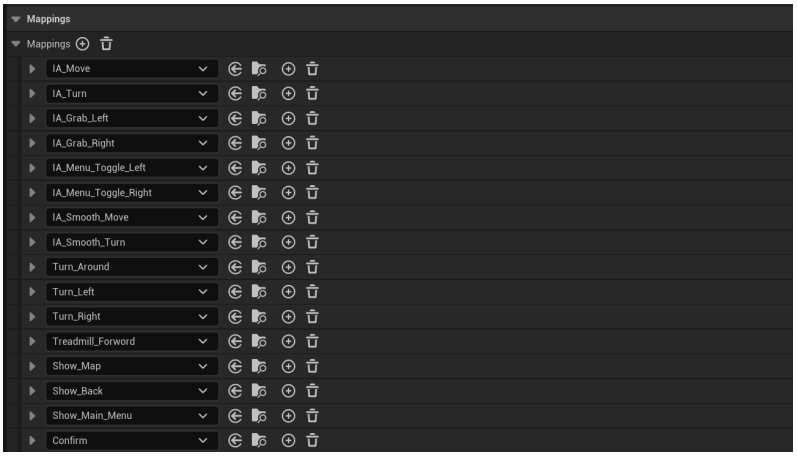


Figure 34: Enhanced Input

Procedural Level Generation: The Unicycle mini-game is designed around an infinite random cave that you have to ride through. Technically, this means that the level and world need to be generated dynamically as you progress through the level. Furthermore, you need to be careful to dynamically unload bits of the level that are no longer in use, especially considering the performance limitations inherent of VR systems. To implement this we actually use something akin to a doubly linked tree structure; where the nodes are subclasses of a base tile blueprint which store references to their previous and future nodes. There are collision boxes which pass a message to the future tiles to spawn their children if they haven't, and if they have then to pass that message forward. Simultaneously, a messaged is passed backwards in a similar way to delete the nodes at the other boundary of the tree to clean up old unused tiles. Finally we keep track of directions that haven't been visited so that there are no 'hanging pointers' - i.e. tiles that don't get de-spawned. This method allows us to continuously traverse through a world whilst making sure only a small number of tiles are ever loaded at once. In addition, because the tiles are implemented as arbitrary subclasses of a default tile class, they have complete control in the look and feel of the tile as well as to how many future tiles are able to be spawned and where they should spawn from. This means that not only are we able to develop a variety of different tiles, but also that a future develop who might take up this project will be able to seamlessly and easily extend the existing system with their own content.

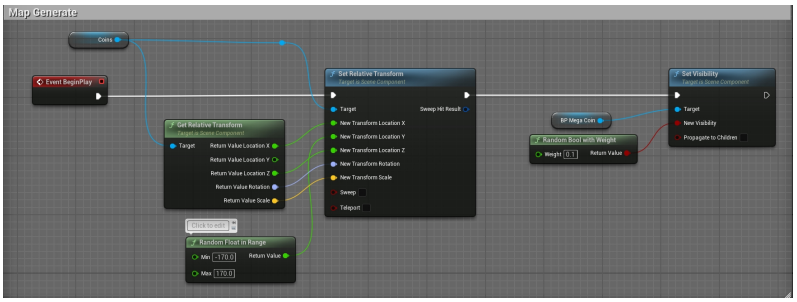


Figure 35: Generation

Unicycle Pawn and movement controller: By default, Unreal Engine provides a subclass of `UPawn` called a character. This character contains a pre-implemented bipedal movement system, which for the standard first person controls used in the maze, chase and red light green light was a suitable base to build their respective pawns over; however, because the movement model for the Unicycle is very different, it was not possible to use the `UCharacter` as a base. To implement the equivalent functionality we needed to instead create a child of the base `UPawn` class. We could make use of the simple floating movement controller that is built in to the engine which handles

(de)acceleration for us but we still needed to implement gravity and steering that responds to the head movement.

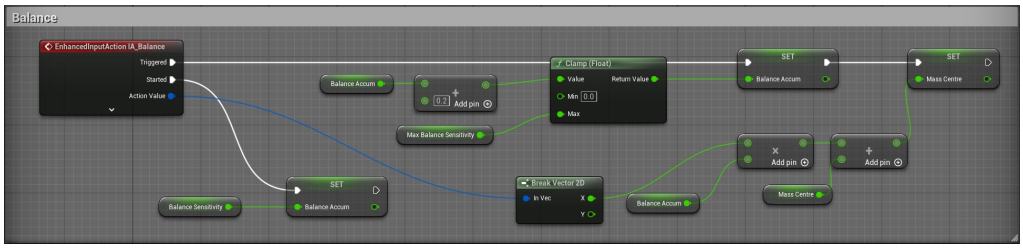


Figure 36: Unicycle Balance

7 Configuration

7.1 Game configuration

7.1.1 Maze Game Parameters

The `Maze` level includes several configurable parameters that control gameplay dynamics, enemy behavior, and player performance. The current default values and their corresponding implementation locations are listed below:

- **Enemy Behavior** (defined in `BT_Enemy`):
  - `Chase Speed:` 300 units
  - `Patrol Speed:` 100 units
  - `Patrol Radius:` 1000 units
  - `Wait Time:` 1.0 ± 0.5 seconds
- **Game Difficulty** (stored in `MyGameInstance`):
  - `difficulty:` Integer values from 1 (easy) to 3 (hard)
- **Player Attributes** (defined in `VRPawn_Maze`):
  - `speed:` 0.8 (easy), 1.0 (medium), 1.2 (hard)
  - `health:` Integer range from 0 to 3

7.1.2 Chase Game Parameters

The `Chase` level includes a set of configurable parameters that govern enemy behavior, player attributes, game difficulty, and collectible tracking. The current default values and their implementation references are listed as follows:

- **Enemy Behavior** (defined in `BT_Enemy`):
  - `Chase Speed:` 300 units
  - `Patrol Speed:` 100 units
  - `Patrol Radius:` 1000 units
  - `Wait Time:` 1.0 ± 0.5 seconds
- **Game Difficulty** (stored in `MyGameInstance`):
  - `difficulty:` Integer values from 1 (easy) to 3 (hard)
- **Player Attributes** (defined in `VRPawn_Chase`):

  - `speed:` 0.8 (easy), 1.0 (medium), 1.2 (hard)
  - `health:` Integer range from 0 to 3

- **Coin Tracking** (handled in `VRPawn_Chase` and associated blueprints):
  - `howmanycoins:` Number of coins collected during gameplay
  - `Total_Coins:` Total number of collectible coins in the level, typically ranging from 50 to 150

7.1.3 Red Light Green Light Game Parameters

The `Red Light Green Light` level includes timing-based enemy behavior configurations that control the attack frequency and animation variation of the zombie character. The current gameplay parameters are as follows:

- **Zombie Attack Behavior**:
  - `Attack Interval:` Randomized between 4 to 5 seconds
  - `Attack Animation Probabilities:`
    - Jump attack: 50%
    - Left swipe: 25%
    - Right swipe: 25%

7.1.4 Unicycle Parameters

In addition to the tunable and extensible tile and powerup system the following parameters are available, in particular to customise the Unicycle Pawns movement:

- Unicycle Pawn Parameters:
  - `TurnAmmount`: Controls the sensitivity to head tilt
  - `Boost`: A temporary multiplication to the treadmill input
  - `FloatingPawnMovementController`: There are a number of parameters in the floating pawn movement controller that control the feel of the movement. These include `acceleration, deceleration` and `maxSpeed`.

- Default Tile Parameters:
  - `TileList`: Controls the tiles that will be spawned. It is possible to add the same tile multiple times to increase the probability of certain tiles spawning.
  - `BackpropThresh`: This controls the recursion depth of the tile despawning algorithm. Note that this can have a big impact on performance.

- Misc:
  - `FogDepth`: In the Unicycle map world, there is a postprocessing effect box. This contains the settings for the fog, most importantly fog depth. The purpose of the fog is twofold: firstly to create an atmospheric environment, but also to hide the fact that not many tiles can be spawned at once without affecting performance. If future work is done on the performances issues of the Unicycle map, then it may be possible to turn down the fog.

7.2 Input mappings

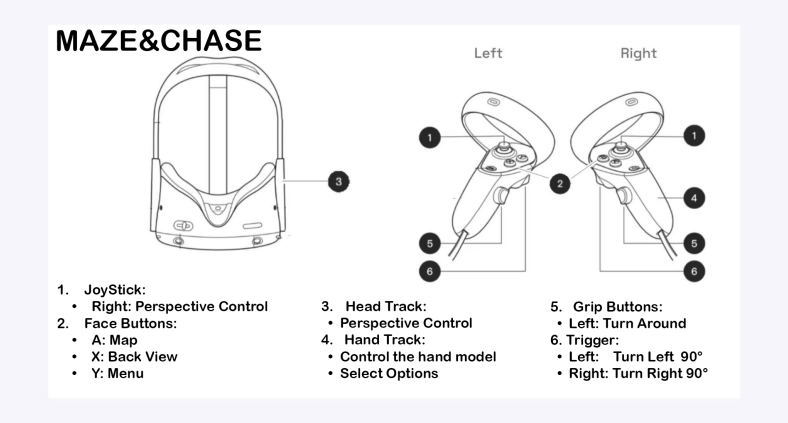


Figure 37: Control Mapping for Maze & Chasing Game

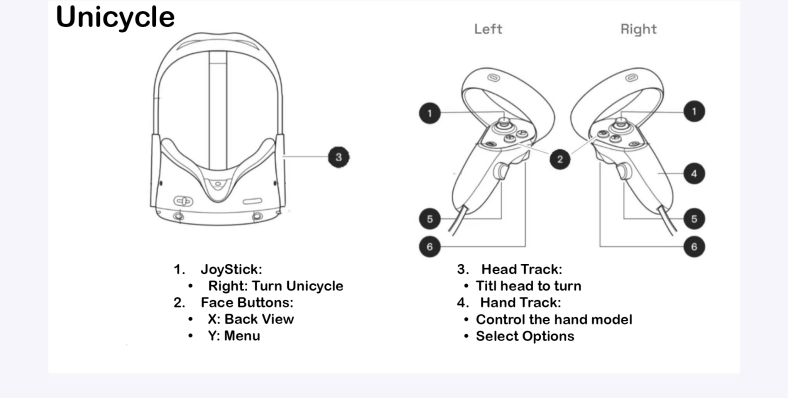


Figure 38: Control Mapping for Unicycle Game

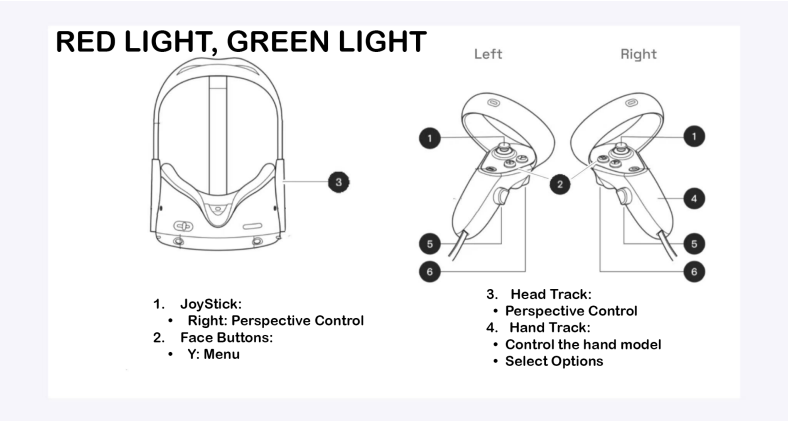


Figure 39: Control Mapping for Red Light, Green Light Game

8 Troubleshooting

Issue	Solution
File content missing	Since this repository has Git LFS enabled, installing it was necessary for proper functionality.
Modules failed to build	Delete the folder **Binaries**, **DerivedDataCashe**, **Intermediate** and **Saved**. Run **buildproject.bat**. This will directly build the project and any necessary modules.
Unable to package - ExitCode: 8	This happens because Unreal Engine is trying to package a module that is editor-only. To fix this, either disable the plugin that depends on that module or rewrite the build file to exclude the editor-only module.
Unable to package - ExitCode: 1	This happens when a dependency that Unreal Engine is trying to package is not in the game Content folder. To fix this, try to migrate the required asset into the content folder.

Table 9: Caption

9 Version Control & Collaboration

9.1 Git Branching Model

Given the binary-heavy nature of Unreal Engine 5 projects—particularly with assets such as textures, 3D models, and blueprint files—version control posed significant technical challenges. Large binary files quickly inflated the repository size and degraded performance for common Git operations like `clone`, `fetch`, and `pull`.

To address this, the team adopted Git Large File Storage (Git LFS)[5], a Git extension designed to handle versioning of large binary assets. Git LFS replaces large files in the repository with lightweight pointers and stores the actual files externally. This significantly reduced the repository size and improved operation speed.

9.2 Merging Workflow

While Git LFS mitigated repository bloat, it introduced challenges in merging conflicting binary assets, such as Unreal blueprint files. Due to Git's inability to perform meaningful diffs on binary content, even minor, non-overlapping changes required full manual resolution.

To minimize such conflicts, the team adopted a collaborative practice based on structured coordination. Team members were encouraged to:

- Work in isolated branches scoped to specific features or levels.
- Synchronize frequently with the `main` branch through small, incremental merges.
- Prioritize communication when potential overlap in blueprint editing was expected.

Although this approach temporarily relaxed the convention of merging only fully complete features, it was effective in reducing merge effort and maintaining momentum in development. For larger teams or enterprise environments, more robust solutions—such as Git LFS file locking or specialized game version control systems (e.g., Perforce Helix)—may be more appropriate.

[5] `https://docs.gitlab.com/topics/git/lfs/`

9.3 Handling Merge Conflicts

During early stages of development, the team encountered a situation where two branches had diverged significantly, both containing essential features. Manual conflict resolution across blueprint assets proved time-consuming and error-prone.

To resolve the situation, the team agreed on a short-term *feature freeze*, during which no new features were developed. All efforts focused on integrating the divergent branches, resolving conflicts, and validating project integrity. This incident reinforced the importance of regular merging and effective communication. Subsequent development cycles proceeded more smoothly, with merge conflicts kept to a minimum.

10 Known Issues & Limitations

10.1 Current bugs

Right-hand Controller Trigger Unresponsiveness: Occasionally, the trigger on the right-hand VR controller fails to respond when performing click interactions, resulting in inconsistent input detection.

Chase Map Flickering Issue: In the Chase level, overlapping map assets cause visible texture flickering upon game start, due to Z-fighting between duplicate or closely positioned meshes.

Chase Enemy Navigation Bug: Enemies in the Chase level may occasionally get stuck in certain areas of the map, especially near corners or geometry seams, preventing them from pursuing the player as intended.

10.2 Limitations of Motus hardware or Quest 2

Quest 2 Headset: The Quest 2 headset offers limited hand gesture recognition capabilities, primarily detecting basic gestures without capturing precise or nuanced hand movements. Additionally, it relies heavily on straps for secure fitting, making prolonged use uncomfortable due to pressure and strain around the head.

Motus Explore Treadmill: The Motus Explore Treadmill can only detect whether the user's feet are in motion, outputting a simple movement signal. It does not track the direction of foot movement, and therefore cannot be used to determine the user's movement direction based on foot orientation.

11 Future Work & Extension Points

11.1 Architectural Extensibility and Refactoring Plans

The project is currently developed using Unreal Engine 5 Blueprints, which facilitates rapid prototyping but requires structure for long-term scalability. The following architectural practices are suggested to support future extension:

- **Dedicated Folder Structure per Mini-game:** Each mini-game should reside in its own folder under the `/Games` directory. Inside each game folder, include:
  - `/Blueprints` – all logic-specific Blueprints (e.g., GameMode, PlayerPawn).
  - `/Assets` – static meshes, textures, and audio used specifically for this mini-game.
  - `/Animations` – animation sequences and blendspaces.
  - `/UI` – widgets, HUD elements, and interface Blueprints.
  - `/Data` – any DataTables, configuration files, or tuning data.

- **Class Inheritance Structure:** Introduce a base class hierarchy to simplify behavior reuse. For instance:
  - All mini-game `VRPawn`s inherit from a generic `BaseVRPawn`.
  - Shared functionality (movement, camera control, tracking input) is handled in the base class.
- **Modular Blueprint Structure:** Refactor gameplay logic into reusable modules (movement, tracking, UI, etc.) to ensure separation of concerns and ease of updates.
- **Resource Cleanup:** Periodically audit and remove unused assets from the project to reduce clutter, save disk space, and improve loading performance.
- **Consistent Naming Conventions:** Apply standard naming schemes across folders, Blueprints, and variables to improve clarity and support collaborative development.
- **Data-Driven Design:** Externalize tunable values (e.g., speed, scoring thresholds) via Blueprint Data Assets or DataTables to allow non-programmers to tweak gameplay.
- **Event-driven Component Interaction:** Use Blueprint Interfaces and Event Dispatchers to create decoupled communication between actors and components.

11.2 Potential New Features

Feature	Description
Reward Systems	Include motivational incentives such as progress badges, trophies, and performance milestones to encourage consistent engagement during rehabilitation exercises.
Rest and Pause Mechanisms	Incorporate clear and user-friendly pause and rest functions, enabling users to take breaks comfortably. This is especially valuable for those undergoing physical therapy.
Customisable Visuals	Provide options to adjust text size, color contrast, and interface simplicity to accommodate users with visual impairments or cognitive processing difficulties.
More Natural and Flexible Controls	Replace or complement snap 90° turning with joystick or smooth-turn options. Allow adjustable sensitivity for foot or head movement.
Expanded Gameplay Mechanics	Add more in-game mechanics and layers of interaction to prevent repetition and maintain user interest, such as stretch-release or heel-lift movements.
Difficulty Scaling	Introduce levels of difficulty or progression systems that adapt to users' rehabilitation stages and physical capabilities.
Immersive Tracking Features	Enhance hand and head tracking fidelity to increase immersion and physical responsiveness.
Time-based Goals and Leaderboards	Include timers, speed-based rewards, and high score boards to introduce friendly competition and performance tracking.
Boost Elements and Game Flow Enhancers	Implement dynamic speed boosters (e.g., auto-triggered accelerators) to intensify pacing and engagement.
Immersive Audio Experience	Incorporate vivid and adaptive background music to enhance user immersion and emotional engagement.
Rehabilitation Analytics Integration	Work with medical advisors to track step frequency and movement data to generate progress records and insights.
Bug Fixes and UX Improvements	Address issues such as confusing layouts (e.g., maze printers) and reward coin glitches to improve satisfaction.
Directional Treadmill Integration	Integrate hardware that can detect footstep direction to improve locomotion control in future iterations.
More 3D Models	Multiple 3D models have already been created but due to time constraints and some changes during development, they weren't used. Many of these models could be re-used at a later date.

Table 10: Proposed Features and Enhancements for Rehabilitation-focused VR Gameplay

University of Nottingham
UK | CHINA | MALAYSIA

Team 44

USER

Mannual

University of Nottingham

Part III
User Manual

Contents

1 Introduction

1.1 What is the game?

Retread is an amusement park themed game centred around the use of the *Motus VR Treadmill* to play the game and aid in the recovery of lower limbs. In this collection of mini-games you will navigate haunted mazes, collect prized coins, test your reactions and traverse treacherous terrains on a unicycle!?

1.2 Who is it for?

This game is designed with rehabilitation in mind for those with injuries to their lower limbs. However, this game is accessible to anyone and everyone regardless of physical ability.

1.3 What's the goal?

The primary goal of this game is to help mitigate the tedious and chore-like nature of rehabilitation through a fun, engaging and thrilling experience that will leave you coming back for more. The game aims not only to aid lower limb recovery patients, but to provide an entertaining video game for any and all users.

2 Hardware Setup

2.1 Required equipment

To play this game you will need:

- A ***Meta Quest 2 VR Headset and Controllers*** (Figure 40).
  - **What comes in the box?**
    1. The Meta Quest Headset
    2. A set of Meta Quest hand controllers
  - **What does it all do?**
    * The Meta Quest Headset will place you in the driver's seat of your character in the Virtual World, allowing you to look around as if you were truly there yourself!
    * The Meta Quest Controllers allow you to interact with the world in different ways

Figure 40: Meta Quest 2 VR Headset and Controllers

- A ***Motus VR Treadmill*** (Figure 41)
  - **What comes in the box?**
    1. The *Motus VR Treadmill*
    2. A pair of slip-on overshoes designed for use with the Treadmill
  - **What does it all do?**

* If the Headset puts you in the driver's seat, then the Treadmill is the pedal to move the car, granting you the ability to move forward!
* These overshoes allow the treadmill to detect footsteps, seamlessly simulating your steps and walking in-game!

Figure 41: Motus VR Treadmill

2.2 Charging & connectivity

Please ensure your *Meta Quest 2 VR Headset and Controller* has sufficient charge before playing the game. The Headset can be charged using the USB Type C cable provided in box.

2.3 Wearing the headset

To safely use the *Meta Quest 2 VR Headset*:

1. Loosen the strap at the back of the Headset
2. Put the headset over your head
3. Adjust the length of the Upper Strap and the height of the headset until you have a clear view
4. Tighten the strap to secure the headset

2.4 Seating requirements

Please ensure to set up your chair in a location away from walls or objects that may easily be hit or knocked over accidentally. A fixed (non-spinning) chair is recommended for this game. Simply place the *Motus VR Treadmill* in front of the chair, and sit down to get started!

3 Game Controls

Each Meta Quest 2 controller contains the following elements (Figure 42):

- **Joystick** (omnidirectional)
- **Trigger Button** (rear side, index finger)
- **Grip Button** (side handle, middle finger)
- **Face Buttons** (A/B on right, X/Y on left)

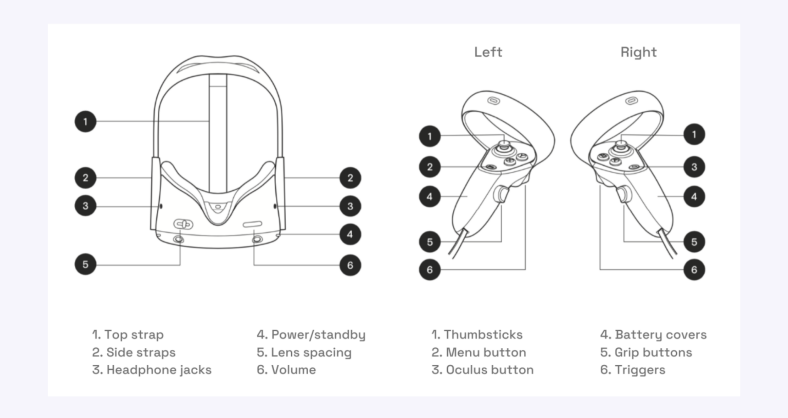


Figure 42: Meta Quest 2 Input Guide

3.1 Forward

To move forward, simply wear the overshoes provided with the *Motus VR Treadmill* and shuffle your feet back and forth on the treadmill!

3.2 Turning

There are 3 different systems for turning/changing direction (note: These systems will automatically be applied depending on what mini-game is being played)

1. **Gridlocked Movement (Maze & Chase):**
   - Your character will only be able to move in **four** directions: North, West, South, East (Forward, Back, Left, Right)
   - To change direction, you can use the *left and right controller triggers*, or press the *left grip button* to turn around 180°
   - **Helpful Tip: Try not to move your head too much when these controls are active as too much head movement may lead to disorientation and confusion!**
2. **Head-Following Movement (Red Light, Green Light):**
   - As you walk, your character will follow the direction that you are facing
   - Simply **look** where you'd like to move and start walking. To change direction while walking, **just look**!
3. **Unicycle-Based Movement (Unicycle):**
   - Have you ever ridden a unicycle? If the answer is yes then this control system will be just like riding a bike!
   - While walking you can turn by **tilting** your head left or right to 'steer' the unicycle in either direction

4 Navigating the Game

Welcome to the Hub-World!!! (Figure 43)

Heavily inspired by the design of Amusement Parks, this is your central hub of sorts. Navigate to your games from here!

Figure 43: The Hub-World

4.1 Main menu overview

At any point, you are able to press the *Y button* on the left controller to open up a menu (Figure 44). From this menu you will be able to perform a number of actions that change depending on where you are in the game.

Figure 44: Main Menu

To select an option on the menu just point one (but not both!) of your controllers at the option and press the trigger button of the controller!

4.2 Selecting A Game

The Hub-World uses the **GridLocked** system of movement (for more information on movement systems, please read *Section 3.2*).

As you explore the Hub-World you will find blue portals around the place (Figure 45). These portals will take you to the mini-games!

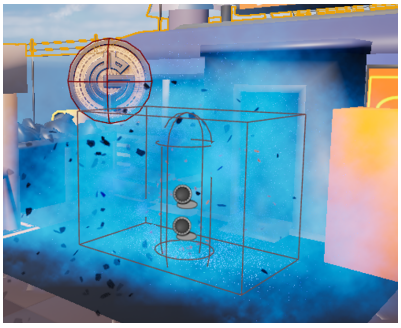
Figure 45: A portal to a mini-game

Next to each portal is a character or model that helps indicate which game the portal is for:

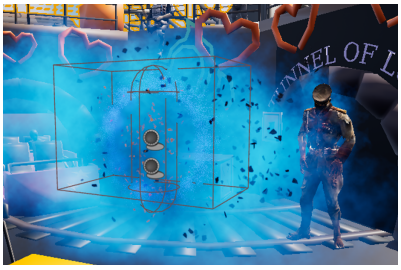
- Chase Game - Zombie Chaser (Figure 46a)
- Maze Game - A Coin (Figure 46b)
- Red Light, Green Light - Zombie Watcher (Figure 46c)
- Unicycle Game - A Unicycle (Figure 46d)

Once you have chosen a mini-game to play, simply enter the portal to be transported to the game of your choice!

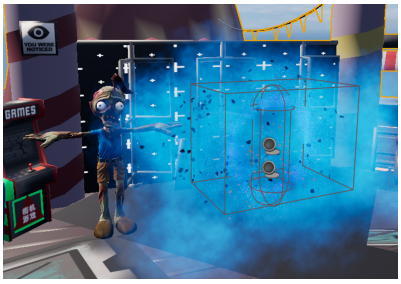
Alternatively you may use the *Main Menu* to select your mini-game!



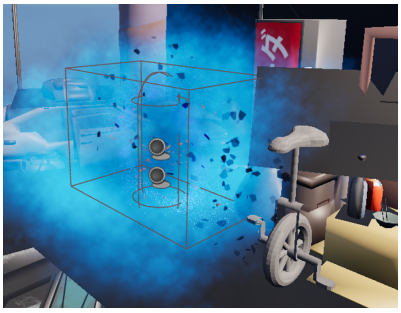
(a) Chase Portal



(b) Maze Portal



(c) Red Light Green Light Portal



(d) Unicycle Portal

Figure 46: MiniGame Portals

4.3 Difficulty selection

To select a difficulty, please use the *Main Menu* from within the mini-game and navigate to your desired difficulty.

Figure 47: Difficulty Selection from within a game

5 Mini-Games Instructions

5.1 Maze Game

Images from Maze and Chase Games will be displayed below both sections in Figure 49

5.1.1 Overview

In this game, you will find yourself trapped within a maze with one goal... **escape**!!! Be careful though... ***You are not alone***

5.1.2 How To Play

This game uses the **GridLocked** system of movement (for more information on movement systems, please read *Section 3.2*).

With aid of a map, you must navigate the maze to find the key to the exit. Once you have collected the key you can carefully make your way to the exit where you will have completed the game as you successfully escape! There will be Zombies chasing you as you explore the maze. Be sure to avoid them, you don't want to be attacked!

Detailed controls will be shown below in Figure 48

Objectives:

- Navigate the maze (Figure 49a)
- Find the key (Figure 49c)
- Find the exit

- Escape!
- Avoid the hunting zombies (Figure 49d) throughout the process

Win Conditions:

- Reach the exit with the key in possession

Lose Conditions:

- Zombies take your final *life* (Figure 49e)

Helpful Tip: Try not to move your head too much while navigating the maze as too much head movement may lead to disorientation and confusion!

5.1.3 Difficulty Levels

This game offers three difficulty levels, please select one based on physical capability/familiarity with the game:

- **Easy:** 3 Lives, 1.2x movement speed
- **Medium:** 2 Lives, 1.0x movement speed
- **Hard:** 1 Life, 0.8x movement speed

5.1.4 Exercise Intensity

This game is recommended for users with an at least moderate level of control over their lower limbs as the chasing aspect of the game may demand quick reflexes and movement.

5.2 Chase Game

5.2.1 Overview

In this game, you will find yourself trapped within a maze with one goal... **Collect all the Coins**!!! Be careful though... ***You are not alone***

5.2.2 How To Play

This game uses the **GridLocked** system of movement (for more information on movement systems, please read *Section 3.2*)

With aid of a map, you must navigate the maze to find and collect every coin available. Throughout your travels you may also find helpful items to pick up! There will be Zombies chasing you as you explore the maze. Be sure to avoid them, you don't want to be attacked!

Detailed controls will be shown below in Figure 48

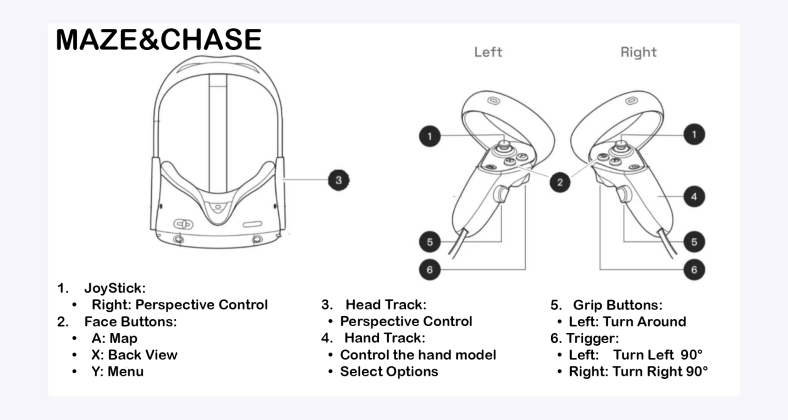


Figure 48: Control Mapping for Maze & Chasing Game

Objectives:

- Navigate the maze (Figure 49b)
- Find and collect coins (Figure 49c)
- Pick up extra lives (Figure 49f
- Avoid the hunting zombies (Figure 49d) throughout the process

Win Conditions:

- Collect every coin

Lose Conditions:

- Zombies take your final *life* (Figure 49e)

Helpful Tip: Try not to move your head too much while navigating the maze as too much head movement may lead to disorientation and confusion!

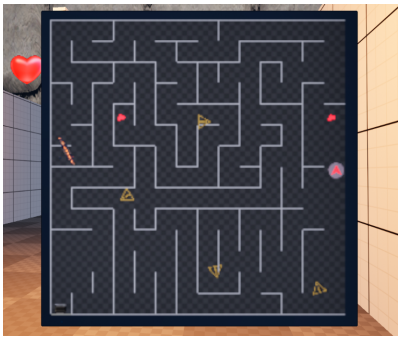
5.2.3 Difficulty Levels

This game offers three difficulty levels, please select one based on physical capability/familiarity with the game:

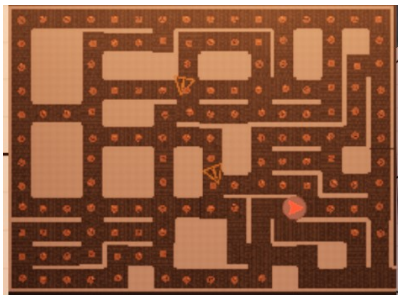
- **Easy:** 3 Lives, 1.2x movement speed, 50 Coins to win
- **Medium:** 2 Lives, 1.0x movement speed, 100 Coins to win
- **Hard:** 1 Life, 0.8x movement speed, 150 Coins to win

5.2.4 Exercise Intensity

This game is recommended for users with an at least moderate level of control over their lower limbs as the chasing aspect of the game may demand quick reflexes and movement.



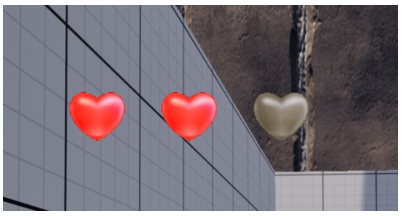
(a) A Map of the Maze Game



(b) A Map of the Chase Game

(c) Collectable Coin/Key

(d) Hunting Zombie



(e) A Health Bar

(f) An Extra Life

Figure 49: Images from Maze and Chase game

5.3 Red light Green light Game

5.3.1 Overview

In this game, you will find yourself facing a... Giant Zombie???? Looks like he'll attack you if you don't follow his orders.

5.3.2 How To Play

This game uses the **Head-Following** system of movement (for more information on movement systems, please read *Section 3.2*).

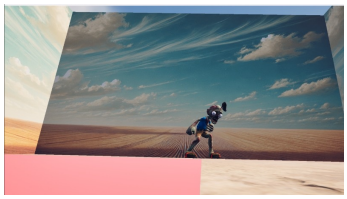
You must make your way forward, ensuring that you do not aggravate the Zombie as you do so. The end goal? Cross the **finish line** (Figure 50a) next to the Zombie.

The Zombie can only attack you when it is facing you, that is to say do whatever you want while it's looking away! (moving forward hopefully) When the Zombie turns around you must fulfil one of two conditions depending on what the Zombie does:

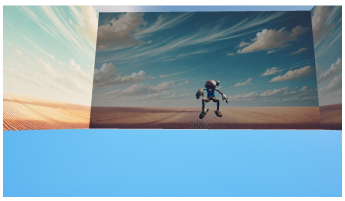
1. Attack Left/Right:
    - The floor that will be affected by the Attack will light up **red** (Figure 50b).
    - Make sure you're not standing on the red when it swings!
2. Jump:
    - The floor will light up **blue** (Figure 50c) to indicate this attack.
    - Stay still while the Zombie jumps to survive!

If you have not fulfilled the condition required to progress, the Zombie will attack you leading to your death...

(a) Finish Line



(b) Red Floor



(c) Blue Floor

Figure 50: Red Light Green Light conditions

Detailed controls will be shown below in Figure 51

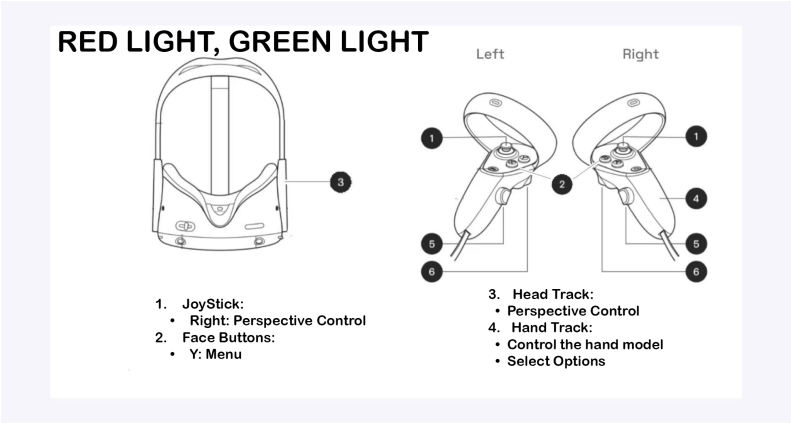


Figure 51: Control Mapping for Red Light, Green Light Game

Objectives:

- Move forward while the Zombie (Figure 52) is looking away
- Get across the finish line
- Obey the Zombie's conditions when it turns around!

Figure 52: The Zombie Watcher

Win Conditions:

- Reach the Finish Line without being attacked

Lose Conditions:

- Failing to fulfil the Zombie's condition, leading to death

5.3.3 Difficulty Levels

This game offers three difficulty levels, please select one based on physical capability/familiarity with the game:

- **Easy:** 1.2x movement speed
- **Medium:** 1.0x movement speed
- **Hard:** 0.8x movement speed

5.3.4 Exercise Intensity

This game is recommended for any and all users! The low physical barrier of entry allows for anyone to play this game regardless of rehabilitation stage.

5.4 Unicycle Game

5.4.1 Overview

In this game, you will find yourself in a cave. Try your best to follow the path and collect as many coins as you can on the way!

5.4.2 How To Play

This game uses the **Unicycle-Based** system of movement (for more information on movement systems, please read *Section 3.2*).

Welcome to our endless game! In this game you will progress through the cave trying to escape a chaser, following the twists and turns of the paths, collecting coins, squeezing through tight gaps, balancing over planks, and going over boost pads for a quick boost in speed!

Detailed controls will be shown below in Figure 53

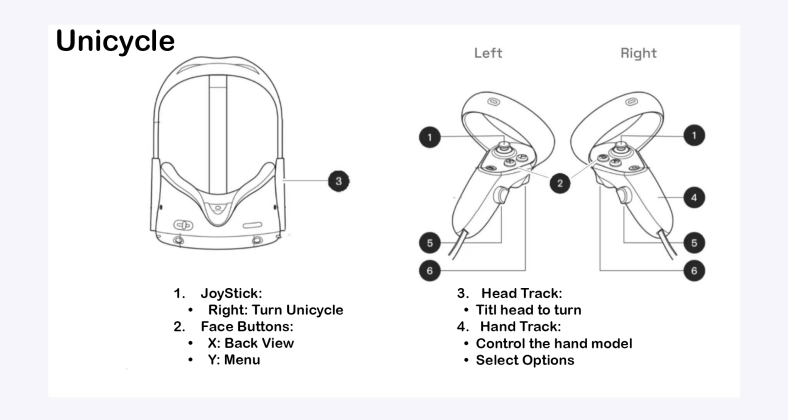
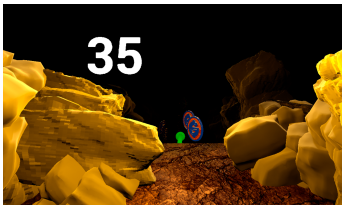


Figure 53: Control Mapping for Unicycle Game

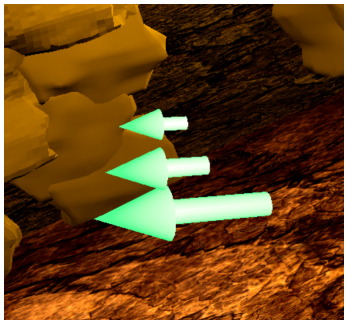
Objectives:

- Move forward and progress through the cave (Figure 54)
- Collect coins
- Squeeze through tight gaps (Figure 55a)
- Boost up hills (Figure 55b)
- Balance over a plank (Figure 55c)
- Try not to go too slow to avoid being caught by the chaser

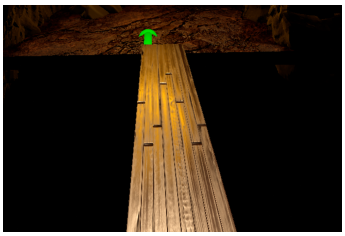
Figure 54: A Snapshot of the Game Environment



(a) A tight gap



(b) A Boost Pad



(c) A plank

Figure 55: Unicycle Game assets

Win Conditions:

- N/A

Lose Conditions:

- Being caught by the chaser
- Falling of the plank

5.4.3 Difficulty Levels

This game starts easy and progressively becomes more and more difficult as you (and the chaser) speed up as the game goes on!

5.4.4 Exercise Intensity

This game is recommended for any and all users! The low physical barrier of entry allows for anyone to play this game regardless of rehabilitation stage.

6 Safety and Comfort

- Take regular breaks of around 10-15 minutes every 30 minutes to avoid eye strain.
- If you are at risk of seizures and epilepsy, are currently pregnant, need an implanted medical device, or suffer from any emotional distress, consult a doctor for further advice.
- If you do not experience any of these symptoms yet and are still concerned, you should also consult a doctor for advice.
- It is recommended for newer users to play in shorter intervals at a time and gradually increase exposure to VR to build resistance to motion sickness.
- Remain seated at all times and ensure that the headset, controllers, and overshoes are securely attached to you during gaming sessions.
- If you are under the influence of alcohol or substance abuse, please refrain from playing the game.
- Adjust the difficulty of the games to your comfort. Do not try to overexert yourself if you are playing the game for rehabilitation.
- This game should be played indoors in a safe and controlled environment with sufficient space for arm movement.

- Be weary of any wires or other similar hanging objects that may cause tripping.
- Do not operate any machinery or partake in any physically strenuous activities that may have serious consequences.
- Remember that all objects see in VR are merely virtual and do not reflect reality
- We are not liable for any deliberate misuse or violation of the recommended safety uses.

7 Contact & Support

Brendan Cheng (Group Leader): eeybc3@exmail.nottingham.ac.uk
Joe Marshall (Project Supervisor): pszjm2@exmail.nottingham.ac.uk
Jamie Pierce (Motus VR Contact): jamie.pierce@motusvr.com
Tom Baker (Motus VR Contact): tom.baker@motusvr.com