Rooted Realms: A Mixed Reality Meditative Interface

Abi Saleh, Joyce
s243948@student.dtu.dk

András, Mátyás Dávid
s242962@student.dtu.dk

Hotait, Hassan
s203211@student.dtu.dk

Soleimani, Zahra
s170422@student.dtu.dk

Figure 1: Banner for RootedRealms generated using OpenAI [1]

Abstract

"Rooted Realms" is an artistic mixed reality meditation experience that transforms a user's physical environment into a mystical space for self-reflection. By integrating hand-tracking, gesture recognition, an interactive particle systems, and spatial audio, the system encourages users to engage in movement-based meditation. Central to the experience is the Tree of Life, a dynamic virtual entity that evolves in response to user interactions, symbolizing personal growth and emotional states. This paper outlines the design, implementation, and user experience of "Rooted Realms," highlighting its potential to foster mindfulness through immersive technology.

CCS Concepts

• **Human-centered computing** → **Mixed / augmented reality**; *User interface design*; *Interaction techniques.*

Keywords

Mixed Reality, Gesture Recognition, Hand Tracking, Particle System, Mindfulness, Immersive Interaction, Spatial Computing, Human-Centered Design, Augmented Reality, Meditation Technology

ACM Reference Format:
Abi Saleh, Joyce, András, Mátyás Dávid, Hotait, Hassan, and Soleimani, Zahra. 2025. Rooted Realms: A Mixed Reality Meditative Interface. In *Proceedings of Creating Digital Visual Experiences — Project Report (02566 CDVE Project '25).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

02566 CDVE Project '25, Technical University of Denmark (DTU)

ACM ISBN N/A
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction and Motivation

Virtual reality-based mindfulness training has been shown to improve mental health outcomes such as anxiety, mood, sleep quality, and emotion regulation often, surpassing traditional screen-based methods in effectiveness [2]. In the past year, immersive VR experiences for meditation and emotional wellness have become popular. Examples include *TRIPP*, which blends breathing exercises with reactive audiovisual feedback [3], *Nature Treks VR*, which transports users to serene landscapes [4], and *Calm Place VR*, which simulates tranquil environments like beaches and forests [5]. However, despite their potential, fully immersive VR systems have key limitations. VR often obstructs users' awareness of their physical surroundings, leading to increased risk of collision and disorientation [6]. Sensory mismatch between visual and vestibular inputs contributes to motion sickness in up to 80% of users [7]. Additionally, cognitive overload caused by VR immersion can diminish user

comfort and long-term engagement [8]. Mixed Reality (MR), by contrast, retains real-world context and spatial awareness while layering interactive virtual elements. This helps mitigate these issues [9]. Our aim through Rooted Realms, was to design an artistic mixed reality meditative interface that balances the emotional depth of immersive experiences with the grounding familiarity of the real environment. By transforming the user's surroundings into a mystical, responsive space, the experience aims to ease the transition into a meditative state.

2 Design

The core idea was to create a mystical mixed reality space where your physical surroundings gradually transition into a realm of magic, reflection, and emotional release. Rather than immersing users in a fully virtual world, we chose mixed reality to maintain a tangible connection to the real environment. The goal was to support a gentle nervous system adaptation, allowing users to shift into a meditative state without abrupt disconnection. We envisioned the mystical space as something that emerges from the real world: roots sprouting from your floor, mushrooms growing out of surfaces, and walls cracking open to reveal portals into animated aurora skies. These details are designed to inspire disconnection, curiosity, and wonder. The visual design language relies heavily on glow, transparency, emissions and soft motion to support a mindful interaction. Since the experience is intended as a meditative journey, the environment encourages users to slow down and focus on the details: flickering fireflies, pulsing mushrooms, shifting skies. These elements are distributed across the user's space to spread their attention across the whole space. Our experience was intentionally designed around movement meditation, as such practices have been shown to promote emotional regulation and reduce anxiety and depression [10]. As users begin to integrate into the space, they are invited to move freely, loosen up, and physically engage through the medium of an attraction particle system. Spatial audio was integrated into the experience for its therapeutic benefits, research has shown it can reduce stress and enhance mood within immersive environments [11]. The particle attraction system paired with hand tracking and gesture detection is the primary medium through which users interact with the environment. It was designed to be visually responsive, playful, and mesmerizing, triggering curiosity through color shifts and motion patterns that respond to hand movement. Particles can be manipulated between the hands, attracted or dispersed through gestures, and even sent toward the tree, where they orbit, and grow leaves. These interactions are intended to pull the user deeper into the environment making them want to explore and experiment. We extended this gesture-based responsiveness to other elements such as mushrooms that react to the user's hand. Figure 2 presents an early concept sketch created during the ideation phase with the assistance of an image generation model from OpenAI [1]. It visualizes our initial vision: the Tree of Life surrounded by mushrooms, seamlessly integrated into a mixed reality living room environment.

3 Implementation

The project was developed in Unity 2022.3.58f1 [12] using the Universal Render Pipeline (URP) [13] , OpenXR [14] and Meta SDK

Figure 2: AI Generated Ideation Image

version 74.0.1 [15], targeting the Meta Quest 3 [16]. Core modules include hand tracking, gesture detection, environment mapping via MRUK [17], shader-based visual effects, spatial audio, and procedural object spawning and animation. These systems are interconnected using custom C# scripts and an event-driven logic structure to enable an interaction and feedback pipeline.

3.1 Spatial mapping and Mixed Reality integration

We used Meta's MR Utility Kit (MRUK) [17] to integrate spatial understanding and room-aware asset placement. MRUK enables environment scanning, anchor recognition, and procedural spawning by identifying surface types such as floors, furniture, walls, and ceiling. The Effect Mesh [18] was used for environmental mapping optimization. It creates special-effect meshes out of the Scene objects in the room. This enables virtual elements to react to the geometry of the room, which enhances the mixed reality layering. After the environment is mapped, spawning is triggered based on spatial data and asset type. Spawn location is randomized within constrained parameters such as bounds, clearance distance and surface type. This prevents overlap with real furniture or with other virtual assets. Semantic filtering is applied to spawn on the appropriate surface type:

- **Mushrooms** are spawned on horizontal surfaces such as floors and furniture tops.
- **Cracked aurora portals** are placed on vertical wall surfaces.
- **Fireflies** float freely, animated to hover in space.
- **The Tree** is placed 2 meters in front of the user, relative to the headset position.
- **The animated ceiling portal** is placed in a radius on top of the tree and the user, as it is best experienced directly overhead.

3.2 Visual Asset Design and Animation

The Tree is a central element in the experience. It is procedurally spawned in a gradual manner, it first appears in the scene without

leaves. It is composed of sub-meshes and the leaves are animated. Part of the leaves grow to life at the start of the experience, the rest are grown later on in response to user interaction. The original model was sourced from Sketchfab [19]. Blender [20] was used to remove non-fitting branches and to adjust leaf count and placement. Some parts of the trees were disregarded to reduce vertex count and improve performance. The asset and animation were extracted and integrated into Unity. The animator controller was used to integrate leaf growth effects. Textures (including base color and roughness maps) were edited in GIMP [21] to remove opaque backgrounds, making the leaves fully transparent where needed. To optimize the tree's appearance for Unity URP and Quest 3, a custom URP-compatible shader with alpha clipping was created for leaves. Wall cracks were integrated through custom textures and URP material which include a cracked frame with an aurora interior on a transparent material placed on a plane. Additional assets such as fireflies and mushroom were also sourced from sketchfab [22].The fireflies in the scene were animated procedurally using a sine based time function, applied to their vertical position. This causes each firefly to move up and down in a smooth oscillation, creating an ambient floating motion. We assigned slightly randomized amplitudes and speeds per firefly, for the movement to appear organic and non-uniform. The interactive mushrooms employ a multifaceted animation system implemented in **C#**. The core growth animation is driven by a sinusoidal function that modulates a normalized size factor over time. This factor is smoothed using interpolation controlled by the sizeSmoothing parameter, and the final scale is calculated by combining this factor with a maxScale value and a level-based multiplier. This creates a subtle pulsing effect that visually suggests organic growth. For color animation, the system operates in **HSV** color space, cycling the hue over time while maintaining consistent saturation and brightness. This enables smooth rainbow-like transitions. The emission color is dynamically recalculated for each frame and scaled by a configurable emission Intensity, which increases with each level up to produce a more intense glow for higher-level mushrooms. Some animations were explored but did not make it to the final demo due to design choices. Transparency is enabled by manually configuring the material's blend modes and render queue to support semi-transparent rendering. However, this setup can introduce depth-sorting artifacts in the Universal Render Pipeline (**URP**) when multiple transparent objects overlap. Movement is handled through three distinct modes: **circular** (based on cosine/sine oscillations), **spiral** (combining circular motion with vertical displacement), and **levitating** (using a vertical sine wave). These modes can be switched interactively, and each mode subtly adjusts animation parameters such as growSpeed and float frequency. Ambient spatial audio was added to enhance immersion and guide movement. Each gesture has a soft feedback sound to confirm detection.

3.3 Shader Graph Development

The aurora shader was built in Unity's Shader Graph using a procedural, layered noise approach inspired by a guided tutorial [23]. It combined gradient sampling, noise distortion, and dynamic UV manipulation to simulate atmospheric light phenomena. The core logic begins with normalized world-space coordinates projected onto the XZ-plane to prevent spherical distortion in the skybox. A gradient node defines the aurora's color spectrum (teal-to-purple), sampled via UV coordinates where the Y-component, modified by GradientOffset, controls vertical color blending. Wave structures are generated through Perlin noise [24] (Simple Noise node, scale=20) intensified via a Power Node (AuroraPower), creating non-linear luminance fall-off. Temporal animation is achieved through UV scrolling (Time × ScrollSpeed), with periodic boundary handling via **Sine** and Remap nodes to prevent artifacts. Spatial distortion is introduced via a Twirl Node, dynamically perturbed by secondary noise modulated between DistortionMinMax values, producing organic undulation. The final output blends with the skybox using a Power Node (AuroraBlend) for emissive intensity control. The main shader graph is shown in Fig 3.

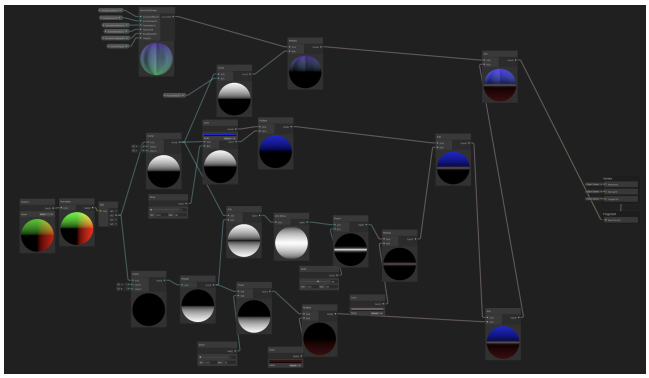

Figure 3: Skybox Shader Graph

The aurora shader incorporates a procedural subgraph responsible for UV distortion and noise modulation. This subgraph takes inputs like ScrollSpeed and NoiseScale, applies Perlin noise with dynamic tiling, and outputs distorted UVs for the main gradient sampling. The subgraph ensures reusable noise generation while maintaining performance by avoiding redundant calculations in the main shader. A high-resolution 2D star texture was blended into the aurora Shader Graph to give a twinkinlg star effect.To animate the aurora itself, we used a Time node combined with a Sine function to implement a looping oscillation. This resulted in an animated, wave-like color shift that cycles across the shader over time giving the aurora a breathing motion.

3.4 Particle Attraction System

Controlling the particle system is the primary way the player can interact with the environment in the experience. The behavior of the particles was inspired by flowing elemental magic and is built upon a project by Till Holzapfel from the Cyberdelics team [25]. The original project implemented a highly detailed and performance-optimized control system. It defined attraction points controlled by hand movements, to which the particles were drawn with varying biases. We found the original behavior intuitive and enjoyable, so we decided to include it in our project with minimal changes, while expanding its functionality to enable user interaction with the broader experience.

Aesthetics

The look and feel of the particles fits in well with the mystical, magical, and colorful aesthetic of the environment. The particle system uses Unity's built-in particle system and the appearance is modified programmatically. The color of the particles is constantly shifting and also influenced by their movement, giving them a vibrant, reactive quality. This creates a dynamic and immersive visual experience that aligns with the overall atmosphere of the environment. To implement this, Unity's `Gradient` system combined with sinusoidal time-based animation to drive continuous transitions across a palette is utilized . Particle color updates are interpolated on each frame in a GPU-efficient manner, and the transparency is adjusted using a global `Alpha` control. These values are passed directly into the particle shader using the `AttractionJob`, ensuring responsiveness with minimal overhead.

Particle Behavior States

The behavior of the particles can be broken into three main stages:

(1) Initial 'blob' stage
(2) Close-to-hand control stage
(3) At-object control stage

This progression is managed through a state machine embedded into each particle's data structure. Each particle is assigned a *ParticleState* (e.g., `InHands`, *SentToTarget*, *Orbiting*) and these states are updated in real time using gesture recognition. The logic for controlling particle movement is executed through Unity's Job System. Specifically, a custom parallel job (*AttractionJob*) is scheduled using *IJobParticleSystemParallelForBatch*, allowing thousands of particles to be updated per frame without CPU bottlenecks.

Transition and At-Object Stage

When the player points toward an object (specifically, the tree), the particles enter a transition state. They travel along a direct path from the player to the tree, with added noise for a more natural movement aesthetic. Once they reach the vicinity of the tree, the particles enter the final stage—*at-object control*. In this stage, the particles orbit the tree, no longer directly attracted to the player's hands. However, the player retains indirect control: the height and radius of the orbital path can be adjusted by raising or extending the hands.

Interactive Feedback: Leaf Spawning

In the at-object stage, the particles link to a *leaf spawning mechanism*. Raising the particles triggers the growth of leaves on the tree, and extending the rotation radius expands the spawning area. This mechanic provides satisfying visual feedback, offering players a rewarding sense of contribution to the environment. It is implemented by monitoring changes to the constraint center and radius and activating corresponding visual effects. The system encourages engagement while reinforcing the relaxing nature of the experience. If the player wishes to return to the previous stage, they can do so by making a fist, which calls the particles back. Due to the use of Unity's multithreaded particle jobs and Burst compilation, these feedback mechanisms remain performant, even with tens of thousands of particles under active control.

3.5 Hand Gesture Detection & Interaction

To enable gesture detection for interaction with the environment the package *OpenXR Hands* was used. The package enabled us to capture the position and orientation of 26 hand joints which are used to build custom gestures. By tracking those joints, each finger is described using the 5 states shown in Fig. 4.

State	Description
Full Curl	The overall curve of a finger. (A combination of base and tip curl.)
Base Curl	The angle between the hand and the base of the finger.
Tip Curl	The curve of the outer portions of the finger.
Pinch	Whether the finger is in a pinching posture based on how close the tip of the finger is to the tip of the thumb.
Spread	The spread between this finger and the next (moving from thumb to little finger).

Figure 4: Open XR Finger State Description

By setting a target range for states for one or more finger, the gesture is detected when the current hand shape satisfy the conditions for each state for every finger. An example of gestures that can be detected using the "Full Curl" state is shown in Fig 5.

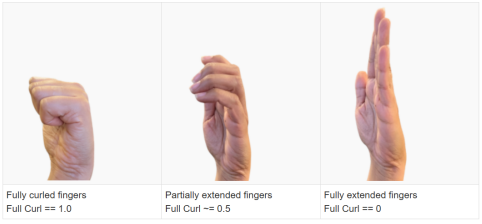

Figure 5: Example of target state for different gestures.

The gesture detection Game Object used allowed us to call a function when a gesture is performed and/or when it ends. This enabled us to add different interaction mechanisms as shown in Fig 6. As a concrete example, when the pointing gesture is detected, the function *SetParticlesToSentToTarget* is called from the script *EMerge*. This causes the particles to orbit the tree. Additionally, to provide the user with some feedback, a sound is played and the gesture icon in the UI turns blue on gesture performed. The described framework is used to develop the interactions shown in Fig 7 to create a fun and engaging experience.

4 Results, Testing and Discussion

4.1 Results

The effect mesh which was used to place objects and check mapping accuracy. As shown in figure 8, the environment furniture is mapped, creating a box layout used for semantic asset placement. Here, we can see elements placed relative to these boxes. The blue outline does not appear in the actual experience. The aurora shader in Figure 9 effectively simulated flowing lights with tunable wave and color controls.

The screenshots from the experience in Figure 10 shows the tree as the central object, with its dynamic leaves. The aurora cracks were dispersed on wall faces, and the ceiling portal was placed above the tree. Fireflies moved around the space, while mushrooms were spawned on the surfaces, responding with dynamic growth, animated color transitions, and interactive movement.

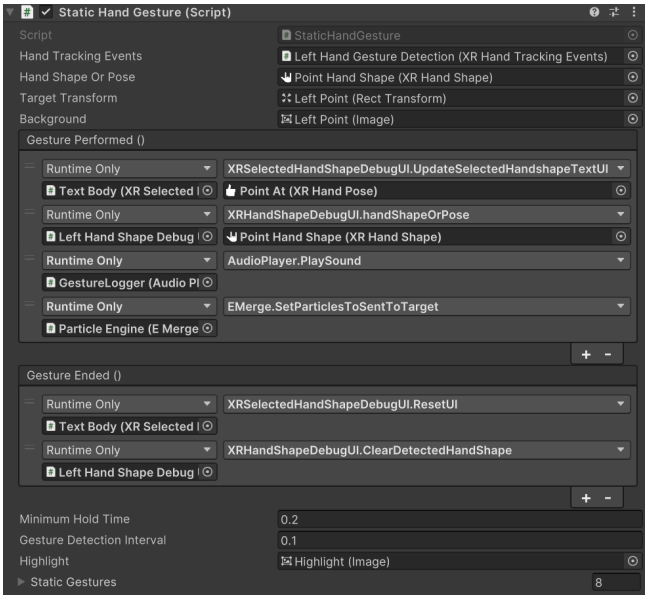

Figure 6: Interaction via gesture detection.

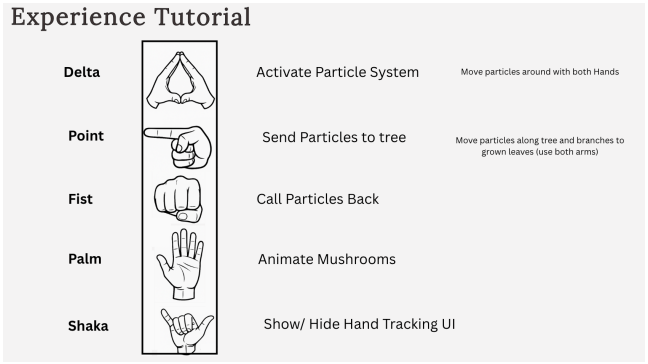

Figure 7: Different interaction mechanisms.

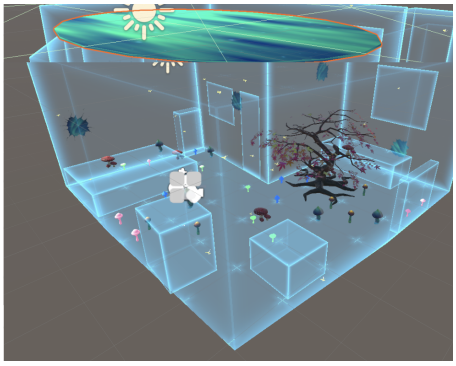

Figure 8: Effect Mesh for environmental mapping

The particle system, as seen in the figure above, is attracted to the hand joints and forms dynamic shapes based on your movements. The UI for gesture detection can be seen in its collapsed version, which can be activated with a gesture for debugging. The full version is shown in the Figure below.

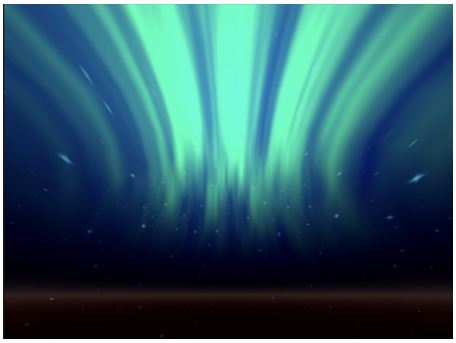

Figure 9: Screenshot from the shader graph Aurora

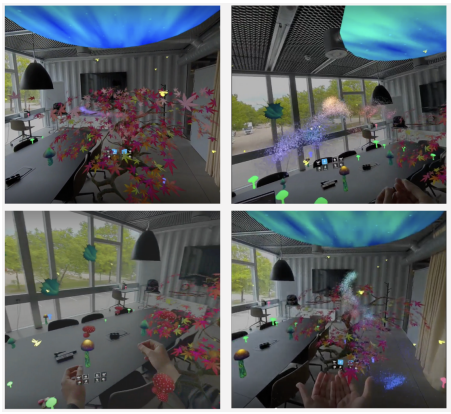

Figure 10: Screenshots from the Experience through Quest 3

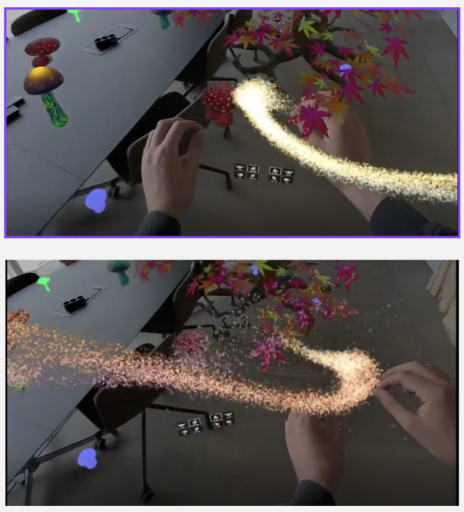

Figure 11: Particle System Screenshot

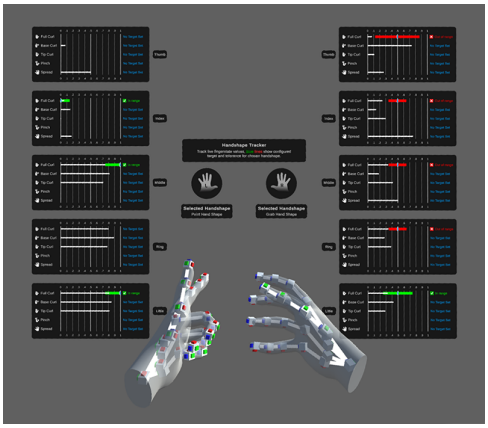

Figure 12: Gesture Detection Debugger UI

4.2 Testing and Discussion

Internal Testing We tested the system in four different environments with varying sizes, features, and sunlight exposure. This allowed us to identify parameters that needed adjustment, such as the size and height of the ceiling aurora and tree. We also found that the number of assets like the amount of mushrooms and fireflies needed to be adapted based on the room size to prevent it from feeling too empty or overcrowded. In rooms with a lot of furniture, tighter bounds were necessary, meaning less space for each item. The gestures and their corresponding interactions were intentionally chosen with the goal of being intuitive and easy for users to adapt to. During internal testing, we fine-tuned certain gestures to avoid unintentional triggers. For example, we noticed that people naturally defaulted to an open palm hand shape, which was frequently mistaken for a command to call back the particles. To resolve this, we changed the trigger gesture to a fist. Additionally, for palm detection, we set a delay of at least one second to prevent accidental activation.

User Testing

A showcase was conducted by the course instructors, where users were introduced to the experience, presented with a tutorial and fitted with the headset. Users were instructed to take their time, look around the mixed reality space and explore the particle system. The feedback we received was positive. Users particularly enjoyed the visual aspects of the experience, and many were impressed by the mixed reality effects, and the particle system. The gestures were smoothly detected, and users quickly adapted to the experience, understanding the function of each gesture. The most enjoyable interaction for users was sending particles to the tree. However, one issue that arose was that, despite our goal to encourage users to slow down and explore the environment, they tended to rush and focused primarily on sending particles to the tree. This led to less interaction with other elements. Some users seemed to experience a bit of a sensory over stimulation, especially those who weren't accustomed to mixed reality, due to the sounds, colors, and animations happening all together. We would like to adapt the user experience to encourage more engagement with the environment and the particles. This could involve making adjustments to the layout or the flow to help users better adapt to the experience. It's also worth noting that, as this was a showcase, the busy atmosphere may have influenced how users approached the experience. Further testing in calmer environments would be required to gather more objective data.

4.3 Challenges and Limitations

In the course of developing the experience, several challenges and limitations were encountered. One of the primary issues was the trade-off between artistic control and performance. To ensure efficient real-time rendering, real-time gradient editing was sacrificed, and pre-baked color ranges had to be used. Additionally, transparency depth artifacts arose due to the limitations of the URP's transparent rendering system. The lack of proper depth sorting caused rendering issues with overlapping transparent objects, particularly between the mushrooms. These artifacts were resolved by manually adjusting render queues and utilizing depth offsets to maintain visual accuracy. Further limitations were encountered with the tree model complexity, where testing multiple tree meshes revealed performance issues on the Quest 3, with crashes or heavy lags. Simplified models were chosen to mitigate this, and mesh simplification was applied to improve performance. Shader layering conflicts also presented challenges when stacking the passthrough, cracked frame, and animated aurora shaders. This caused rendering issues, which were addressed by simplifying the shader setup to a static aurora crack texture on the walls and keeping the animated shader solely on the ceiling. In low-light conditions, MRUK scanning failed to perform properly, leading to inaccurate anchor detection and random asset placement. This issue caused flickering and unstable rendering during the mapping phase. The gesture detector was also limited in functionality, as it could only detect static poses. This restriction prevented the integration of more dynamic gestures, such as swipes or throws, which had been considered for more interactive elements like grabbing or throwing digital objects at the tree. Due to this limitation, the anticipated interactive mechanisms could not be implemented within the current framework. An additional challenge was integrating our particle system features into the pre-existing behavior without compromising the system's stability. This was solved by implementing a *state machine* for the particles, allowing us to layer new behavior on top of the original codebase without extensive rewrites. A critical aspect of the technical integration was ensuring the stability of job-based execution. We had to carefully manage data dependencies between particle updates and gesture input to avoid race conditions. Additionally, state transitions were propagated through a central update loop, which queued `CompleteAllJobs()` calls to prevent concurrent writes during frame updates. These challenges highlight the need for further optimization and testing, particularly with respect to performance in varied lighting conditions, improved gesture detection, and scalable asset handling for large scenes.

4.4 Future Work

There are several potential improvements to both particle control and interaction. In the *at-object* stage, the control system could be expanded to respond to more dimensions of hand movement, creating a more immersive and fluid experience. Additionally, interactions could be extended to include multiple objects, giving the

player more to explore and do. This would enrich the environment and deepen engagement without undermining the calm, meditative tone. We also envision enhancing the experience by allowing the target for the particles to be dynamically set, which would increase user engagement and time spent within the environment. Currently, the particles are limited to orbiting the tree. However, by introducing a hand-aim feature, users could select the target for the particles themselves, offering more control and interactivity.

Figure 13: Quest 3 Hand Aim

The hand-aim feature could be implemented using the Meta Aim Hand extension of the OpenXR package, which provides hand position and orientation for targeting within virtual space, as shown in Fig. 13. This extension supports both controllers and hand tracking, enabling intuitive pointing or aiming. Additionally, the environment design can be further refined to create a smoother, more cohesive experience by providing a slower integration process to help users engage with the environment more thoughtfully. Incorporating grabbable objects, such as mushrooms or plants would enhance the immersion and customization of the experience. We also plan to conduct further testing to refine the integration process, ensuring the space feels welcoming rather than overwhelming. Further research on the psychological impact of immersive experiences would be required to optimize the flow.

5 Conclusion

Rooted Realms leverages immersive technology to support meditative practices and emotional regulation by blending symbolic elements with gesture-based interactions, spatial audio, and real-time visual feedback. We managed to build a system that brings together hand tracking, shader-driven visuals, a particle system and procedural animated assets in a way that invites users to move, explore, and engage with their surroundings. The fusion of real-world geometry with virtual elements created an experience that felt dynamic, embodied, and at times enchanting. During user testing, reactions were positive; people enjoyed the visuals and instantly connected with the particle interactions. But we also noticed tendencies to rush through the experience or focus solely on the tree, sometimes missing the slower details. In some cases, sensory stimulation was a bit intense for first-time MR users. These moments highlighted the need for better pacing and more intuitive onboarding. Looking forward, there's a lot of room to grow. Expanding the gesture system, enabling dynamic targets, and making the space more responsive and customizable could deepen the meditative qualities of the experience. There's also potential for therapeutic use in wellness, stress reduction, or clinical settings which would require further research and collaboration with mental health professionals. Ultimately, Rooted Realms is a small but meaningful step toward immersive technology that isn't about escape, but presence.

6 Acknowledgments

We extend our gratitude to Cyberdelics (cyberdelic.nexus) for their invaluable input and resources that significantly contributed to the development of "Rooted Realms."

Contribution	Joyce	Zahra	Hassan	Matyas
Particle System Behavior				X
Particle System Appearance				X
Mixed Reality Integration	X			
Environment and Asset Design	X			
Asset Animation	X	X		
Materials and Shaders	X	X		
Hand Tracking Integration			X	
Hand Gesture Recognition			X	
System Integration and Testing	X		X	

Table 1: Project Contributions by Team Member

7 Supporting Material

The following Google Drive link (available here) contains the full Unity project as a zipped folder, the APK build that can be directly deployed on the Quest, and a video demo of the experience.

References

[1] OpenAI, "Chatgpt: Large language model." https://chat.openai.com, 2023. Accessed: 2025-05-14.
[2] S. Zhang, M. Chen, N. Yang, S. Lu, and S. Ni, "Effectiveness of vr based mindfulness on psychological and physiological health: a systematic review," *Current Psychology*, 2021.
[3] TRIPP Inc., "Tripp – meditation in virtual reality," 2024. Available at: https://www.tripp.com/.
[4] Nature Treks VR, "Nature treks vr," 2024. Available at: https://www.naturetreksvr.com/.
[5] Calm Place VR, "Calm place vr – mindfulness and relaxation," 2024. Available at: https://calmplacevr.com/.
[6] N. C. Nilsson, S. Serafin, and R. Nordahl, "Obstacle avoidance in virtual reality: the importance of real-world awareness," *IEEE Virtual Reality*, pp. 313–314, 2018.
[7] P. Kourtesis, S. Collina, L. A. Doumas, and S. E. MacPherson, "Cybersickness in virtual reality: examining the influence of content, style, and training," *Frontiers in Psychology*, vol. 10, p. 158, 2019.
[8] G. Makransky, T. S. Terkildsen, and R. E. Mayer, "Adding immersive virtual reality to a science lab simulation causes more presence but less learning," *Educational Technology Research and Development*, vol. 67, no. 6, pp. 1561–1582, 2019.
[9] A. Dey, M. Billinghurst, R. W. Lindeman, and J. E. Swan, "Effects of ar interfaces on cognitive load and performance," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1328–1338, 2018.

[10] P. Payne and M. A. Crane-Godreau, "Meditative movement for depression and anxiety," *Frontiers in Psychiatry*, vol. 4, p. 71, 2013.
[11] D. M. Greenberg, E. Bodner, A. Shrira, and K. R. Fricke, "Decreasing stress through a spatial audio and immersive 3d environment: A pilot study with implications for clinical and medical settings," *Music & Science*, vol. 4, p. 2059204321993992, 2021.
[12] Unity Technologies, *Unity Editor 2022.3.5f1*, 2023. Available at: https://unity.com/releases/editor/whats-new/2022.3.5f1.
[13] Unity Technologies, *Universal Render Pipeline (URP)*, 2023. Available at: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@latest.
[14] Khronos Group & Unity Technologies, *OpenXR Plugin for Unity*, 2023. Available at: https://docs.unity3d.com/Packages/com.unity.xr.openxr@latest.
[15] Meta Platforms, Inc., *Meta XR SDK v74.0.1 for Unity*, 2024. Available at: https://developer.oculus.com/downloads/package/unity-integration/.
[16] Meta Platforms, Inc., "Meta quest 3 - mixed reality headset," 2023. Available at: https://www.meta.com/quest/quest-3/.
[17] Meta Platforms, Inc., "Mruk: Unity mr utility kit - getting started," 2024. Accessed: 2025-05-18.
[18] Meta Platforms, Inc., "Effectmesh: Unity mr utility kit - features - effect mesh," 2024. Accessed: 2025-05-18.
[19] Sketchfab User, "Japanese maple tree," n.d. Accessed: 2025-05-18.
[20] Blender Foundation, *Blender: Free and Open 3D Creation Software*, 2024. Version 4.x, Accessed: 2025-05-18.
[21] The GIMP Team, *GIMP: GNU Image Manipulation Program*, 2024. Version 2.10+, Accessed: 2025-05-18.
[22] Sketchfab, "Sketchfab: Discover the best 3d models," 2024. Accessed: 2025-05-18.
[23] Unity Dev Tutorials, "Aurora shader graph tutorial in unity urp," 2023. Accessed: 2025-05-18.
[24] K. Perlin, "Perlin noise," 1985. Perlin noise algorithm used in procedural animation and textures. See Perlin, K. (1985). An image synthesizer. ACM SIGGRAPH Computer Graphics, 19(3), 287–296. https://doi.org/10.1145/325165.325247.
[25] Cyberdelic Nexus, "cyberdelix nexus," [2024].