University of Pavia

Faculty of Engineering

Department of Electrical, Computer and Biomedical Engineering

Master Degree in Industrial Automation Engineering

Navigating Indoor Spaces in Virtual Reality: A System for Spatial Mapping and Interaction Analytics

Supervisor:
Prof. Tullio Facchinetti

Candidate:
Reza Jahromibooshehri

Academic Year 2024/2025

Abstract

This thesis investigates lightweight wayfinding aids for large indoor exhibitions in Virtual Reality (VR), focusing on how they support orientation during continuous locomotion. The research addresses four main questions: how compact overviews like a mini-map help visitors stay oriented without clutter; how trails and heatmaps of visited paths support exploration and reduce repetition; how lightweight pin markers can capture and recall point of interest without complex menus; and how static You Are Here (YAH) signs at key entry points shape the first mental map that guides later movement.

Chapter 1 presents the motivation, derived from real exhibition experiences where visitors easily lose orientation, and formulates the research questions. Chapter 2 reviews prior work and extracting principles for unobtrusive navigation aids. Chapter 3 describes the hardware and software pipeline for implementation on a standalone Meta Quest Pro Head-Mounted Display (HMD). Chapter 4 details how the exhibition hall, mini-map, heatmap, pin system, and YAH boards were realized. Chapter 5 reports on-device observations, including performance checks and visual tests. Finally, Chapter 6 summarizes contributions, limitations, and future extensions.

The results show that a centered, on-demand mini-map reduces eye strain compared to static corner heads-up displays (HUDs); adjusting the heatmap to a world-space panel removed the double-vision effect seen on a flat canvas; and collider adjustments prevent unwanted physics when placing pins. YAH signs proved useful at short range but lose clarity beyond two meters. Performance tests confirmed a steady 72 fps on the standalone headset. These findings demonstrate that lightweight aids can improve orientation while preserving a sense of immersion. The thesis contributes a reference implementation, a reproducible deployment pipeline, and design guidelines for VR exhibitions. Limitations include the absence of a controlled user study, single-device evaluation, and narrow scene scope. Future work will extend to multi-user contexts, note-taking features, and analytics dashboards for designers.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Prof. Tullio Facchinetti, for his invaluable guidance, and patience throughout the development of this thesis. His insightful feedback and constant support were essential in shaping this work.

I am also sincerely grateful to my family, whose unconditional love and encouragement have sustained me during my studies and throughout my life. Their belief in me has been a source of strength and motivation.

Special thanks go to my friends and colleagues, who have been by my side during both the challenges and the rewarding moments of this journey. Their companionship and advice made this experience far more enjoyable.

Finally, I would like to thank everyone who, in various ways, has contributed to my academic and personal growth. This thesis would not have been possible without their support.

Contents

List of Figures

List of Tables

Acronyms

UNIPV University of Pavia

Robolab Robotics Laboratory

VR Virtual Reality

WiM World-in-Miniature

YAH You Are Here

HMD Head-Mounted Display

URP Universal Render Pipeline

UI User Interface

FOV Field-of-View

XR Extended Reality

GPU graphics processing unit

HUD heads-up display

Chapter 1

Introduction

Virtual Reality is now a solid way to rebuild and explore big indoor places like exhibitions, museums, supermarkets, and trade fairs. In these places, navigation is not just moving from point A to point B. It shapes what visitors notice, how they compare options, and what they finally choose. At the same time, these spaces are often large and crowded. People can lose their sense of direction, walk in circles, or rely on tools that feel intrusive and break immersion. We need navigation help that works quietly in the background-useful, but not distracting. This project was inspired by my own experience of feeling lost at a real exhibition: I kept passing the same corridor without meaning to, and I realized how a few well-designed cues could have helped.

Some popular techniques-like teleportation or World-in-Miniature (see Figure 1.1)-are powerful, but they change the natural rhythm of exploration. Fast hops or tiny replicas can pull attention away from the full-scale scene. That may be fine for games, but it does not fit well with exhibitions or shopping, where slow, continuous movement and peripheral vision matter. Prior work also notes differences in workload and attention compared with continuous walking [12, 22, 20]. For these reasons, this thesis avoids WiM and teleport and focuses on lighter aids that sit well alongside continuous locomotion, following guidance on how to activate maps and keep visual load low [14, 23].

The goal here is to design orientation support that reduces confusion while keeping immersion intact. Looking further ahead, the vision is a platform where designers can upload their own indoor spaces, visitors can explore naturally, and even shopping can happen inside the scene. Static cues at key entry points-like a clear “YAH” sign-can also help people build an early mental map [11].

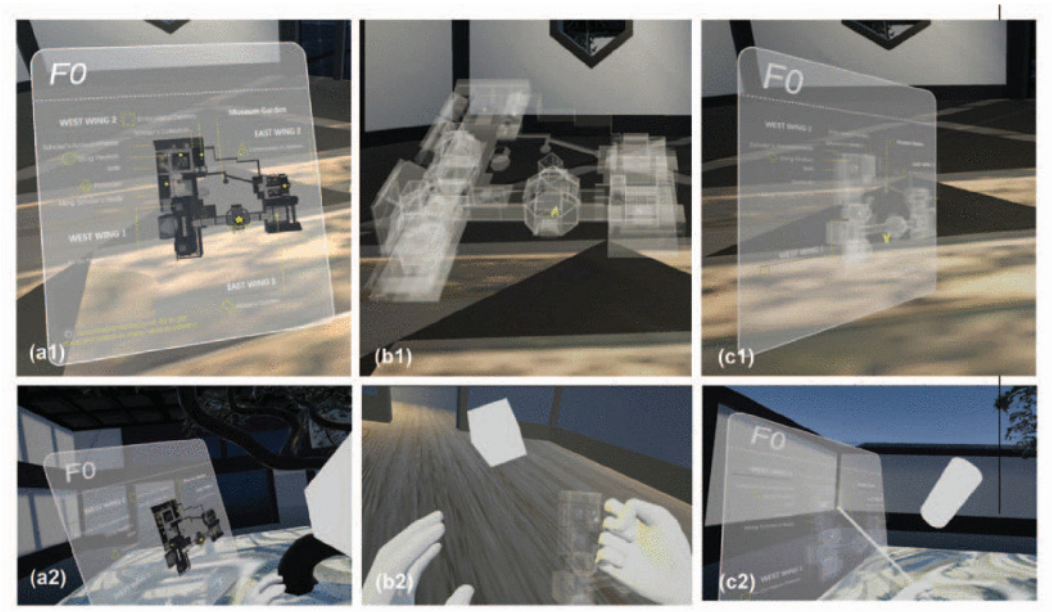

Figure 1.1: Frames illustrating three navigation methods in the VR museum scenario: (a1-2) holding and viewing a 2D mini-map; (b1-2) picking up and dropping the miniature model to teleport within WiM; and (c1-2) pointing and selecting on the WiM map to teleport [20].

1.1 Research Questions and Objectives

Based on this motivation, the thesis asks a set of practical questions.

- How can a compact overview help visitors stay oriented without demanding constant attention or creating clutter-especially when continuous walking is preferred over teleport [6]?
- In what ways can showing the path already taken support exploration in the moment and help reflection later, given evidence that trails and heatmaps improve coverage and reduce repeat visits [12]?
- Can simple, lightweight markers for points of interest support memory and planning without text input or complex menus, and still follow clear mini-map rules that avoid clutter [14, 23]?
- What is the role of static entry cues in shaping the first mental sketch that guides later movement [11]?

These questions lead to a clear objective: build a working prototype on a standalone VR headset that offers unobtrusive orientation support for indoor spaces.

The aim is not only to show that it can be built, but also to understand how such support interacts with presence, orientation, and satisfaction in exhibition-like scenarios. The prototype serves as both a testbed for interaction ideas and a base for future extensions, including multi-user use and e-commerce. Where relevant, we also consider findings about how much 3D mini-maps should show-only landmarks or a wider user-centered context-and how that affects spatial learning [9].

1.2 Contributions of this Thesis

The first contribution is a design stance: the prototype aligns orientation support with continuous locomotion instead of replacing it. By pairing help with the user's natural movement, the system aims to reduce confusion without hurting the parts of VR that make full-scale spaces feel real. This view complements studies that show both the strengths and the trade-offs of WiM and teleport compared with lightweight overviews [20, 6].

The second contribution is the method. Design choices are based on prior evidence, not just convenience. Rather than chasing novelty, the work collects proven ideas-clear overviews, careful on-demand activation, and visual economy-and carries them through to specific interaction and visual decisions [14, 23]. At the same time, the design maps known benefits of mini-maps, trails, and heatmaps to the goals of better coverage and easier retracing [12].

The third contribution is feedback for authors and curators. Orientation aids can guide visitors and, if instrumented lightly, also produce navigation traces that help improve layouts. Simple logs can yield coverage views that show which areas attract attention, which corridors get overused, and where visitors slow down-echoing how prior systems used path and heat distributions [12, 20]. In this way, the same tools that help visitors can also help designers build better spaces.

1.3 Structure of the Thesis

The rest of the document moves from background to implementation and then to evaluation.

- Chapter 2 reviews related work on orientation support in VR-compact overviews, path-based methods, lightweight annotation, and static cues - and extracts principles for design [12, 14, 23, 9].

- Chapter 3 presents the technical base: the hardware and software choices that make a clean, standalone build possible.
- Chapter 4 describes the system itself and explains how the aids were implemented in practice, from interaction logic to visual display.
- Chapter 5 reports the prototype's functional verification and discusses strengths and limits in exhibition-like navigation.
- Finally, chapter 6 reflects on what has been achieved and suggests where the platform could grow next.

Concretely, we sketch a web workflow where designers can upload and manage their indoor scenes; we propose lightweight, tablet-style note-taking as a first-class feature, taking cues from iVRNote, which shows how pen-and-tablet can work well with immersive content [3]; and we outline additions for collaborative exploration and in-context shopping that keep interaction inside the scene.

Chapter 2

State of the Art

This section examines how people find their way inside large VR spaces such as museums and exhibitions while helping the user remain present in the scene. Most studies reach a simple conclusion: basic orientation aids perform better than having no aid. A *mini-map* provides a small top-down view of the layout with user's position and facing direction, so it is easier to plan the next move and recover when disoriented. A *trail* draws a thin line where the user just walked, which makes the recent path explicit and helps avoid unintentional loops. A *visit-trace heatmap* gradually colors the areas already visited, providing a clear record of where has been covered and indicating areas still to be explored (see Figure 2.1).

Each tool supports a different aspect of the task, including spatial coverage, retracing, and memorability [12]. When locomotion relies on *teleportation*, which moves the user instantly to a chosen target without traversing the in-between distance, map-based aids often yield faster overall performance [6]. Eye-tracking shows a consistent pattern: the longer users stare at the mini-map, the slower they com-

Figure 2.1: Three orientation aids in a maze-like VR scene: left-a corner mini-map; middle-a path trail; right-a visit-trace heatmap. The latter two are rendered directly in the main scene and therefore use a "breadcrumbs" approach to show path history [12].

plete the task, so brief on-demand glances tend to work better than prolonged viewing [22]. In museum-like scenarios, interfaces that combine a quick overview with direct navigation are often rated easier and more usable [20]. Results can vary across settings and modalities, so designers should be careful when generalizing from one environment to another [18, 24].

2.1 Mini-map Systems in VR

Mini-maps are commonly described through a few design choices. Activation determines whether the map is always visible or shown on demand when the user toggles it. Alignment sets the frame of reference and can keep north at the top, rotate with the user's heading, or keep the player centered. The view can be a flat 2D map or a lightweight 3D inset that conveys height and depth. The widget can remain fixed in a corner of the display or move with the user as a portable panel. Content density can stay light with only key landmarks or become richer with labels, icons, and paths.

- *Activation/visibility.* Always-on maps can distract from the main view. Allowing the user to summon the map only when needed reduces confusion, and brief fade-in/out effects also help smooth the transition [14].

- *Alignment/framing.* Centered, player-focused views with clear cues that indicate which way is forward reduce placement errors. Surveys also highlight that orthographic projection is easier to read rather than perspective, and that circular crops reduce corner clutter and emphasize the region near the player. Together these settings provide practical defaults that many users find easy to read (see Figure 2.2) [23].

- *Dimensionality/detail.* Dimension and proportion controls how much of the display the map occupies, and surveys report that most mini-maps use about one to three percent of the screen, with far fewer cases above four percent [23]. Relatedly, Compared with full *WiM* views, which present a small 3D replica of the whole scene, lighter 3D mini-maps that focus on key landmarks and avoid clutter can help users learn the space. Adding a thin ring of local context around the user can improve precise *"where am I?"* placement without increasing time [9]. By contrast, fully detailed miniatures can cause occlusion and extra selection effort [20].

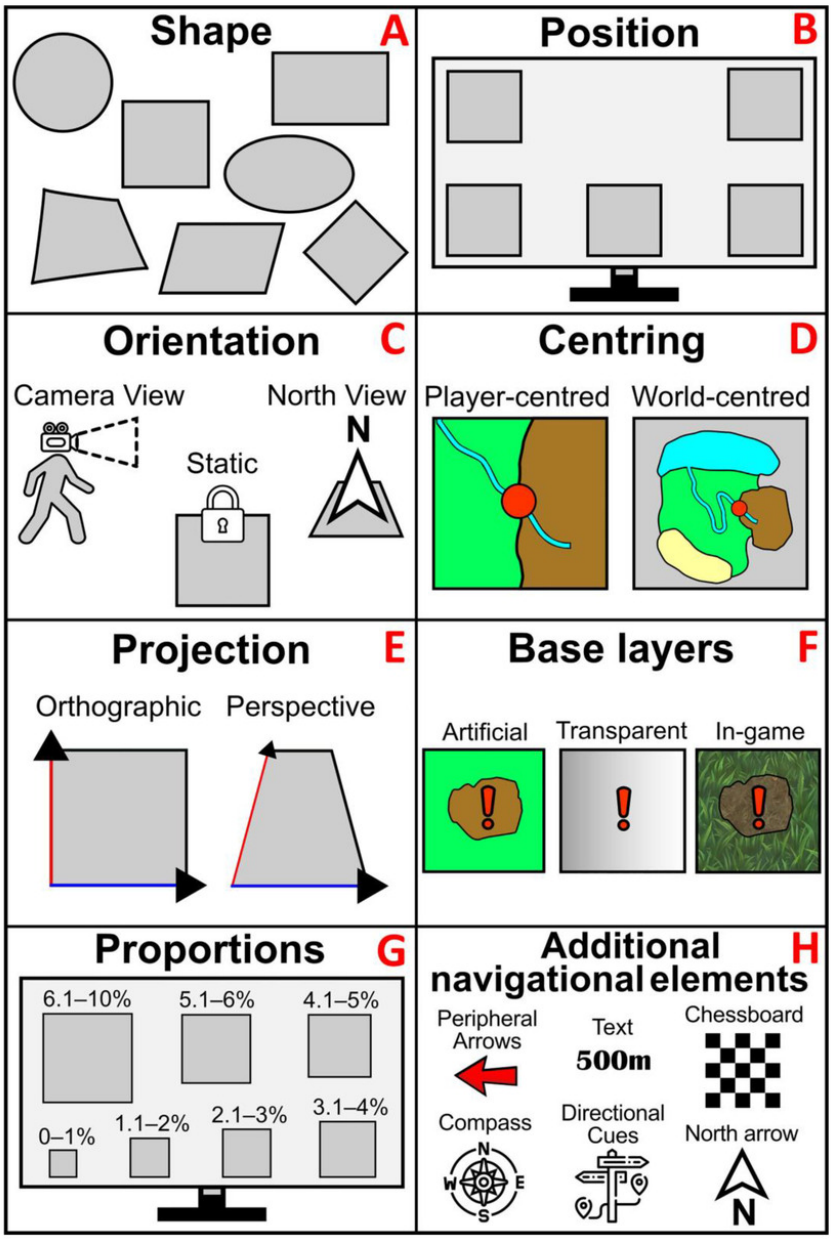

Figure 2.2: Mini-map features and design dimensions (A-H): shape, position, orientation, centring, projection, base layers, proportions, and extra navigational elements. This figure shows features a mini-map can have [23].

Mini map	A	B	C	D	E
Description	Hand-held map attached to the VR controller	Hand-held "dollhouse" attached to the VR controller	Static 2D GUI map in the corner of the FOV	In world navigational objects	Toggleable UI window map
Detailed description	2D top view map displayed on an 3D object in the virtual world attached to the VR controller (e.g., mobile device, paper map)	A 3D miniature version of the facility or environment (i.e., dollhouse) attached to the VR controller	Static 2D top view map attached to the peripheral or corner of the display. Frequently used in desktop gaming applications	Navigational objects and maps added as static objects in the VR environment (e.g., 3D signs or site maps)	Toggleable UI window in front of the user containing a 2D top view map. Commonly used menu interface in various VR applications
Reference	(Kuo et al 2022)	(Horbinski and Zagata 2022)	(Badr and De Amicis 2023)	(Lee et al 2022)	(Hou et al 2021)
Example from Literature reviews					
Developed demonstrator and its design features	 Portable 2D Tangible object	 Portable 3D Tangible object	 Non-portable 2D Non-tangible object	 Non-portable 2D Tangible object	 Portable 2D Non-tangible object

Figure 2.3: Mini-map taxonomy by portability and tangibility [1].

- *Portability/collaboration.* In team or industrial tasks, portable maps such as the one in Figure 2.3 often lower mental and time demands, and improve users' understanding of the layout. In large worlds with frequent teleportation, simple panel-based 2D maps can even outperform free-floating WiM on time and precision (see Figure 1.1) [1, 6].

In summary, a light configuration is recommended: the map should be shown only for short moments, visuals should be kept simple, and the alignment should match what users expect. Extra 3D detail is mainly useful when views are blocked or when very exact selection is needed [9, 20].

2.2 Trails and Visit-Trace Heatmaps

There are two common ways to show a user's history. First, *live aids* in the world-such as trails of breadcrumbs and on-floor coverage heatmaps-make visited regions obvious and can speed exploration and target finding (see Figure 2.1) [12]. When users mostly need a big-picture overview, mini-maps usually beat floor-based visuals for memorability and route planning.

Second, some systems keep history on the panel itself: session traces are drawn over the mini-map so users can quickly ask *"have I already been there?"* without adding any new geometry to the world [20]. For analytics after the session, position logs of paths and dwell time are converted into grids or heatmaps to study which spots attract attention, which corridors are busy, and how VR differs from physical visits; the same history can also support planning or redirection in locomotion control [7, 21, 5]. VR toolkits show how head/hand/eye logs become ready-made reports and 3D/volumetric heatmaps without cluttering the main scene [8]. Overall,

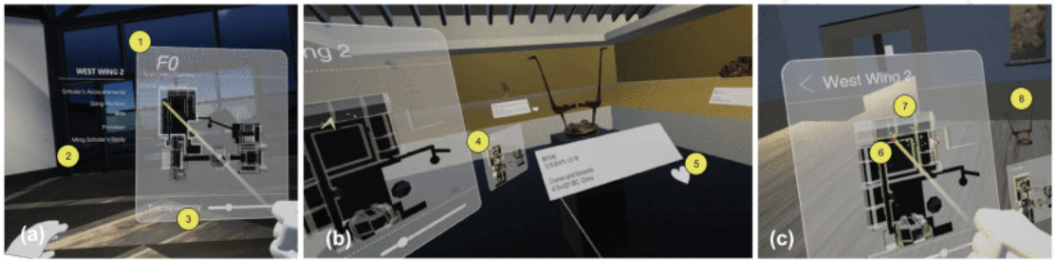

Figure 2.4: Marker interactions and hover preview. When the controller ray points to the object labelled 8 a preview panel labelled 5 appears [20].

history visuals work best as quick, glanceable summaries: persistent in-scene history competes with the environment, while panel-confined layers add context with low visual load[1].

2.3 Pinning and Lightweight Annotation

Pinning is a quick way to mark places of interest and it sits between navigation help and note taking. Pins act as lightweight placeholders that the user can revisit. Simple actions such as tapping to add a pin and selecting a pin and pressing Delete to remove it improve usability and keep mental effort low. Symbol-based markers use simple icons such as a heart, flag, or a star to convey meaning at a glance, and when the controller ray targets them, a small hover preview shows a tiny image or a short label (see Figure 2.4) [20]. Studies recommend setting a small cap on the number of pins and using consistent icons to reduce clutter. Surveys of mini-maps in games reach similar conclusions about simple, readable overlays [23]. VR note-taking systems also show that tablet-style annotations can live alongside immersive content and support later review (see Figure 2.5) [3]. Overall, pinning should be kept minimal, easy to undo, and guided by clear rules [5].

2.4 You Are Here Signboards and Static Anchors

A You Are Here (YAH) sign is a stable in-world anchor that designers often place near an entrance. Adding a simple YAH with a Field-of-View (FOV) arrow that shows which way the user is facing can improve placement accuracy and route recall without raising workload [9]. Using both a world fixed allocentric signboard that stays aligned with the building and an on demand egocentric mini map that stays

[1]Gaze-based heatmaps are out of scope here.

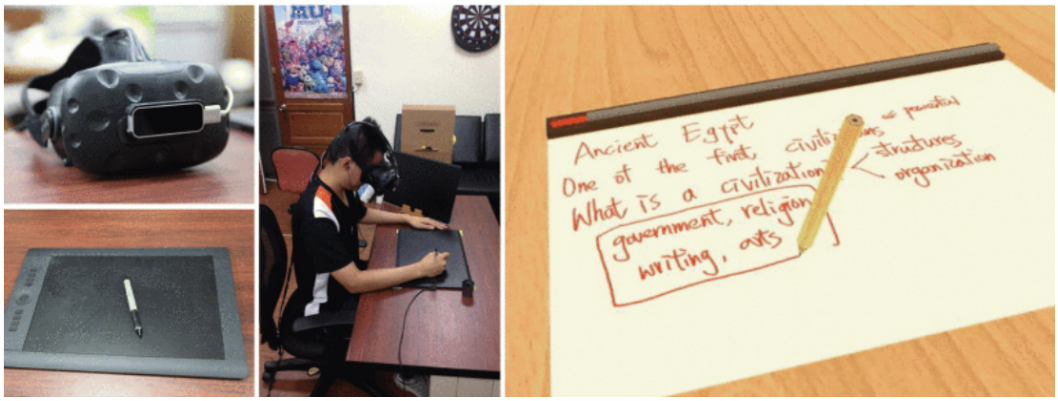

Figure 2.5: Tablet-style note taking [3].

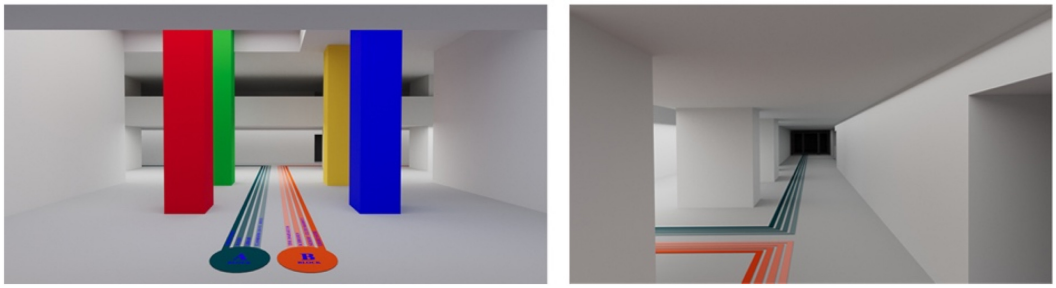

Figure 2.6: Route tracking with on-floor lines [11].

centered on the user's view, is linked to spending less time looking at the map [14, 9]. Early design work on signage reports that clear wall mounted signs are often faster and feel less restrictive than floor lines (see Figure 2.6). Furthermore, VR is a good place to test a design before building real signs [11]. In short, static anchors and transient overviews serve different moments: the sign helps initial orientation at first; the mini-map supports quick checks while moving.

2.5 Comparative Synthesis and Open Questions

In summary, three simple themes appear across studies.

1. Orientation aids beat no aid, though they help different aspects of navigation [12]. A mini-map supports overview, a trail supports backtracking, and a visit trace heatmap shows unvisited places.

2. Attention management matters. How maps are activated, how long they stay visible, and how they are aligned affects workload and comfort; spending more time looking at the map is linked to slower completion [22].

3. More detail is not always better: dense or always-on views can raise distraction and occlusion costs [9, 14].

On the contrary, some questions remains unanswered. Several comparisons are run with teleportation rather than continuous, joystick-style movement. For example, Di Domenico et al. evaluate mini-maps under teleportation [6], so there is less evidence about brief, on-demand maps during continuous locomotion. Standardized tests are still needed to directly compare between history drawn on the floor and traces kept on the panel, especially for how these choices affect presence and scene understanding [12, 20]. Evidence for pinning with simple rules is limited compared with richer annotation tools [23, 3]. There are still no systematic studies on how a fixed in-world YAH signboard and an on-demand mini-map work together during continuous locomotion [9, 11]. Finally, as tasks, measures, and environments differ across papers, results are hard to compare. More uniform study setups and standard ways to handle transitions, such as toggling the map, would help [13].

Chapter 3

Tools and Frameworks

This chapter explains the hardware and software used in the project and why they fit the goal. The usage model is simple. The visitor remains in place, either seated or standing, while a wireless setup allows free movement inside the virtual scene through the use of the controllers. This configuration makes it possible to explore the exhibition stands without the limitations imposed by cables or external sensors. To maintain safety in any space, The guardian stays on as a safety net, showing virtual boundaries when the user gets close to real-world obstacles. However, room-scale walking is intentionally disabled. Section 3.1 outlines the hardware that makes this possible in various spaces and Section 3.2 briefly notes that the environment and objects are created in Blender, then assembled and scripted in the Unity game engine to become a playable experience.

3.1 Hardware

The prototype requires sufficient hardware setup for both execution and development. In this case, the laptop plays a vital role to allow rapid cycles of editing, compiling, and testing. Once the final version of software is complete, the HMD provides the core immersive view. then dedicated controllers handle locomotion and interactions. Consequently, these elements ensure that the system can be deployed reliably in different spaces.

3.1.1 Development Workstation - Laptop

Builds and profiling are carried out on a laptop to keep the process of editing, running, and fixing as short as possible. The laptop is not the target renderer; it only affects build speed and how quickly we test changes. The machine used was an

Figure 3.1: The ASUS Zenbook S 13.

ASUS Zenbook S 13 Flip OLED equipped with an Intel Core i7-1260P processor, sixteen gigabytes of RAM, and Intel Iris Xe integrated graphics (see Figure 3.1). This configuration proved powerful enough to run the Unity editor smoothly and to shorten compilation times so development could proceed efficiently.

3.1.2 HMD: Meta Quest Pro

The application runs natively on Meta Quest Pro, meaning it is installed and executed directly on the headset without requiring a PC or external streaming (see Figure 3.2). This, in turn, keeps the setup simple because there are no cables or base stations, which makes the system particularly suitable for exhibition contexts. The application runs smoothly on the HMD, and on-device checks confirm the clear visuals and the controller inputs respond as expected. Studies on exhibition VR also support using wearable HMDs for better immersion and stable comfort compared with mobile phone based VR solutions [10, 4].

3.1.3 Controllers: Touch Pro

Touch Pro controllers give continuous forward and backward motion, sideways movement, and smooth turning on the analog sticks (see Figure 3.3). While the sticks

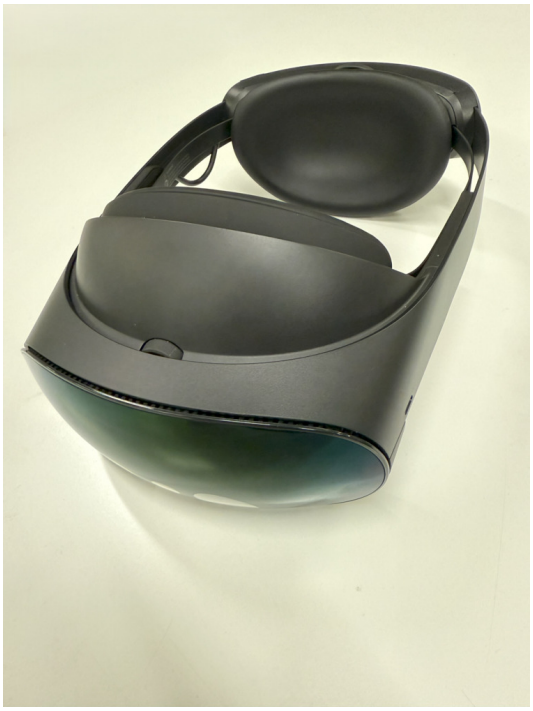

Figure 3.2: The Meta Quest Pro headset.

handle movement, the other buttons activate short actions such as opening the minimap, displaying the heatmap, or placing and removing pins. As all locomotion is handled virtually, the behavior is consistent no matter what physical space the user occupies. Furthermore, light haptics feedback confirm discrete events without drawing attention away from the main task.

3.2 Software

This section explains the software side. How models are prepared, how the virtual space is assembled, and how editing and testing are kept efficient. Blender is responsible for modelling and asset preparation, while Unity manages scene composition, interaction, and execution. The following subsections explain the contribution of each tool and how this environment is brought to life.

3.2.1 Content Creation: Blender

Blender is used to model the exhibition environment and prepare assets with clean shapes, a consistent scale where one unit equals one meter, and correct pivot points (see Figure 3.4). Textures are unwrapped and the number of materials is kept low

Figure 3.3: The Touch Pro controllers.

so the standalone headset renders smoothly. Static objects are sometimes combined to make rendering faster, while collision shapes are kept separate to ensure stable movement. The models are exported in FBX format with applied transforms and consistent axes so they import into Unity without any problem. Blender is also free and open-source, which makes it an accessible tool for this project.

3.2.2 Engine: Unity

Unity is the main engine for the scene and interactions, chosen because it has a large community, extensive documentation, and strong support for VR development (see Figure 3.5). Since the contents are already in 3D, the VR scene can be implemented using Unity's built-in packages and plugins, as described below:

- XR Device

  The OpenXR plugin is used within Unity to make the application compatible with different VR systems (see Figure 3.6). On top of this, the Meta XR SDK extends the functionality with features designed specifically for the Quest headset. The controls are kept simple: the sticks are used for continuous movement and turning, while the face buttons handle quick actions. This makes it easy for first-time users to learn the system comfortably. By relying on the

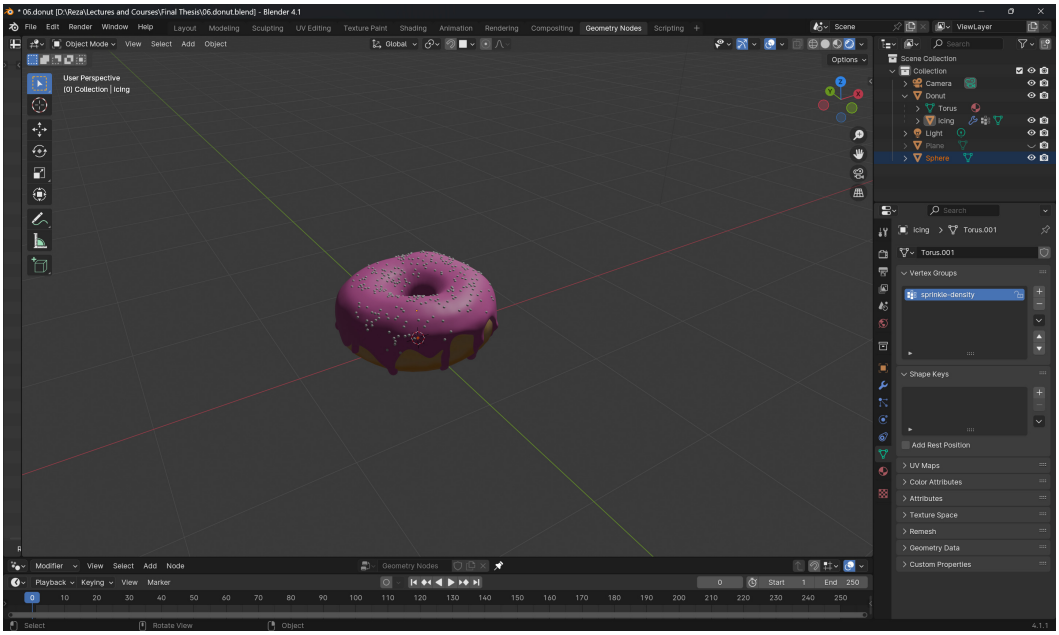

Figure 3.4: The screenshot highlights Blender as a powerful tool for 3D modelling.

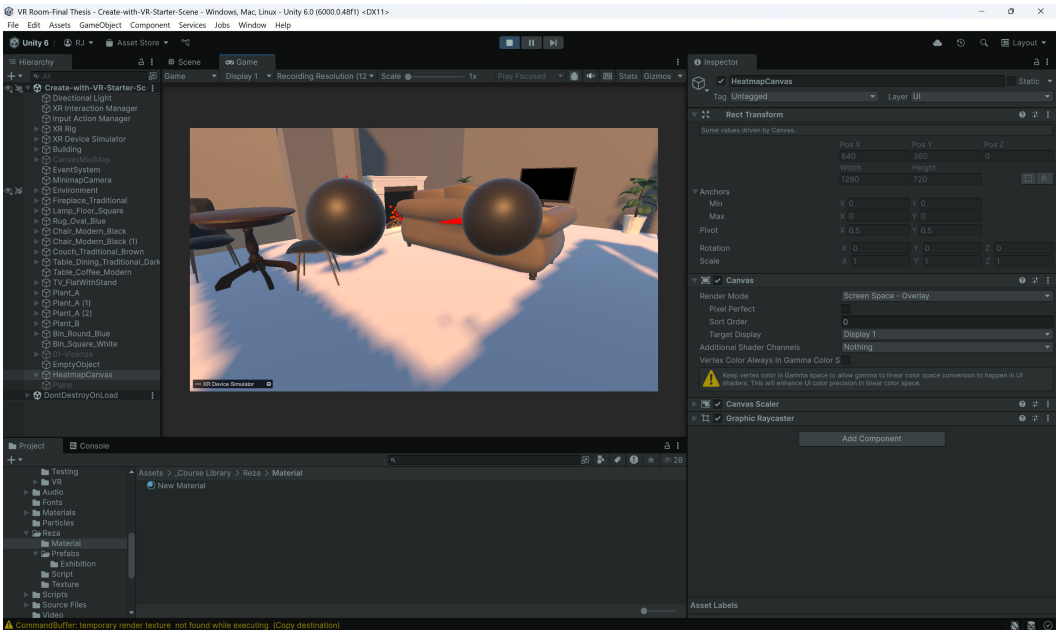

Figure 3.5: Unity in VR development.

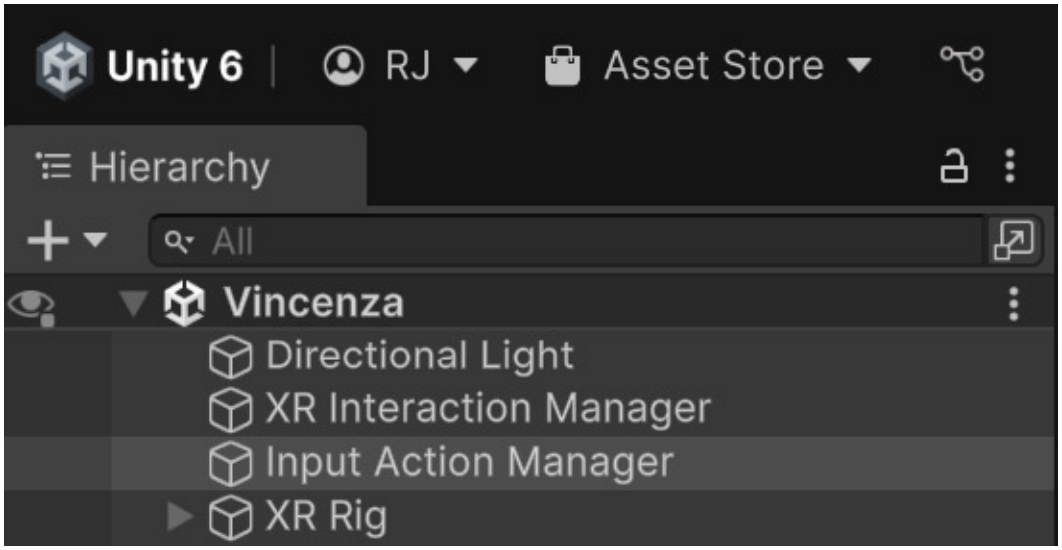

Figure 3.6: Unity project hierarchy showing the XR setup.

Meta XR SDK, the system remains open to future integration of advanced Quest Pro capabilities such as hand, eye and face tracking, and passthrough for mixed-reality features.

- Device Simulator

  Unity also provides a Device Simulator that allows the scene to be tested directly inside the editor with a keyboard and mouse instead of a VR headset (see Figure 3.7). This allows us to move and look around the environment on a laptop in the same way a headset user would, which makes quick checks much faster without always having to connect and wear the headset.

Ultimately, all iteration happens in the Unity editor using Play Mode on the laptop to keep the change-test loop short. The final version is built and then installed on the Quest to verify clarity, comfort, and steady frame rate on real hardware. The graphics are kept simple so panels like the mini-map, heatmap, the YAH and pins stay clear and responsive [6, 23, 15].

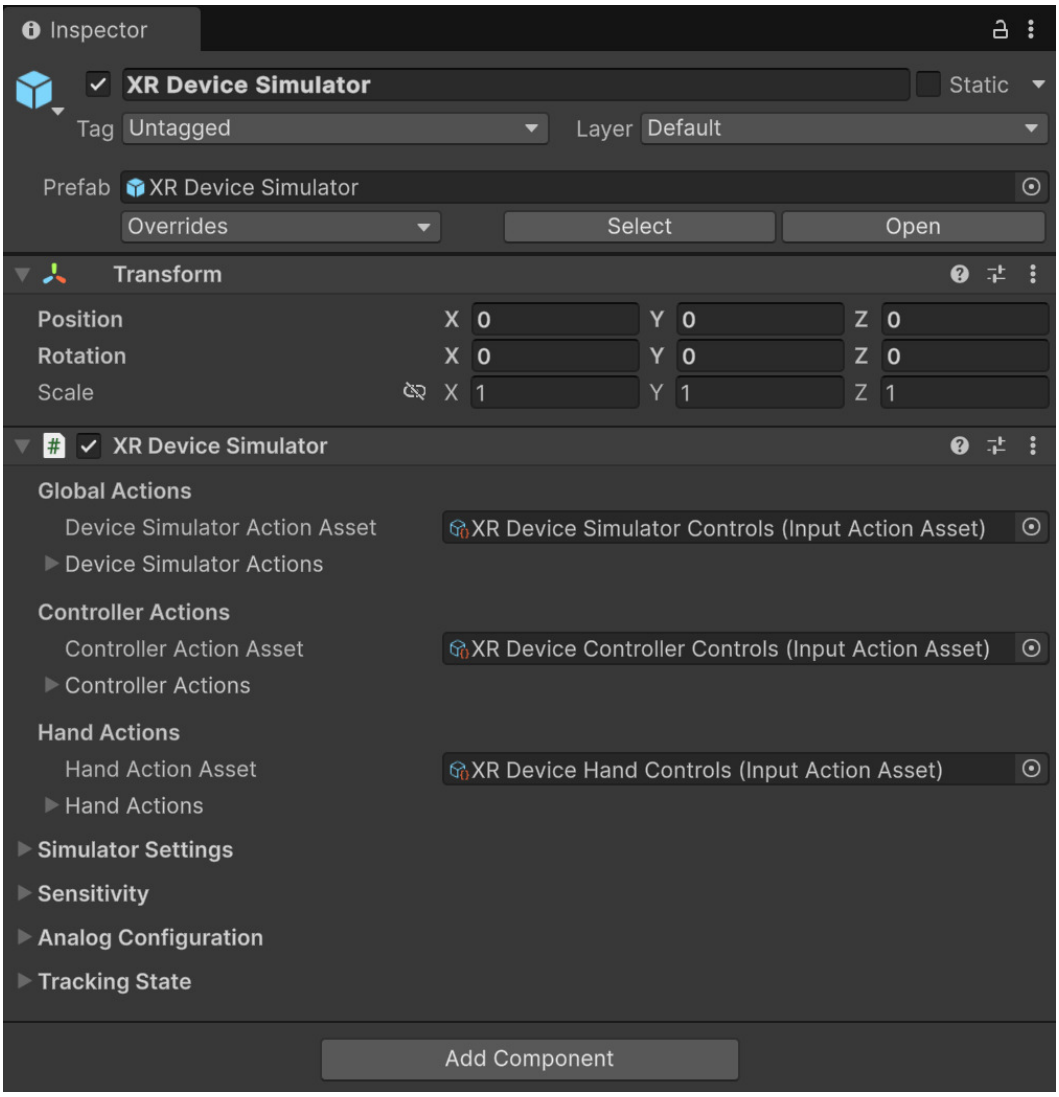

Figure 3.7: XR Device Simulator in Unity.

Chapter 4

Implementation

This chapter starts by explaining how the environment simulation is built from scratch and the challenges encountered along the way. Each decision is supported by evidence from earlier chapters and the reviewed literature, so the most defensible option is selected at each step. First, the exhibition hall is modelled in Blender; next, the exported assets are brought into Unity to implement YAH, pin system, mini-map, and heatmap.

4.1 Exhibition Hall Design

This section details the step-by-step modelling process in Blender and explains how to prepare the assets for Unity import. The process covers scene setup, scale calibration, tracing outer and inner walls, creating the floor and stages, cleaning topology and normals, and exporting FBX files with Unity-friendly axes.

4.1.1 Scene Setup and Scale Reference

Work begins by cleaning the Blender scene and removing the default cube so nothing unwanted affects later steps (see Figure 4.1). Units are set to Metric with Unit Scale equal to 1.0, which establishes the convention that one Blender unit equals one meter. The floor-plan image is imported in top-orthographic view, aligned to the global axes, and renamed for clarity, for example `Plan`, as shown in Figure 4.2. A simple 1×1 plane is added to act like a ruler to calibrate the drawing (see Figure 4.3). Using a known real-world length on the plan-an 89-meter wall in hall C-we scale the reference image until Blender units match the actual size (see Figure 4.4). After this step, one Blender unit truly represents one meter. Getting scale right here ensures that corridor widths, wall heights, and distances will feel natural inside the headset.

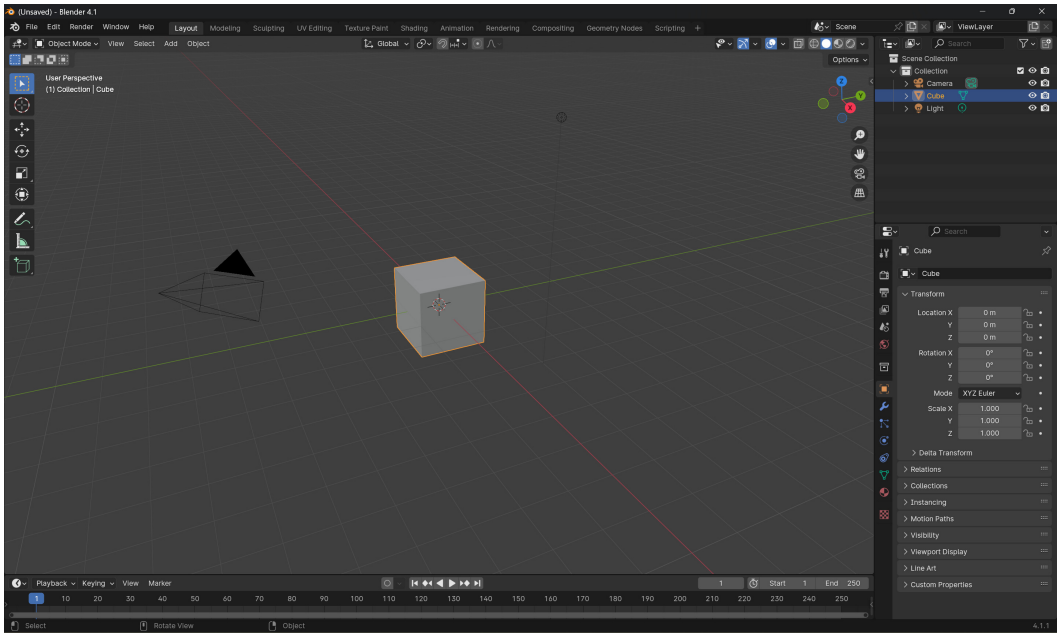

Figure 4.1: Initial Blender scene with the default cube.

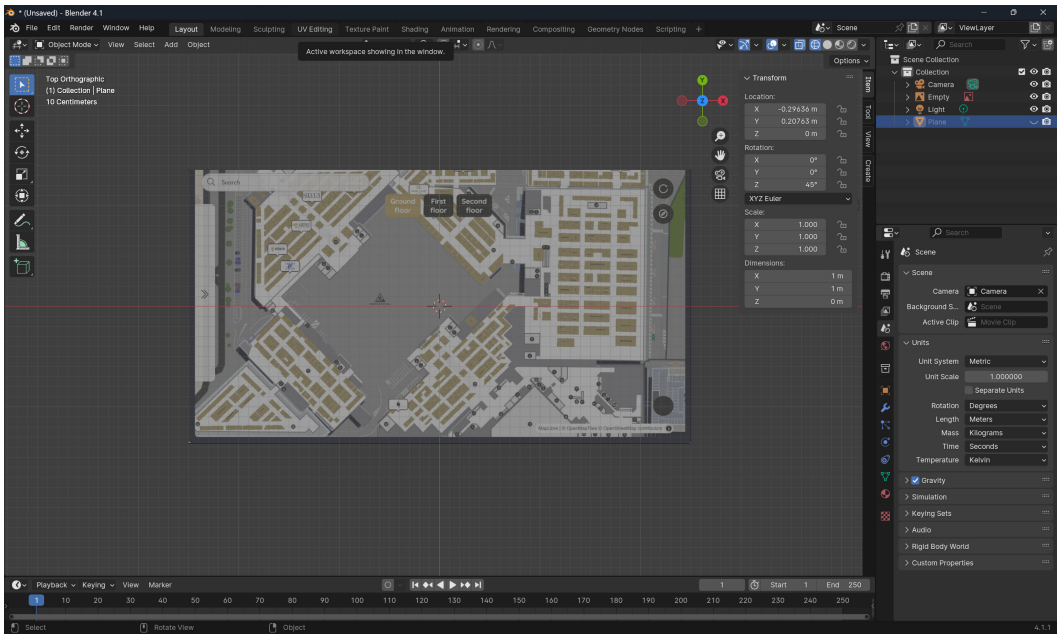

Figure 4.2: Floor plan imported as a reference and aligned to the global axes in the Top-Orthographic view.

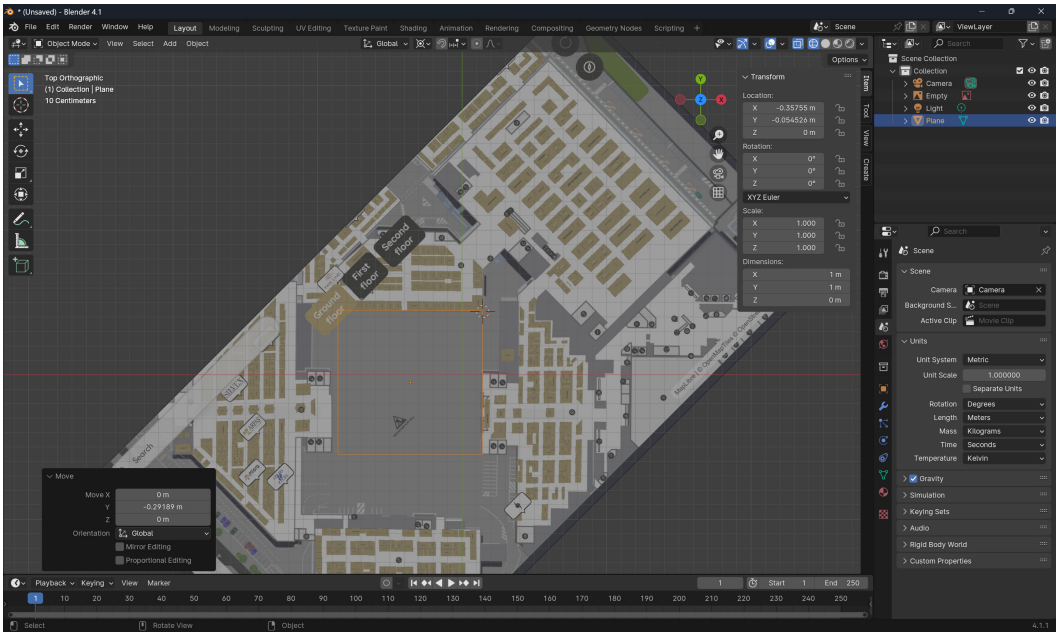

Figure 4.3: A $1 \times 1\,\mathrm{m}$ plane is added on top of the floor plan image to serve as a calibration reference.

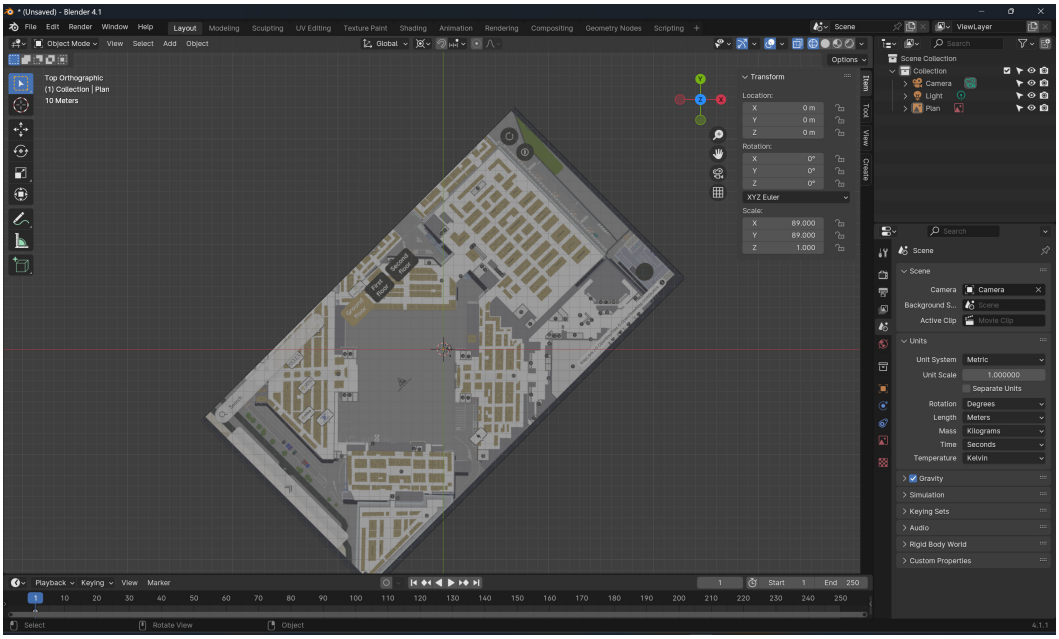

Figure 4.4: The floor plan image is scaled by a factor of 89.

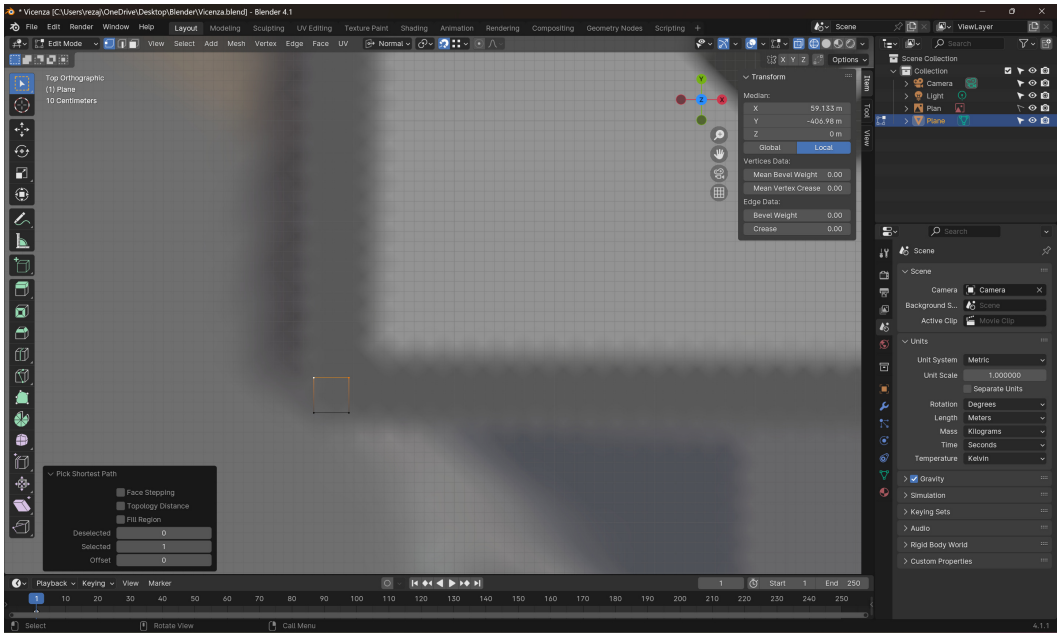

Figure 4.5: Beginning the tracing of exterior walls of exhibition with its actual thickness.

4.1.2 Exterior Walls

With scale fixed, the thickness of the exterior walls defined in the plan is traced directly in Blender, as shown in Figure 4.5 and Figure 4.6. The tracing continues around the perimeter until the loop is closed, with corners kept clean and simple so that the geometry remains light and easy to interpret. Vertices are then checked and merged by distance to remove tiny gaps or duplicates, which yields a single continuous loop for the exterior walls (see Figure 4.7). Finally, the drawing are extruded up to a realistic height of 7 meters to form the main volume of the halls (see Figure 4.8).

4.1.3 Floor, Stages and Partitions

Then floor is created as its own mesh: a new plane is snapped to the outer wall boundary and then extruded to a thin one-centimeter slab (see Figure 4.9). Keeping the floor separate makes UVs, materials, and collisions easier to manage later. With the base in place, the exhibition stages are traced from the plan as a new object and extruded to a height of four centimeters. To make them visually distinct from the floor, the stages are assigned a brown material, so they stand out clearly during navigation (see Figure 4.10).

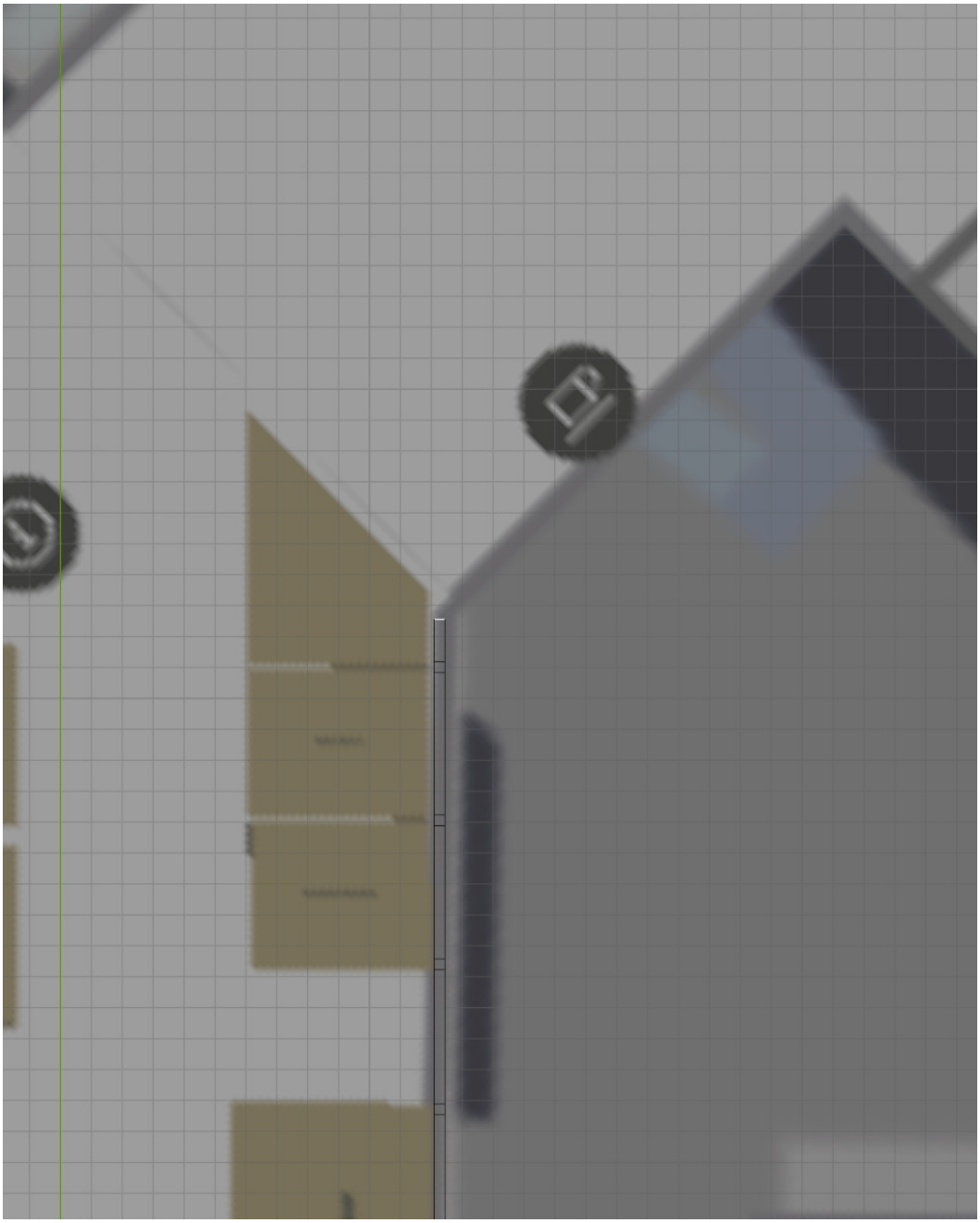

Figure 4.6: Continuation of the modelling process for the structural outer walls.

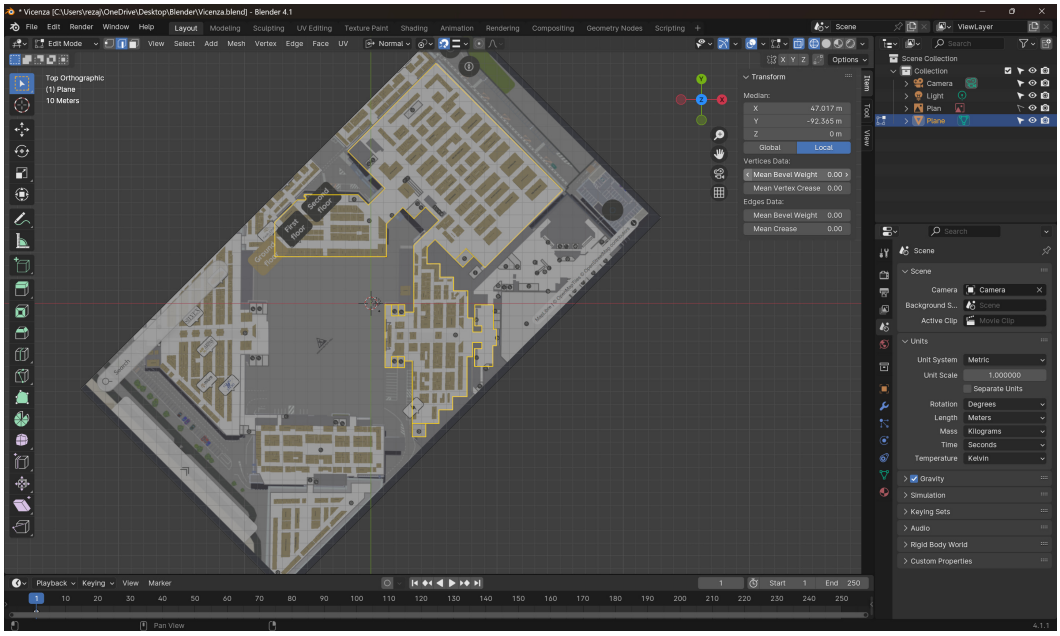

Figure 4.7: Completed tracing of the building's exterior walls based on the floor plan. At this zoom level the walls may look like a simple continuos line, but they actually have the intended thickness.

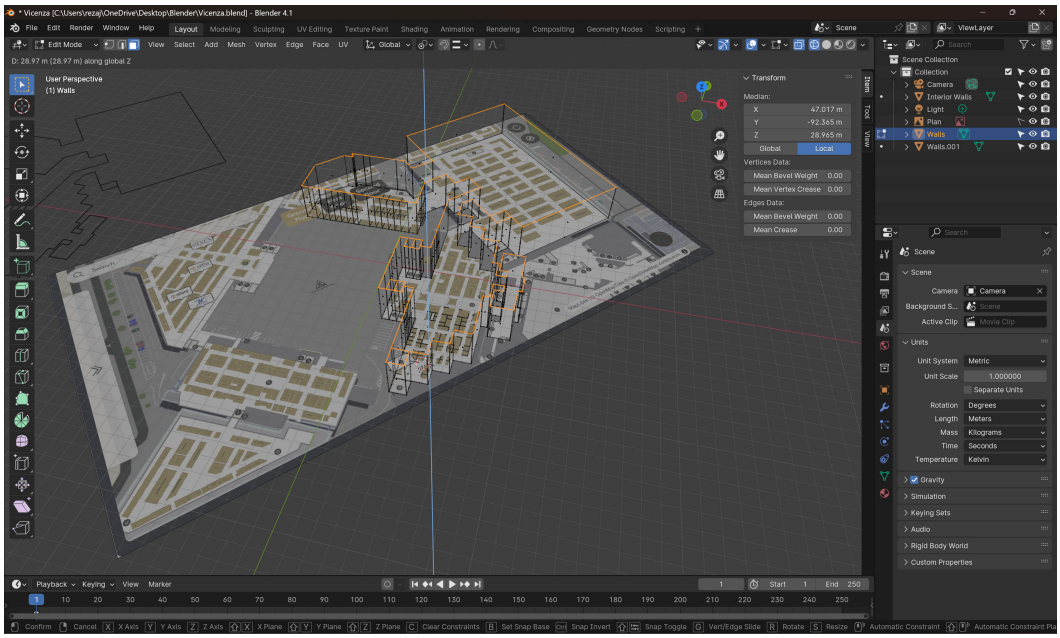

Figure 4.8: The traced exterior walls are extruded upward along the Z axis

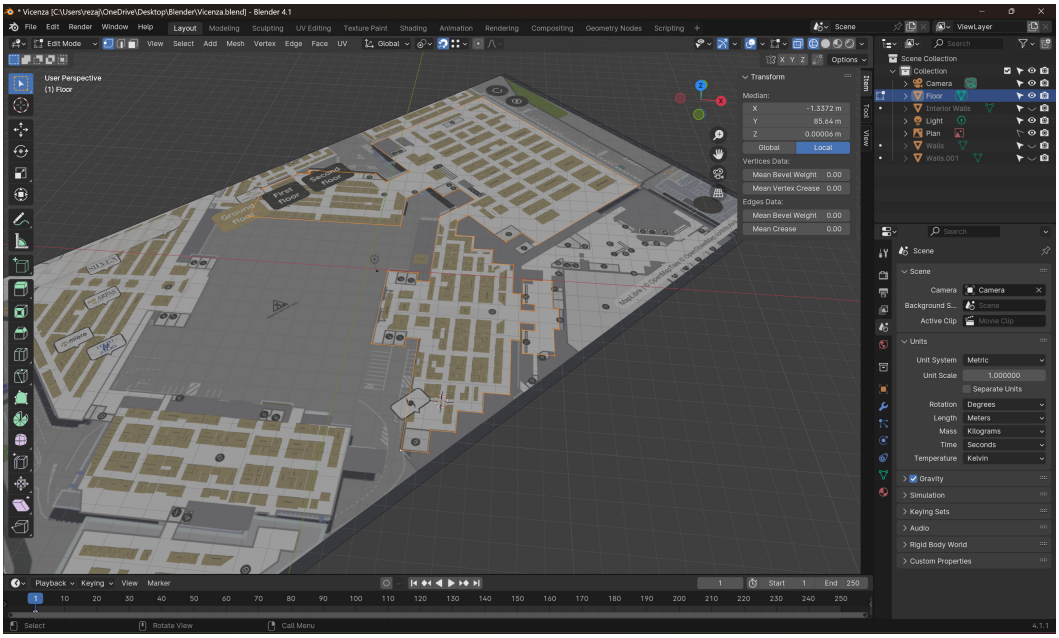

Figure 4.9: The floor is created as an independent object by adding a new plane.

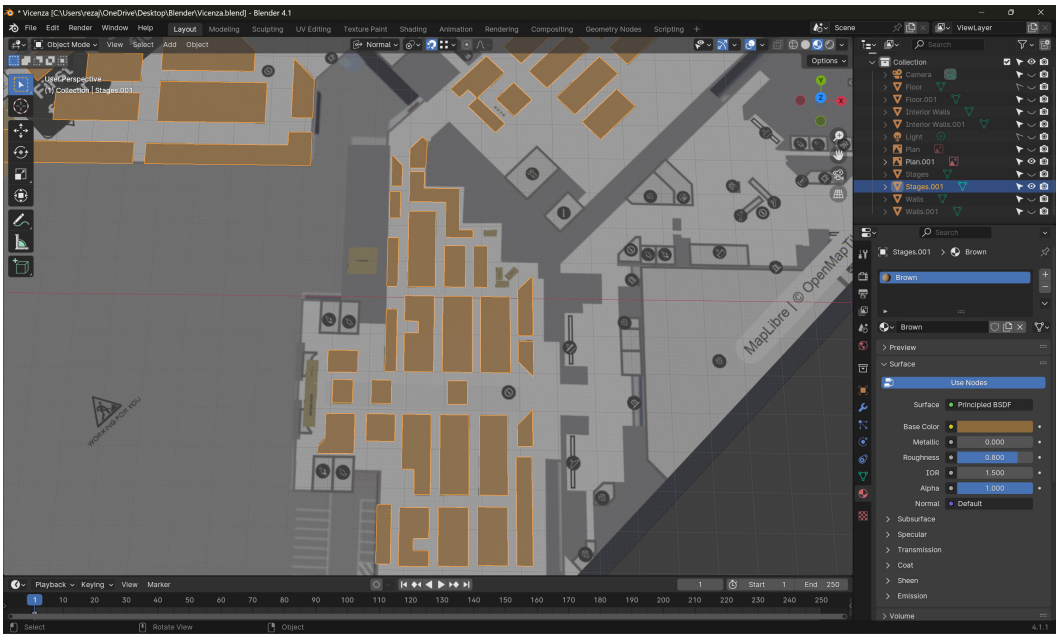

Figure 4.10: Exhibition stages created by tracing their outlines from the floor plan.

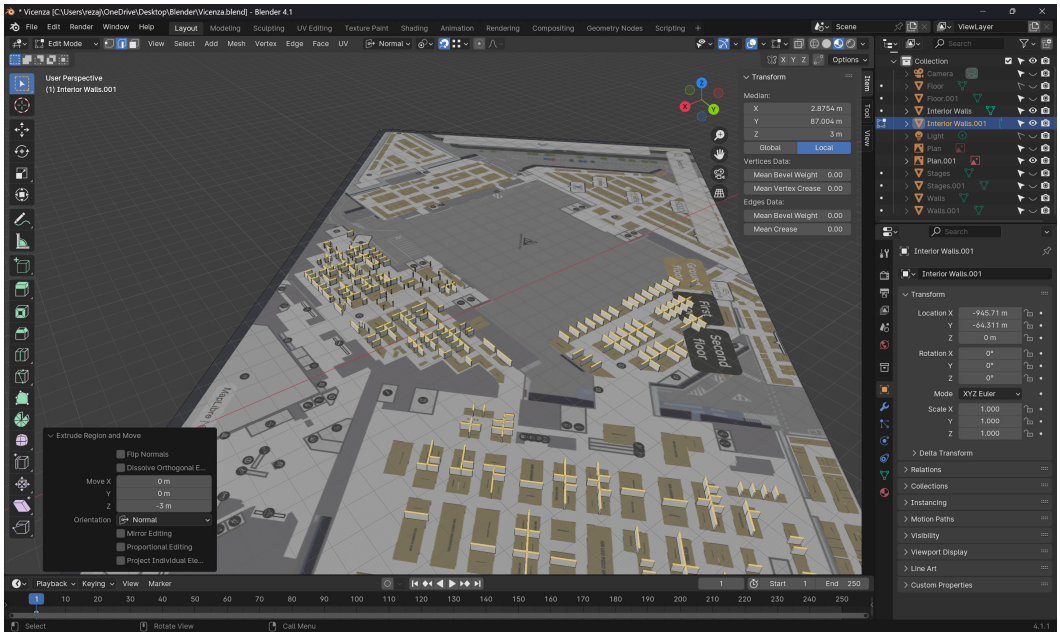

Figure 4.11: Interior walls and partitions.

Interior walls and partitions are then drawn by following the same rule and process. These three-meter-high elements divide the exhibition into individual booths so that the navigation structure matches the real layout while remaining efficient for rendering in VR (see Figure 4.11).

4.1.4 Recalculating Normals and Exporting

Before export, the mesh is cleaned by merging overlapping vertices, removing loose geometry, and ensuring that only simple quads and triangles remain. Normals are recalculated to the outside so lighting behaves correctly (see Figure 4.12). A clean mesh prevents shading artifacts and makes the Unity import predictable.

Using consistent names and well-structured collections makes the project easier to manage. The exterior and interior walls, floor, stages, and the plan reference are each placed in clearly labelled collections. Object origins (pivots) are set either on the floor or at the geometric center, so every element imports into Unity in the correct position without unwanted offsets. This organization also makes it simple to hide, show, export, and test parts independently.

At the end, all transforms are then applied so scale, rotation, and location are baked into the mesh. The FBX export uses Unity-friendly axes (*-Z Forward*, *+Y Up*) so objects import without unwanted rotations (see Figure 4.13). The entire environment is then exported as a single FBX file that preserves the Blender collections

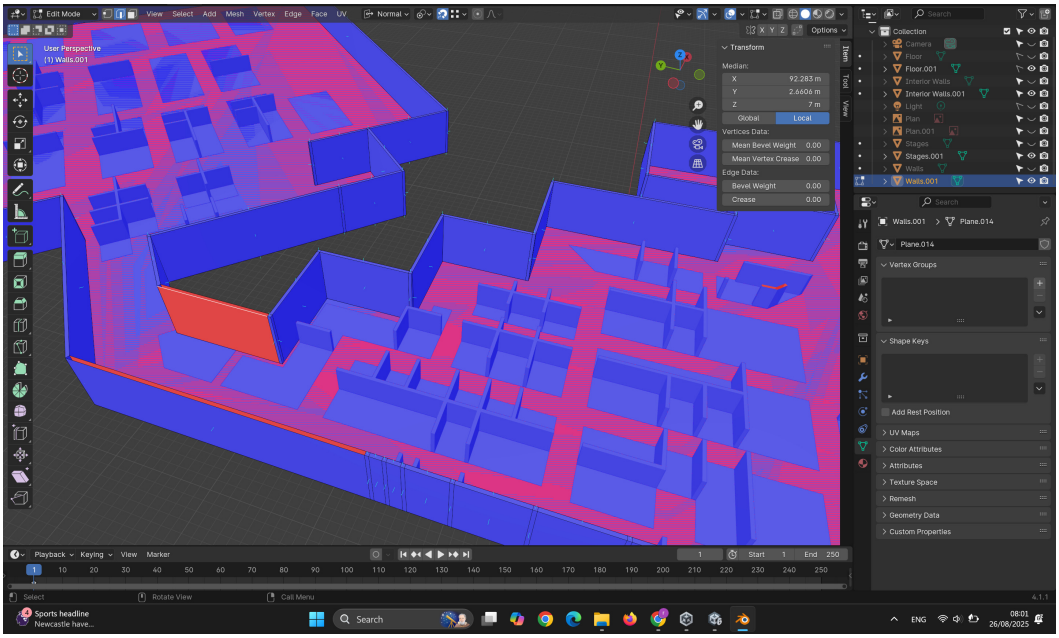

Figure 4.12: Checking and recalculating normals. In this view, a red face can be seen on one wall, indicating that its normal was flipped in the wrong direction. Such issues are corrected by recalculating normals.

as grouped elements—exterior walls, interior walls, floor, and stages-so each part appears as a distinct child in the Unity hierarchy. This keeps the scene organised and allows each layer to be modified during testing process. It also simplifies re-import, since one source file updates all grouped elements consistently and reduces the chance of drift or misalignment across assets.

4.2 Unity Setup

A new Unity project is created, where the first scene only contains the *Main Camera* and *Directional Light* (see Figure 4.14). The project is configured to use the Universal Render Pipeline (URP) with simple lighting and without heavy post-processing, ensuring stable frame rates on standalone headsets.

At this point, *OpenXR* and the *XR Interaction Toolkit* are enabled. OpenXR is a common standard that allows Unity to work with different VR headsets through the same interface, without needing device-specific code. The XRI is Unity's ready-made toolkit for basic VR interactions such as moving with the joystick, turning the view, grabbing or pointing at objects.

From the Unity menu, the *XR Origin (VR)* prefab is instantiated (see Fig-

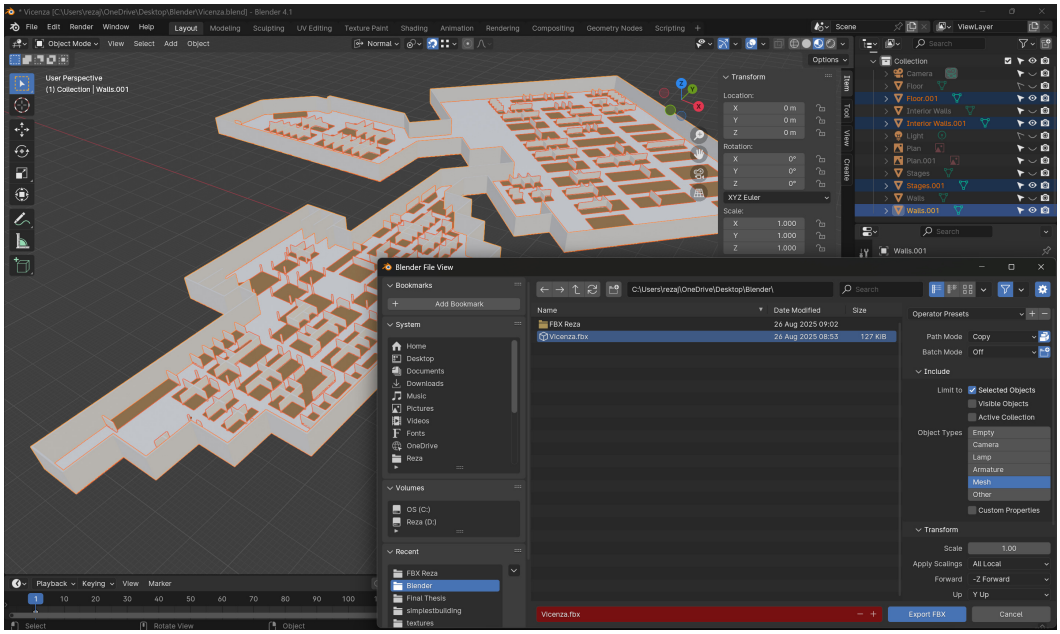

Figure 4.13: Exporting a modular FBX from Blender.

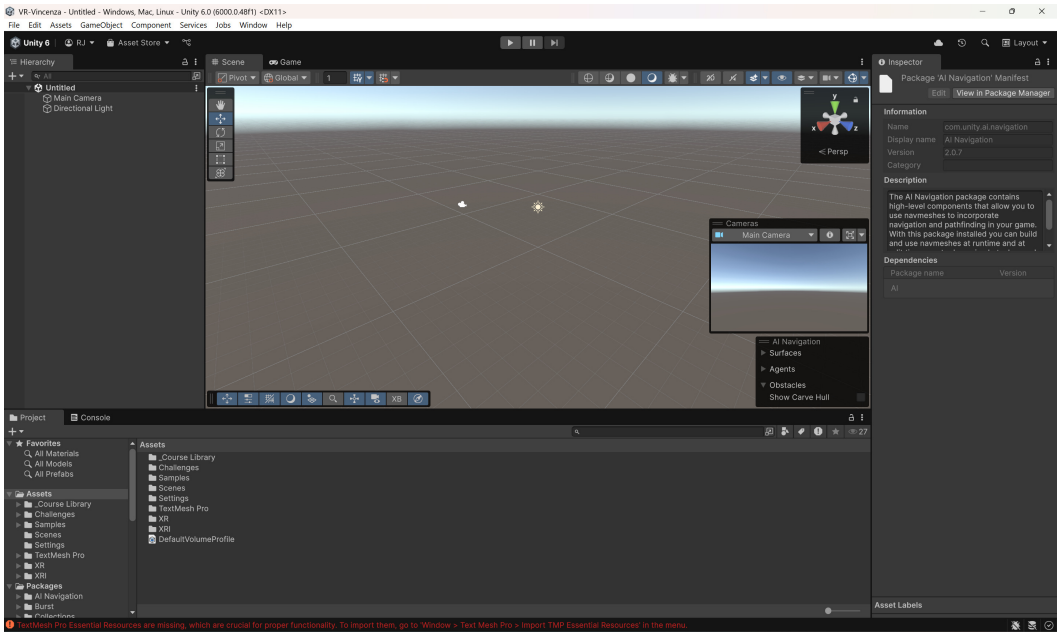

Figure 4.14: The first Unity scene.

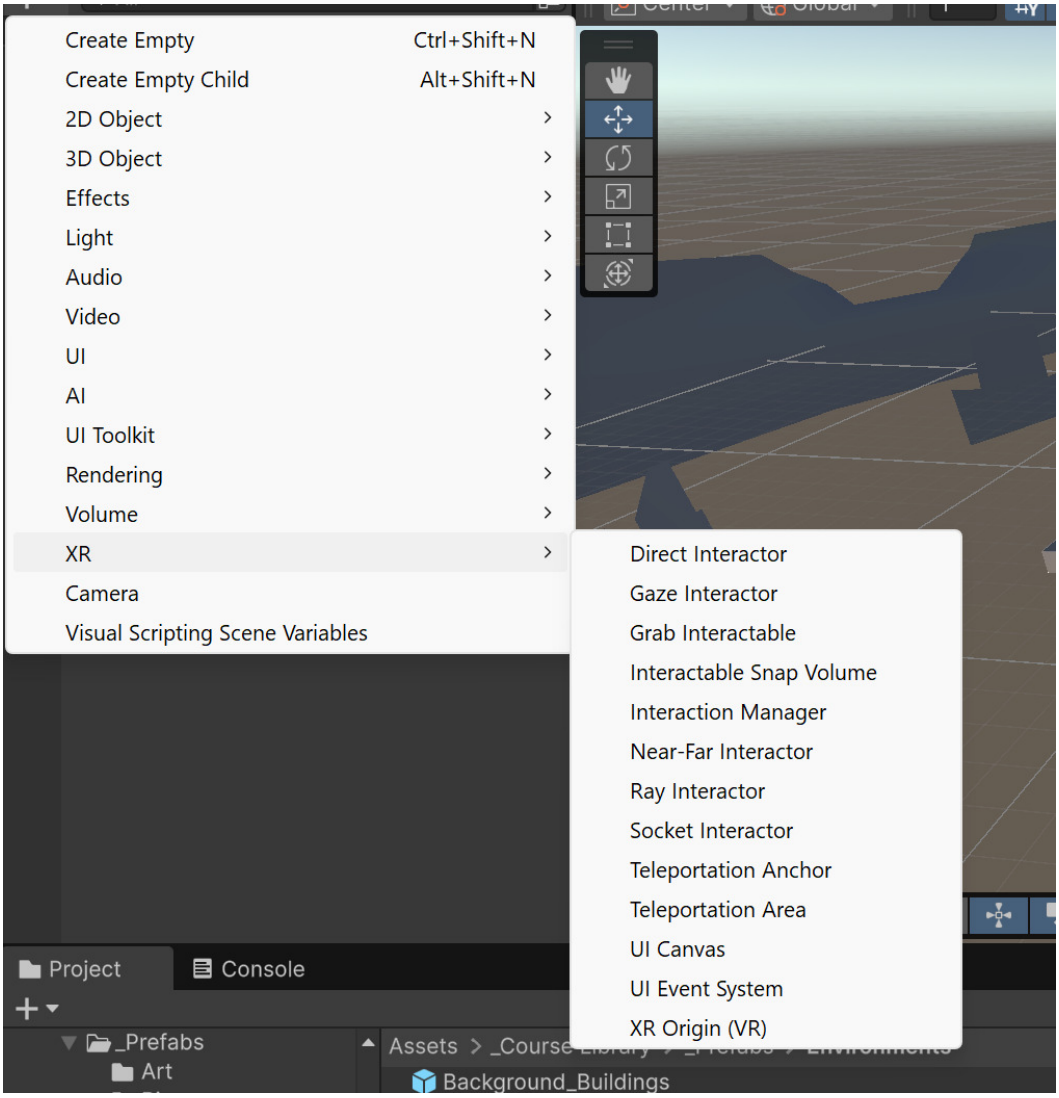

Figure 4.15: XR Interaction Toolkit.

ure 4.15). This prefab sets up the default VR rig-a main camera for headset, left and right hand controller objects, and managers for input and interaction (see Figure 4.16). The rig is configured for continuous move and continuous turn so that the joysticks handle locomotion and rotation smoothly. This baseline confirms correct tracking and input work correctly.

Blender models are then imported and kept at real-world scale which means one Unity unit equals one meter. In a glance, the exhibition hall appears inside Unity (see Figure 4.17), confirming that the FBX import worked and that sizes match the plan . Next, simple background prefabs are placed around the hall to avoid an empty horizon and give the scene depth without touching the indoor layout

Figure 4.16: Hierarchy view in Unity after adding the *XR Origin (VR)* prefab.

(see Figure 4.18). Finally, the XR Device Simulator is enabled so the scene can be tested with mouse and keyboard: the two black spheres act as the left/right hand controllers, and the thin red rays show their pointing direction, allowing movement and turning to be checked even without a VR headset (see Figure 4.19).

4.2.1 YAH

The YAH board is made in Unity as a simple *Quad* with a floor-plan texture applied to an *URP/Unlit* material. The Unlit shader keeps the colours and lines of the map always the same, without being affected by lights or shadows, so the text stays clear and easy to read (see Figure 4.20). To avoid forcing the user to reorient, two versions of the board are placed back-to-back: one facing inward and one outward. This ensures that, regardless of approach, the sign can be read directly (see Figure 4.21). The quad is rotated so the map's "up" matches the real-world orientation of the hall, and the front side faces the intended viewer. Placing a YAH near the entrance gives visitors an early and stable point of reference, which helps them understand the layout and remember their position later [15, 6, 12]. In addition, when corridors are especially long and users are more likely to feel disoriented, extra YAH are added at those points to restore orientation and reduce confusion [20].

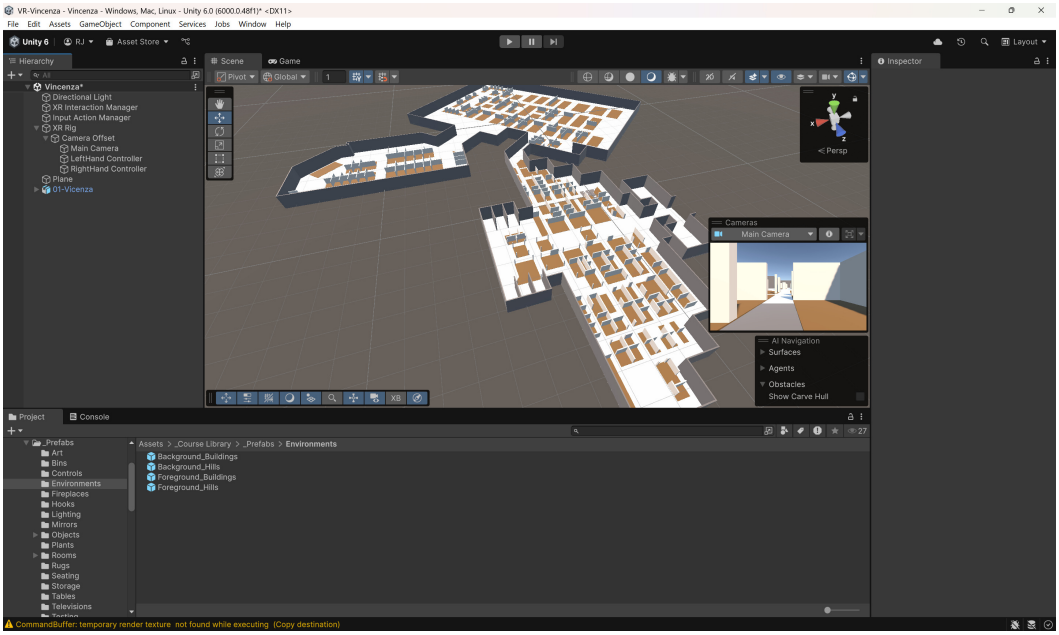

Figure 4.17: The FBX model of the exhibition hall in Vicenza was imported at real scale.

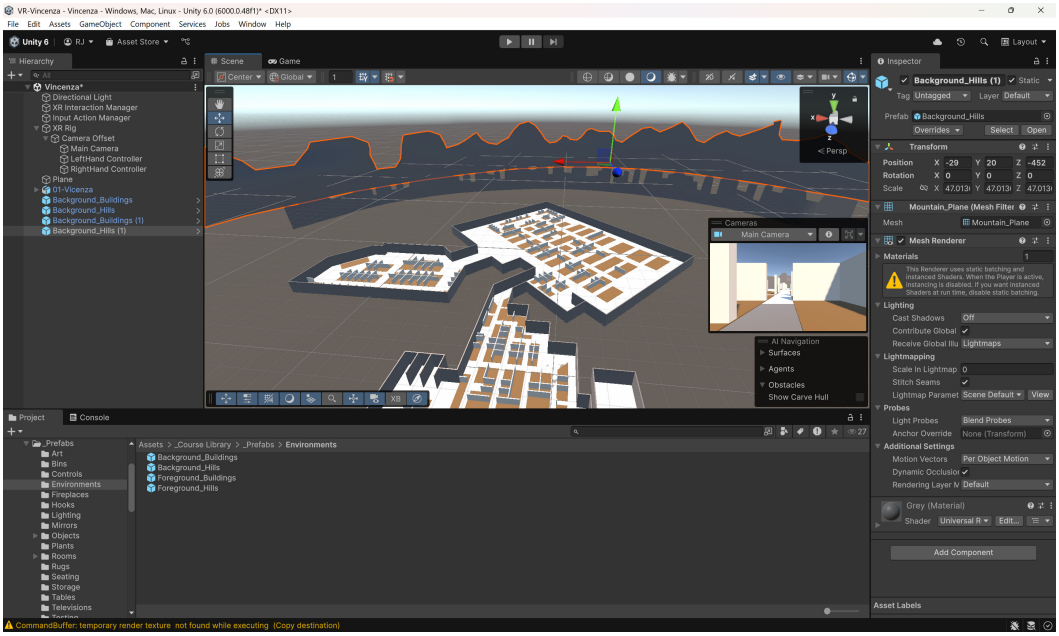

Figure 4.18: Background environment prefabs placed around the hall.

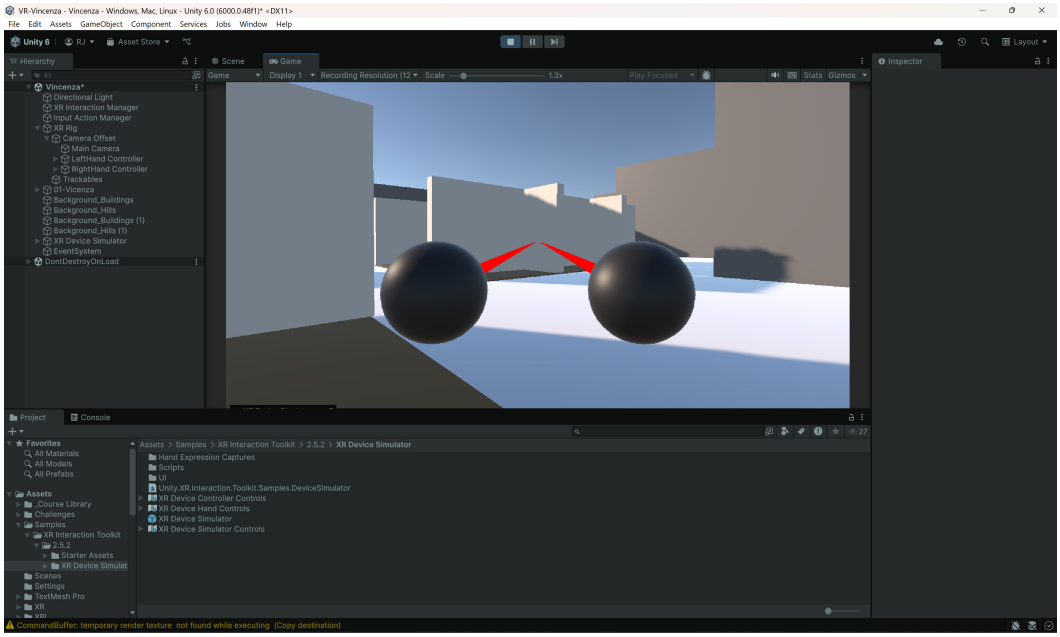

Figure 4.19: Game view with the XR Device Simulator enabled to test the simulation using mouse and keyboard.

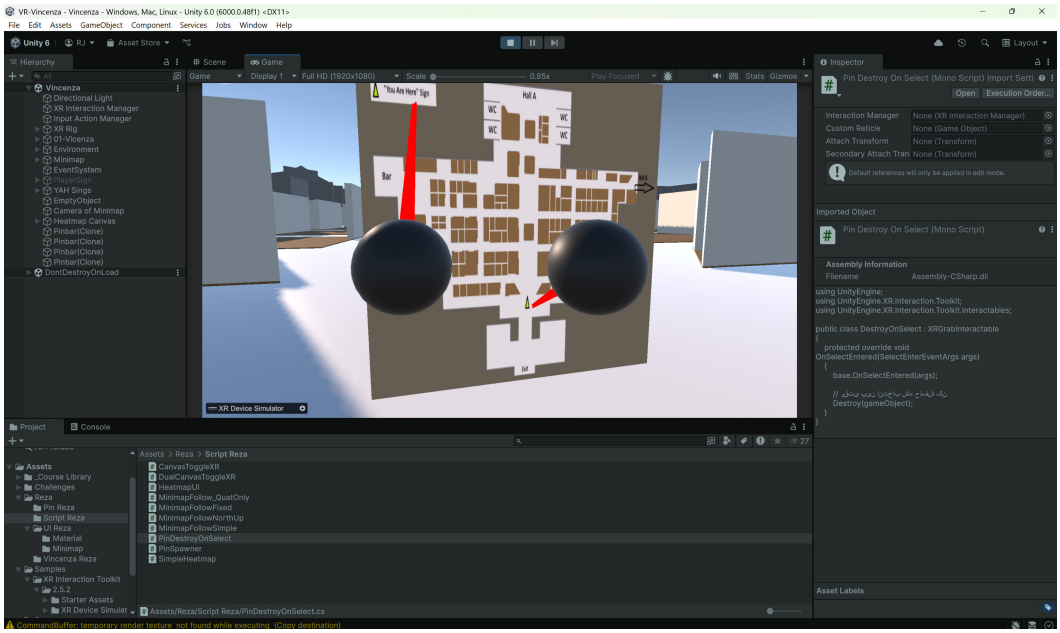

Figure 4.20: First YAH sign.

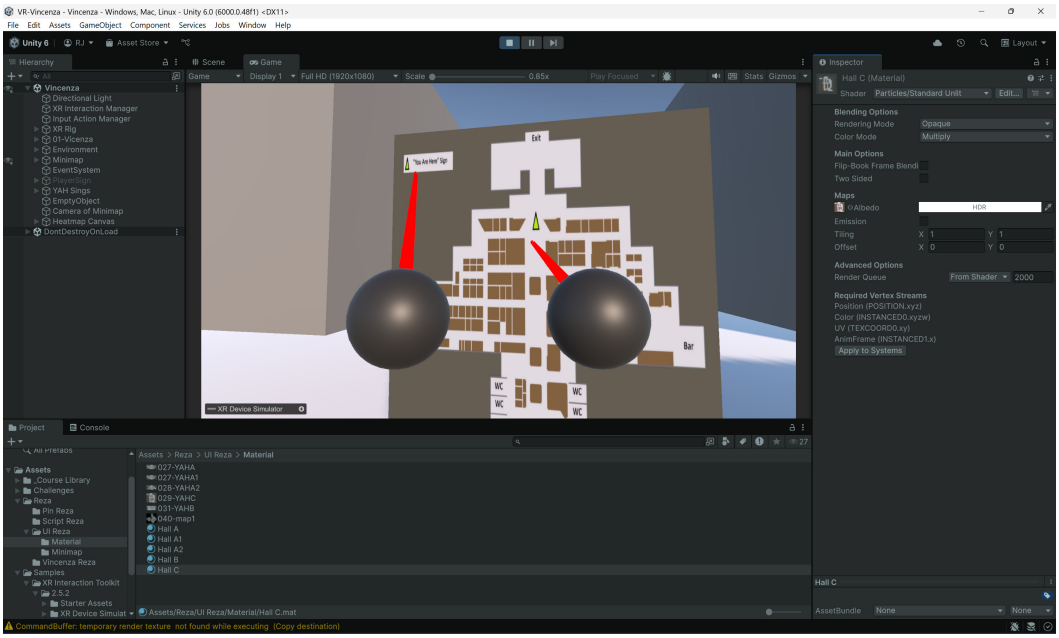

Figure 4.21: YAH sign with the correct forward orientation.

4.2.2 Pin Mechanism

A low-poly pin is modelled in Blender as two simple components, the head and the needle (see Figure 4.22). The model is then exported as FBX and imported into Unity at real-world scale (see Figure 4.22). In Unity, the pin is turned into a prefab with a Box Collider roughly matching its shape, so that the controller rays can detect it reliably, as illustrated in Figure 4.23. An *Empty GameObject* holds a *Pin Spawner* script that listens for the controller's trigger; when pressed, a new pin is placed at the player's current position on the floor, aligned with the surface normal (see Listing 4.1 and Figure 4.24).

Pin removal follows the same lightweight approach used for placement. A small script on the pin prefab listens for selection events (see Listing 4.2 and Figure 4.25). When the ray from the controller hits the pin's *Box Collider* and the user presses the grip button, the corresponding instance is destroyed. This avoids any permanent clutter in the scene and keeps the interaction consistent with the quick, temporary style of all navigation aids [14, 6, 12].

4.2.3 Mini-map

The mini-map follows common design features in previous studies to meet the required needs (as a brief reminder, see Figure 4.26, and Figure 4.27 for the frequency

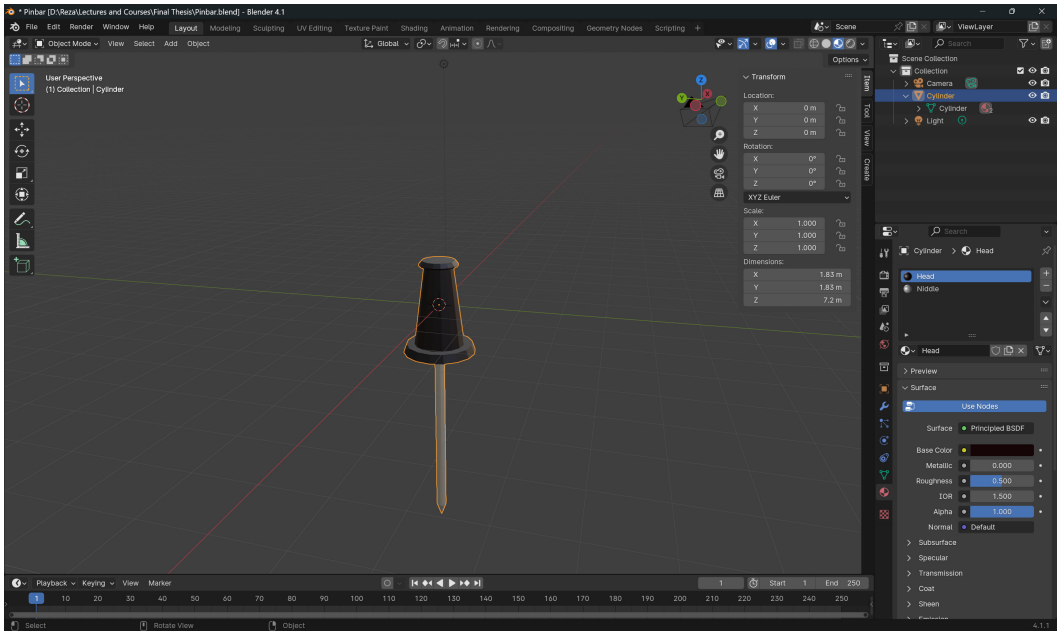

Figure 4.22: A Low-poly pin modelled in Blender.

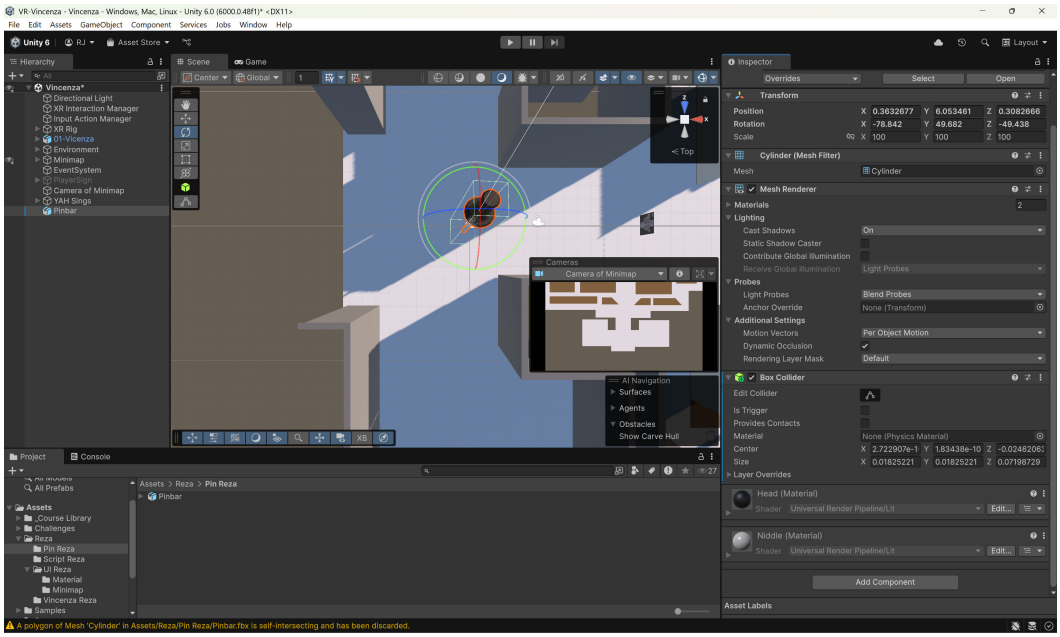

Figure 4.23: The pin model and its box collider.

Listing 4.1: PinSpawner.cs - to place pins at the player's position.

```
using UnityEngine;
using UnityEngine.InputSystem;

public class PinSpawner : MonoBehaviour
{
  public GameObject pinPrefab;
  public Transform player;
  public InputActionReference placePinAction;

  private void OnEnable()
  {
    placePinAction.action.performed += OnPlacePin;
    placePinAction.action.Enable();
  }

  private void OnDisable()
  {
    placePinAction.action.performed -= OnPlacePin;
    placePinAction.action.Disable();
  }

  private void OnPlacePin(InputAction.CallbackContext ctx)
  {
    Vector3 spawnPos = player.position;
    spawnPos.y = 6f;
    Quaternion spawnRot = Quaternion.Euler(-90f, 0f, 0f);

    Instantiate(pinPrefab, spawnPos, spawnRot);
  }
}
```

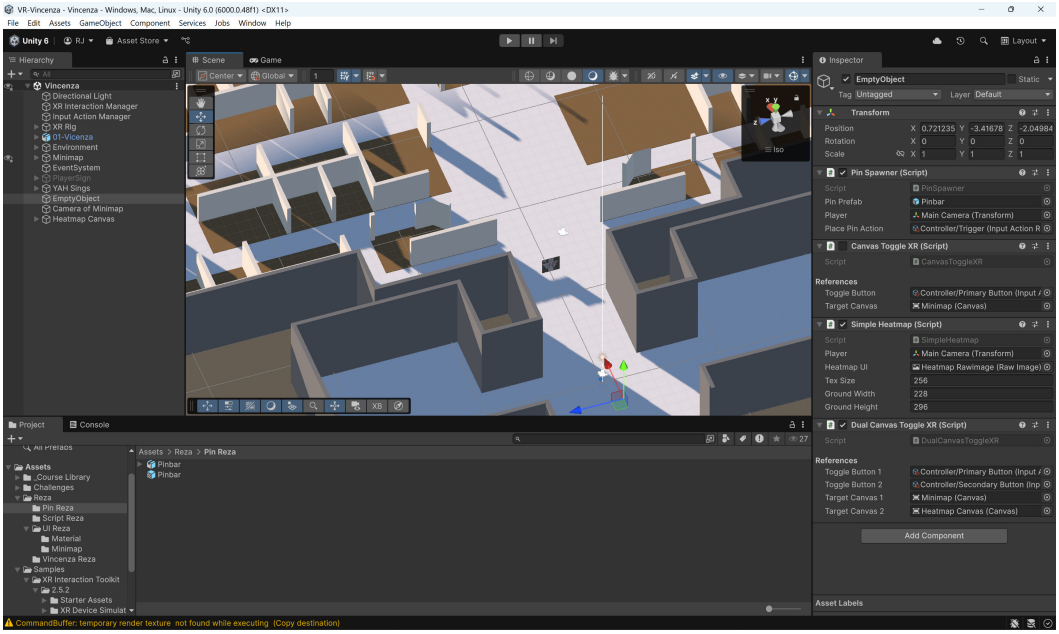

Figure 4.24: On-demand pin spawner.

Listing 4.2: DeleteOnSelect.cs - removes the pin when selected.

```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;
using UnityEngine.XR.Interaction.Toolkit.Interactables;

public class DeleteOnSelect : XRSimpleInteractable
{
  protected override void OnSelectEntered(SelectEnterEventArgs args)
  {
    Destroy(gameObject);
  }
}
```

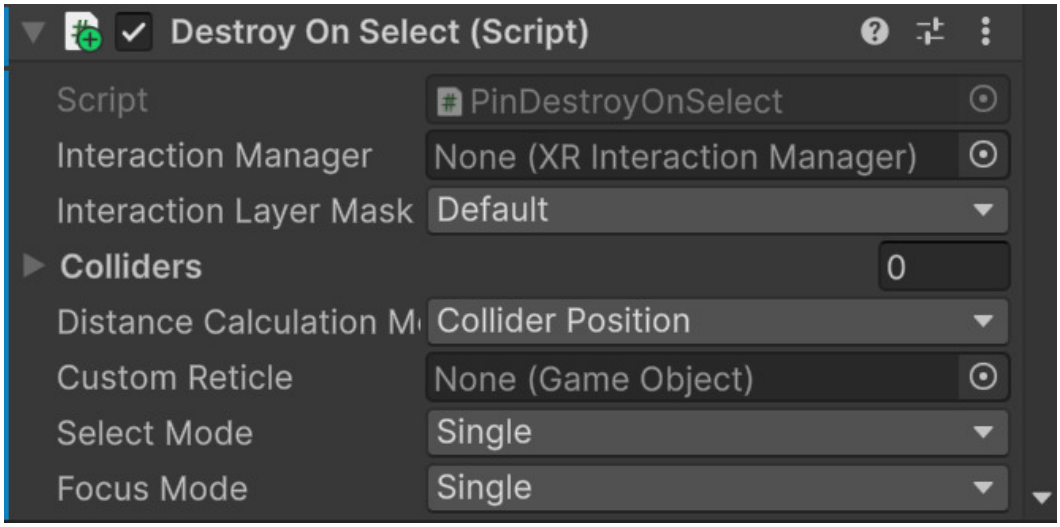

Figure 4.25: Pin removal on select.

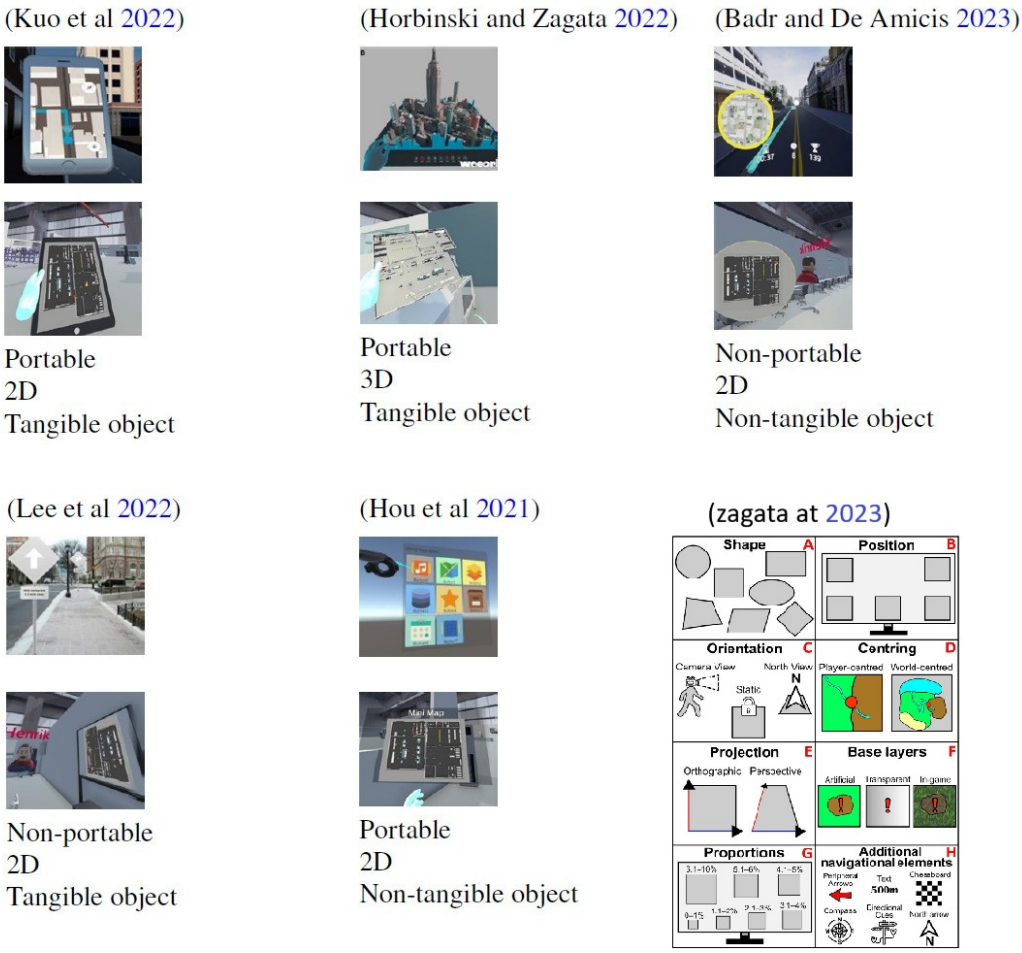

Figure 4.26: Recap of mini-map types and design features. For further information, the reader can refer to the detailed descriptions in Chapter 2 [23].

of design features in video games [23, 1]). First, a screen-space *Canvas* is created and a *Raw Image* is positioned to serve as the minimap panel, as shown in Figures 4.28 and 4.29. The Raw Image is scaled so that the mini-map covers only about three percent of the screen, which matches recommendations for clear but unobtrusive design [23, 1].

Next, a top-down orthographic *Camera* is positioned above the hall and is set to render only the environment layer; shadows, post-processing, and audio are disabled to keep the cost low (see Figure 4.30). The camera's *Output Texture* is directed to a *Render Texture* named `Minimap`, which is in turn displayed on the *Raw Image*.

To make the map appear circular, the *Raw Image* sits under a parent *Image* that uses a circular *Knob* sprite. In fact, the mask shows the child only where the parent image is opaque and hides it where the parent is transparent. A second Image with

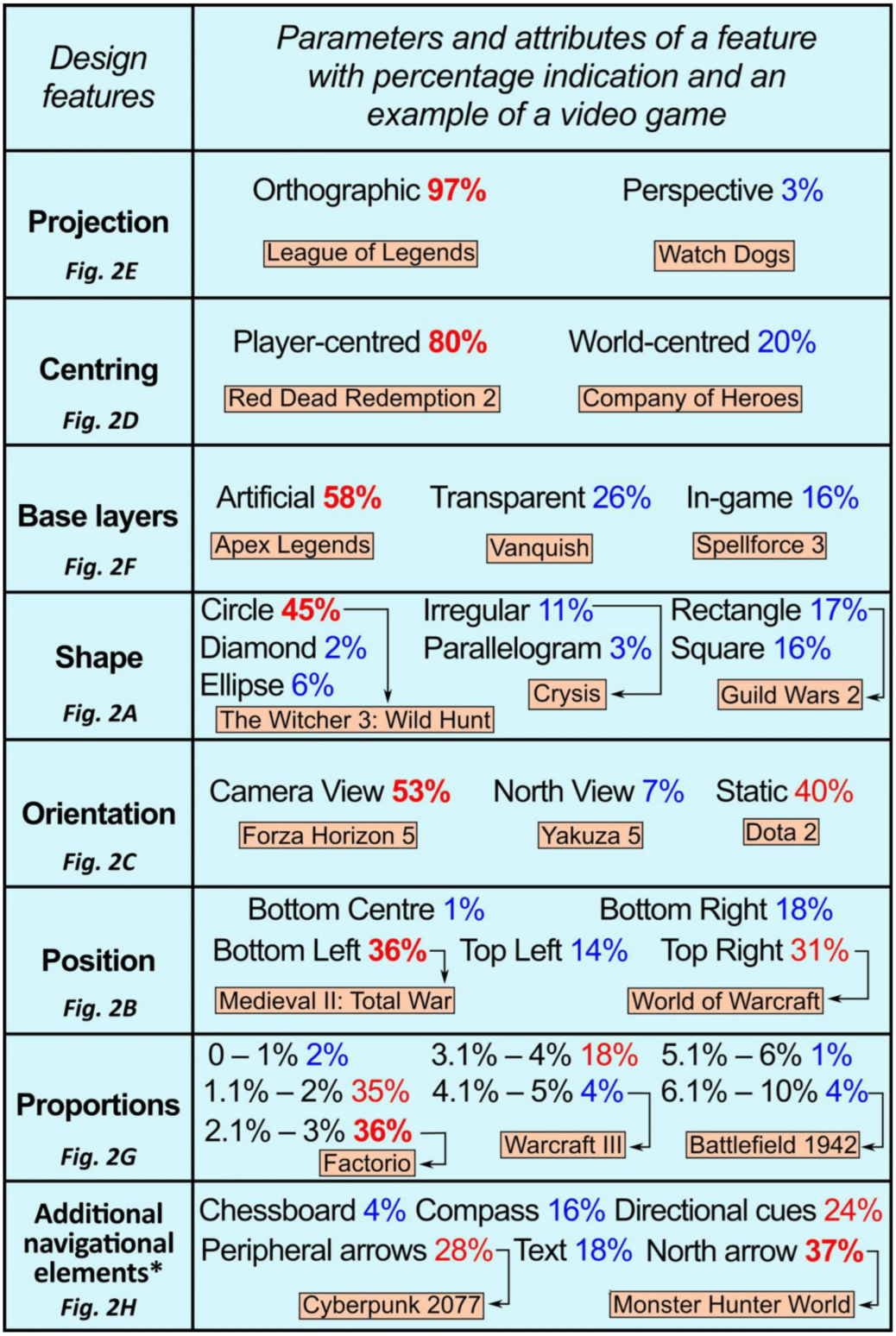

Figure 4.27: Summary of frequent mini-map design choices in games [23].

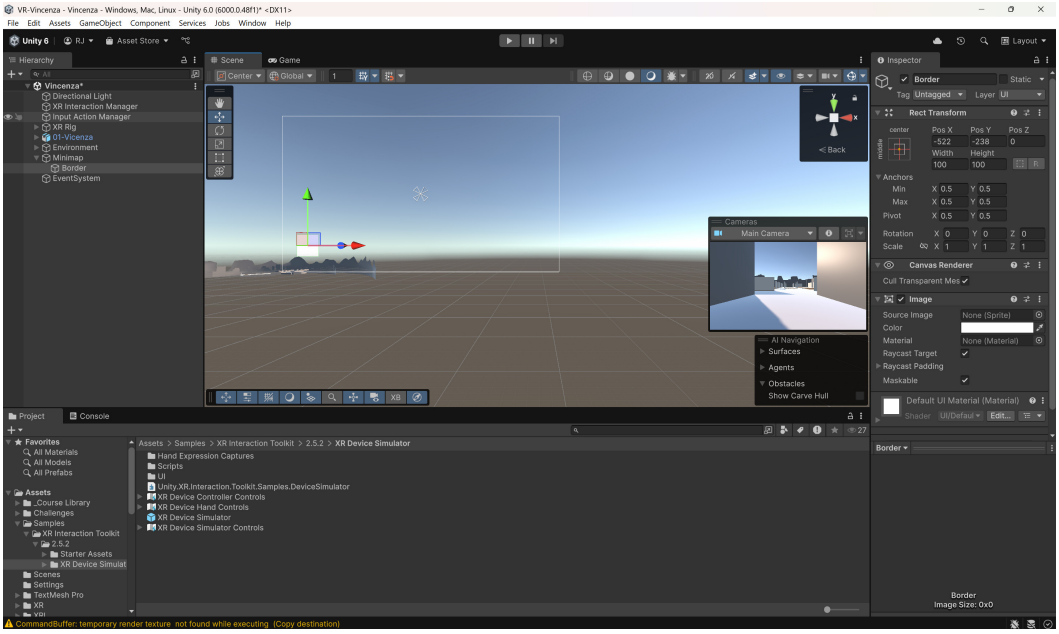

Figure 4.28: Minimap UI canvas.

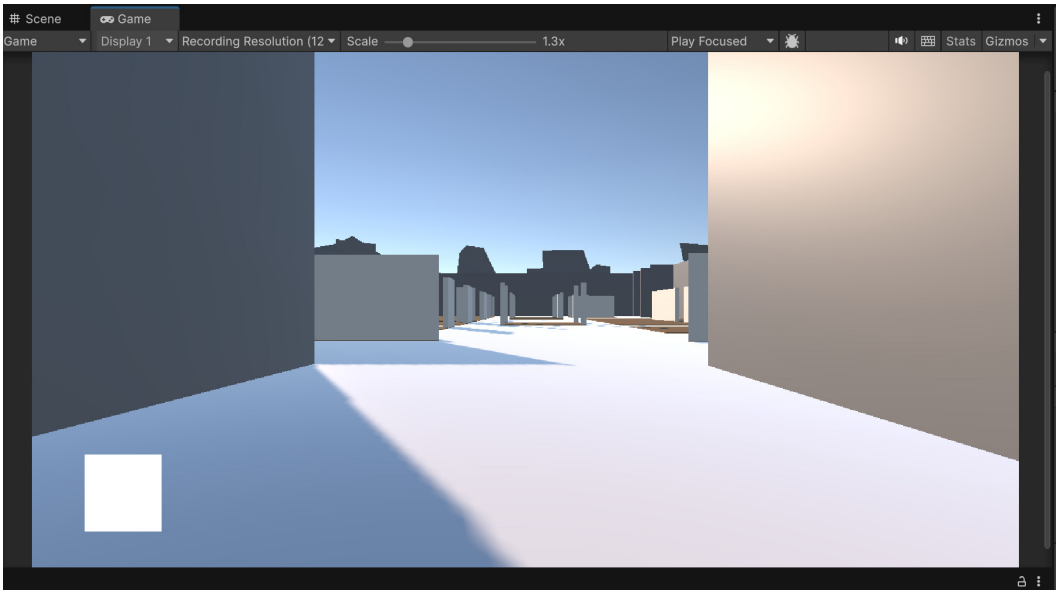

Figure 4.29: Raw minimap panel in the Game view.

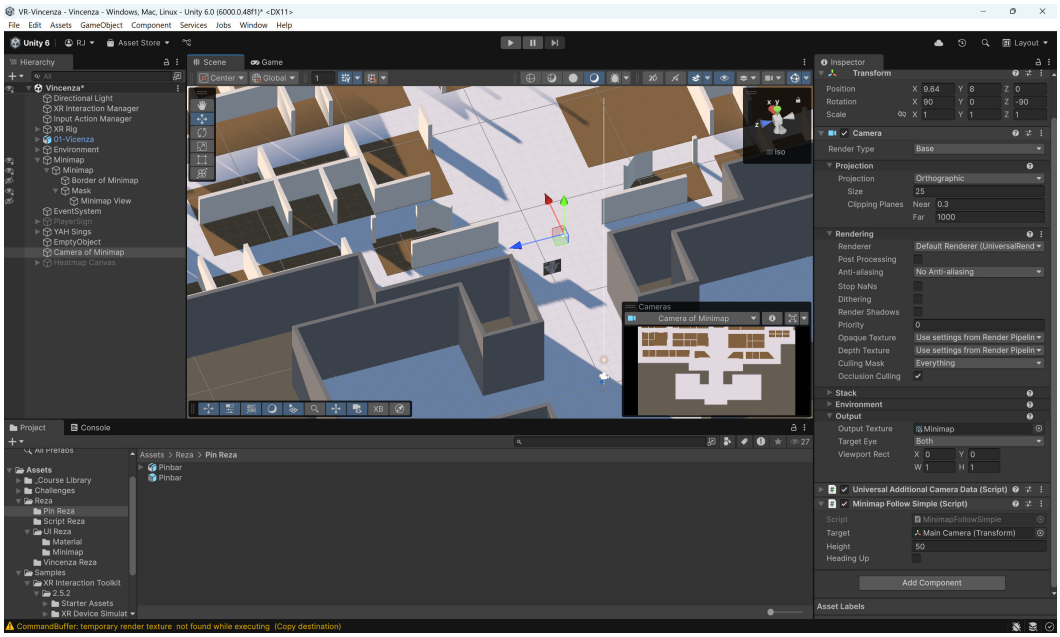

Figure 4.30: Top-down orthographic camera for the minimap.

the same Knob sprite draws a thin ring as the border (see Figure 4.31). Finally, a follow script `Minimap Follow Simple` is attached to the mini-map camera so that the player's X–Z position is copied to the camera while its rotation is kept fixed (see Listing 4.3). In this way the map stays north-up, which helps quick orientation and route planning during wayfinding in VR [12, 6, 14].

4.2.4 Heatmap

A simple heatmap is implemented to highlight where the player has already walked. The process begins with reading the building dimensions from Blender so that they can be translated into pixel units (see Figure 4.32). A dedicated UI *Canvas* contains a *Raw Image* that serves as the display surface for the heatmap texture (see Figure 4.33).

Next, the lightweight script `SimpleHeatmap` keeps a `Texture2D` in memory and updates it frame by frame. In each update, the script converts the player's current X–Z position into pixel coordinates based on the ground dimensions and texture resolution, then draws a soft dot on the corresponding pixel. This logic, presented in **??**, gradually builds a visual trace of movement across the floor (see Figure 4.34). The script exposes a set of parameters in the Inspector, including *Player*, *Heatmap UI*, *Tex Size*, *Ground Width*, and *Ground Height*, which allow flexible adjustment to different layouts, as shown in Figure 4.35.

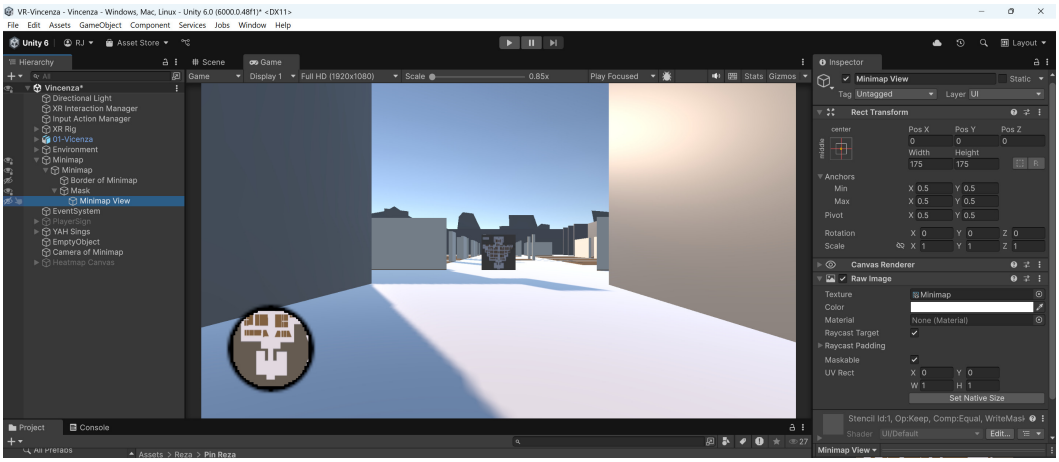

Figure 4.31: Circular minimap with mask and border.

Listing 4.3: MinimapFollowSimple.cs - simple follow script for the minimap camera.

```
using UnityEngine;

public class MinimapFollowSimple : MonoBehaviour
{
  public Transform target;
  public float height = 50f;
  public bool headingUp = false;

  void LateUpdate()
  {
    if (!target) return;

    transform.position = new Vector3(
     target.position.x,
     height,
     target.position.z);

    if (headingUp)
    transform.rotation = Quaternion.Euler(
    90.0f,
    -target.eulerAngles.z ,
    - target.eulerAngles.y);

    else
    transform.rotation = Quaternion.Euler(
    90f,
    0f,
    -90.0f);
  }
}
```

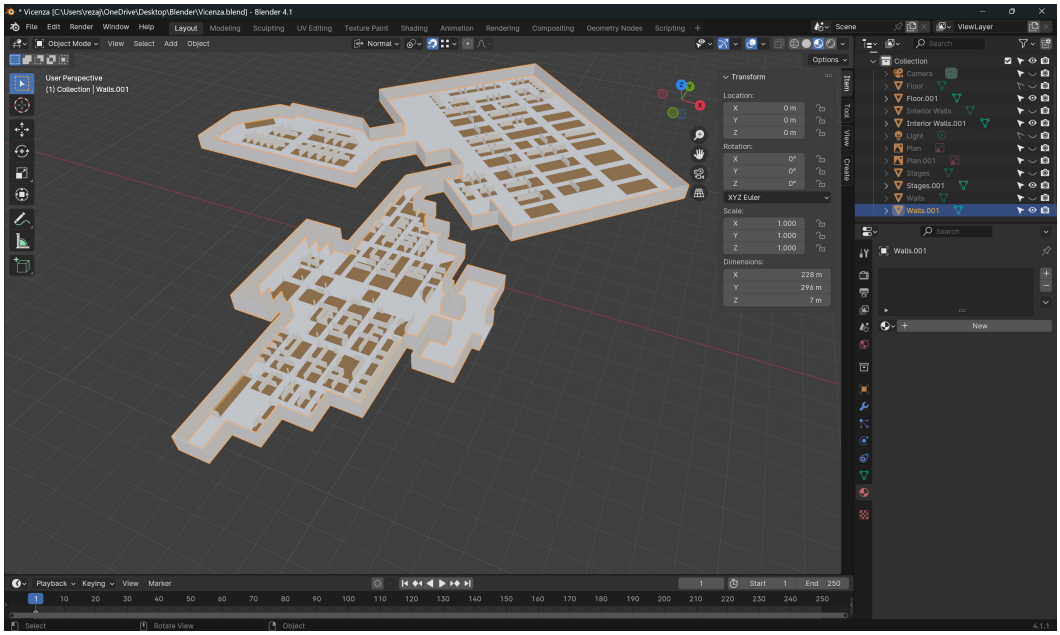

Figure 4.32: Exhibition hall dimension.

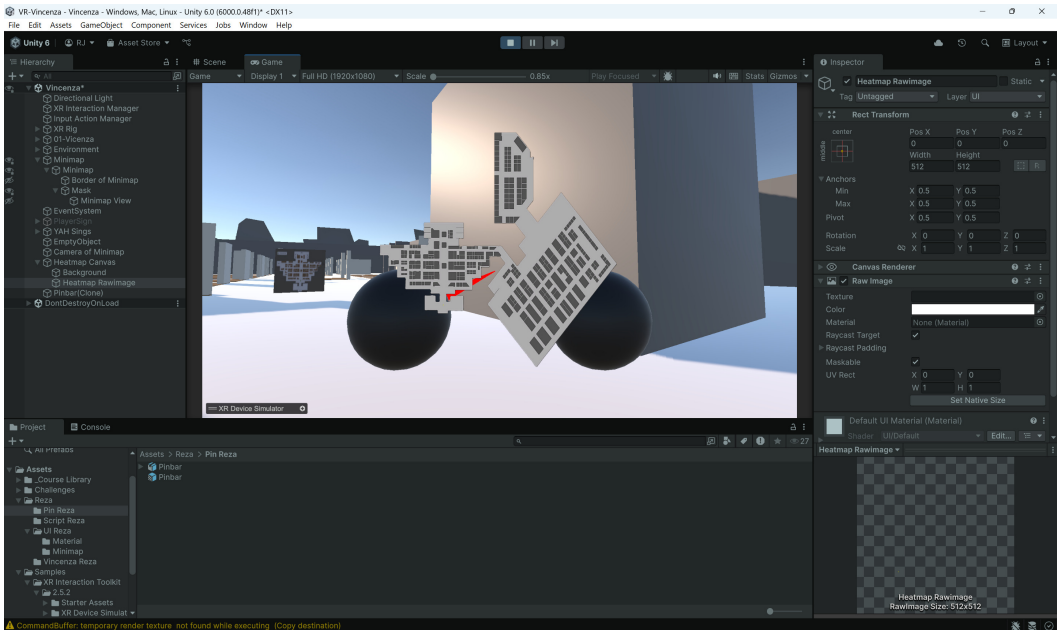

Figure 4.33: Setting up the heatmap panel as a UI.

Listing 4.4: SimpleHeatmap.cs - lightweight heatmap implementation.

```
using UnityEngine;
using UnityEngine.UI;

public class SimpleHeatmap : MonoBehaviour
{
  public Transform player;
  public RawImage heatmapUI;
  public int texSize = 512;

  public float groundWidth = 228f;
  public float groundHeight = 296f;

  private Texture2D heatmapTex;
  private Color[] pixels;

  void Start()
  {
    heatmapTex = new Texture2D(
    texSize, texSize, TextureFormat.RGBA32, false
    );
    pixels = new Color[texSize * texSize];

    for (int i = 0; i < pixels.Length; i++)
    pixels[i] = Color.clear;

    heatmapTex.SetPixels(pixels);
    heatmapTex.Apply();

    heatmapUI.texture = heatmapTex;
  }

  void Update()
  {
    int x = -Mathf.FloorToInt(
    (player.position.z / groundHeight - 0.26f) * texSize
    );
    int y = Mathf.FloorToInt(
    (player.position.x / groundWidth + 0.27f) * texSize
    );

    if (x >= 0 && x < texSize && y >= 0 && y < texSize)
    {
      Color c = heatmapTex.GetPixel(x, y);
      c += new Color(0.05f, 0.02f, 0f, 0.5f);
      heatmapTex.SetPixel(x, y, c);
      heatmapTex.Apply();
    }
  }
}
```

Figure 4.34: Player path and heatmap.

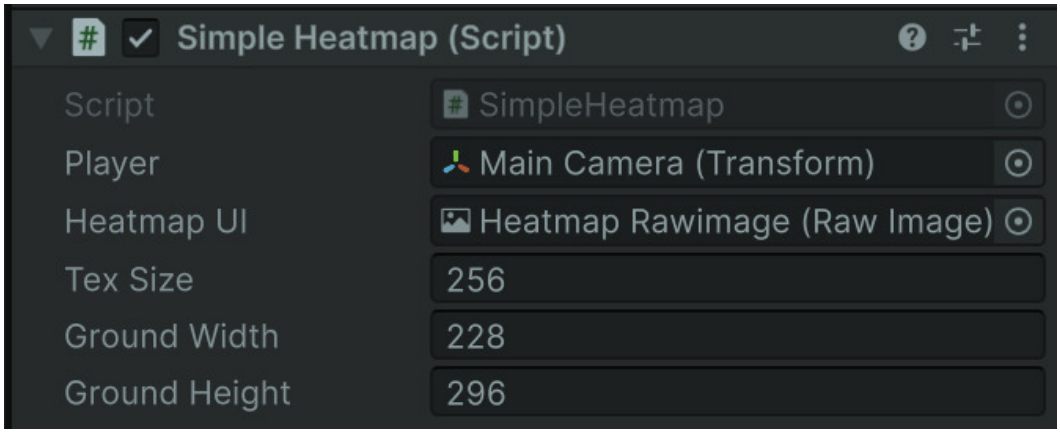

Figure 4.35: Parameters of script for heatmap in inspector.

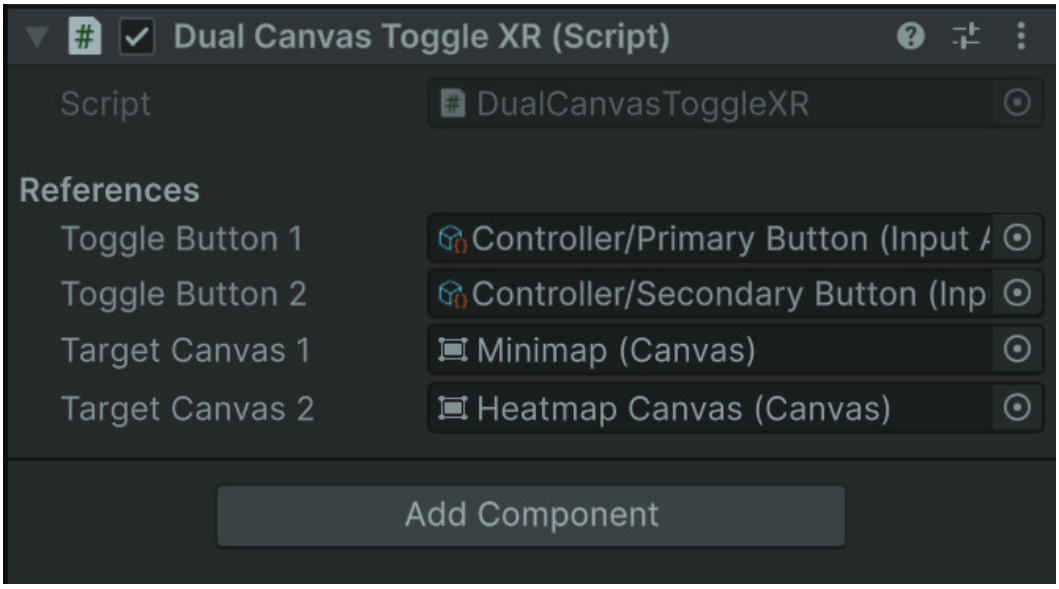

Figure 4.36: On-demand toggle for minimap and heatmap.

As the mini-map and heatmap live on separate canvases, another script named `DualCanvasToggleXR` uses two controller buttons to switch their visibility. This allows the user to summon or dismiss either view on demand, as illustrated in Listing 4.5 and Figure 4.36. Keeping these orientation aids optional, lightweight, and easily toggled aligns with the principles identified in the reviewed literature for unobtrusive navigation support in VR [6, 14].

Listing 4.5: DualCanvasToggleXR.cs - toggling minimap and heatmap canvases.

```
using UnityEngine;
using UnityEngine.InputSystem;

public class DualCanvasToggleXR : MonoBehaviour
{
  [Header("References")]
  public InputActionReference toggleButton1;
  public InputActionReference toggleButton2;
  public Canvas targetCanvas1;
  public Canvas targetCanvas2;

  void Start()
  {
    if (targetCanvas1) targetCanvas1.enabled = false;
    if (targetCanvas2) targetCanvas2.enabled = false;
  }

  void OnEnable()
  {
    if (toggleButton1 != null)
    {
      toggleButton1.action.performed += OnToggle1;
      toggleButton1.action.Enable();
    }

    if (toggleButton2 != null)
    {
      toggleButton2.action.performed += OnToggle2;
      toggleButton2.action.Enable();
    }
  }

  void OnDisable()
  {
    if (toggleButton1 != null)
    {
      toggleButton1.action.performed -= OnToggle1;
      toggleButton1.action.Disable();
    }

    if (toggleButton2 != null)
    {
      toggleButton2.action.performed -= OnToggle2;
      toggleButton2.action.Disable();
    }
  }

  void OnToggle1(InputAction.CallbackContext ctx)
  {
    if (targetCanvas1)
    targetCanvas1.enabled = !targetCanvas1.enabled;
  }

  void OnToggle2(InputAction.CallbackContext ctx)
  {
    if (targetCanvas2)
    targetCanvas2.enabled = !targetCanvas2.enabled;
  }
}
```

Chapter 5

Results

This chapter reports what actually happened on the standalone VR headset in an objective way. It reports observations and measurements, and it lists issues alongside the changes that resolved them.

After each edit to the project, I ran the scene in Unity Play Mode on the laptop as a quick test. The main goal of this step was to confirm that the scene launched, the input mappings worked, and the edit had taken effect. I did not use desktop runs to argue about performance or comfort.

5.1 Visual Results and Placement

Desktop behavior is not a reliable proxy for on-device experience. A laptop typically has a stronger graphics processing unit (GPU) and a different display pipeline, while visual comfort in VR depends on factors that only exist on the headset, such as HMD optics, stereo rendering, head-tracking dynamics, motion-to-photon latency, and controller ergonomics. Content that appears acceptable on a monitor can still produce blur and eye strain.

Therefore, the headset is the final judge. All findings, screenshots, timings, and conclusions in this chapter come from the standalone device. Where relevant, each item states the on-device symptom, the likely cause, the fix that was applied, and the outcome verified on the headset. This section presents a visual review of the scene on the headset and records the observations made during use.

5.1.1 YAH Signs at Corridors

I tested YAH signs at the start of long corridors. At close range they were easy to read, but beyond one to two meters clarity dropped noticeably (see Figure 5.1 and

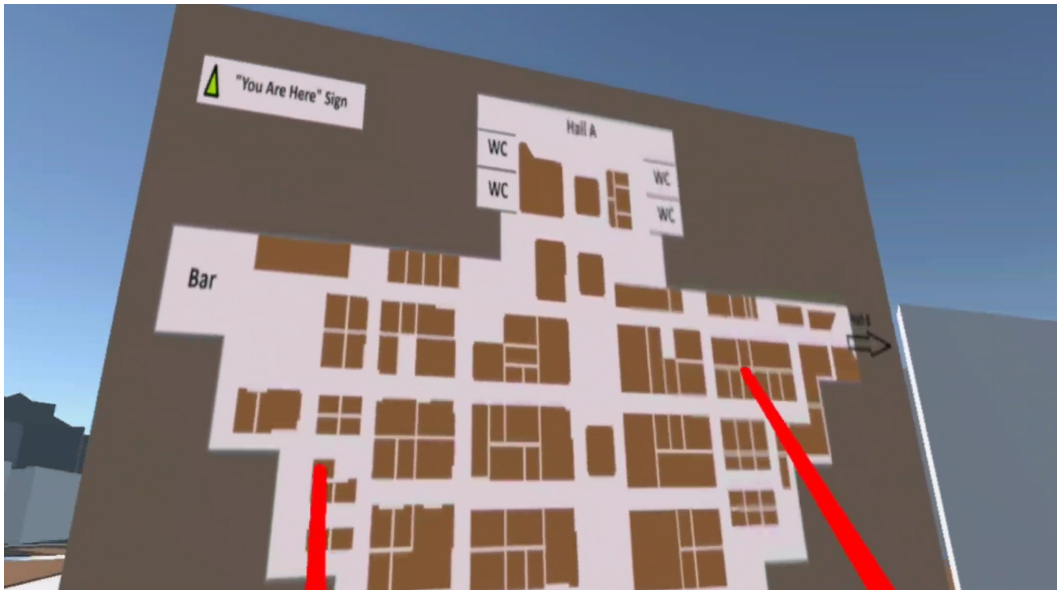

Figure 5.1: The YAH sign viewed from a close distance, where the text and icon are clear and easy to read.

Figure 5.2). I tried several rendering models and adjusted their settings to improve legibility over longer distances, but the difference was minor. The best option I found was to render the signs as unlit text, which kept them clearer across a wider range. Overall, these signs worked well for a quick position check, but they were not useful at long distances.

5.1.2 Pin Interaction and Collider Adjustment

In VR, unlike the desktop version, if user movement is controlled by the joystick, the character needs to be assigned a collider that represents the physical volume it occupies in the virtual scene (see Figure 5.3). A problem appeared when placing a pin, because by script the pin was instantiated exactly at the user's current position. As a result, the pin collider intersected with the player collider, and the physics engine reacted by pushing the user upward in an unnatural way. To solve this, I reduced the size of the player collider and restricted the pin collider to only the upper half of the object, so that the two no longer overlapped and the player body was not displaced (see Figure 5.4 and Figure 5.5). With this adjustment, the interactable surface corresponds to the box collider of the pin, as shown in Figure 5.6.

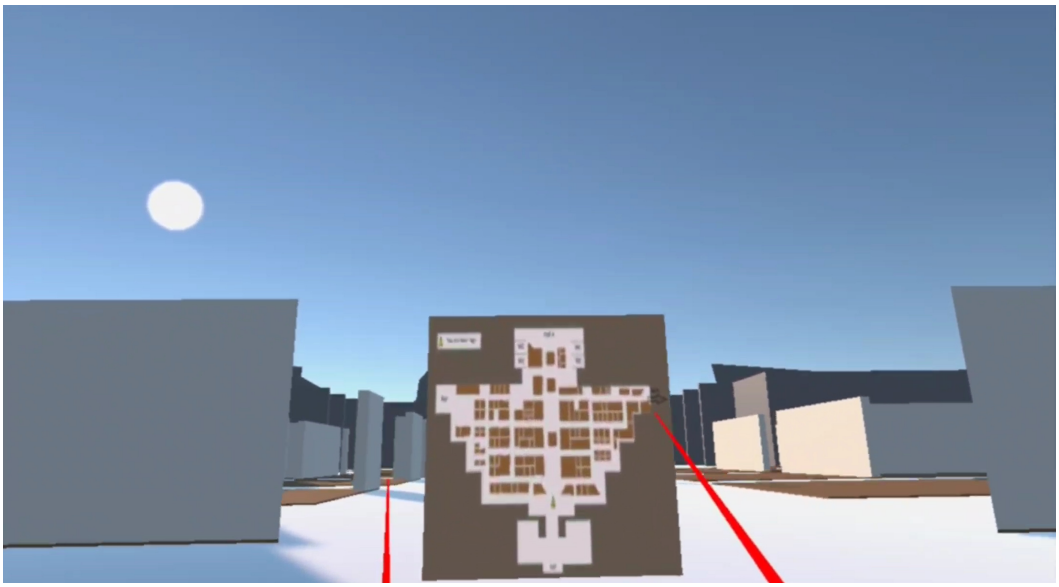

Figure 5.2: The YAH sign viewed from a longer distance, where the text and icon lose clarity and become difficult to read.

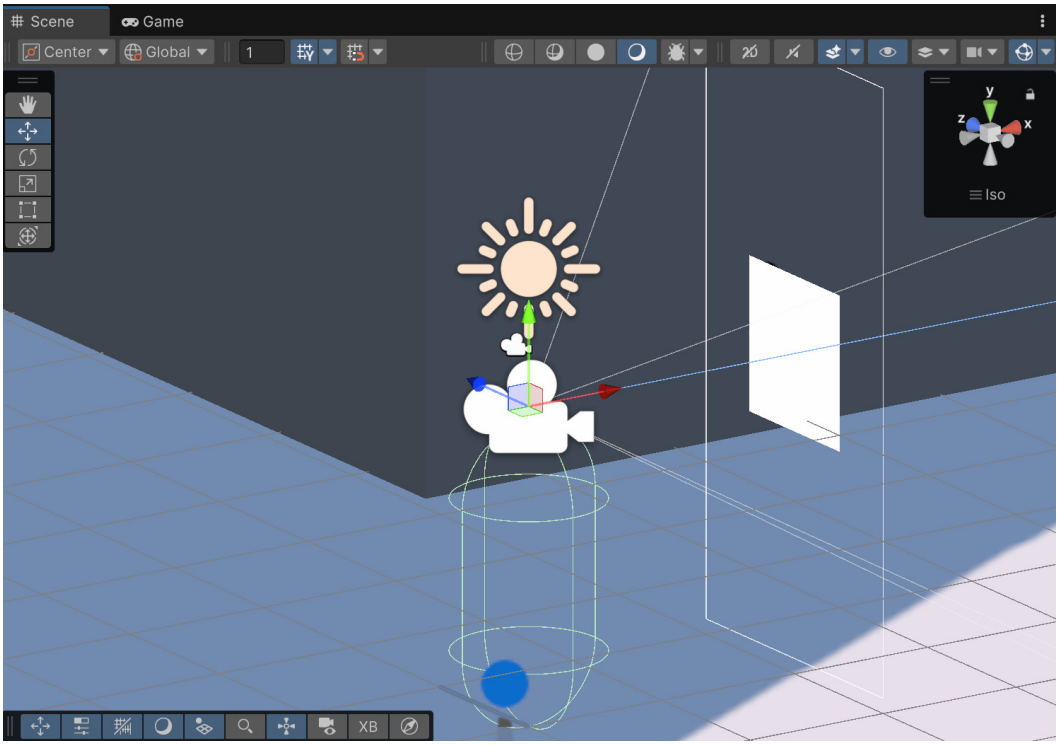

Figure 5.3: The initial collider size assigned to the user, shown in Unity as a capsule surrounding the XR Rig.

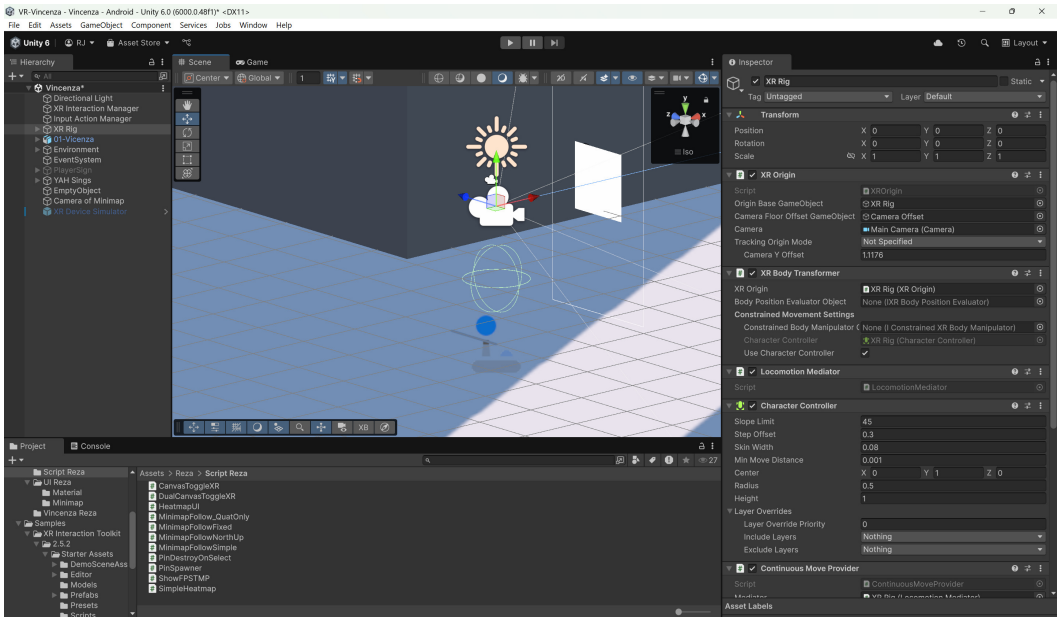

Figure 5.4: Adjustment of the user's collider size in Unity to reduce unwanted interactions with pins and the environment.

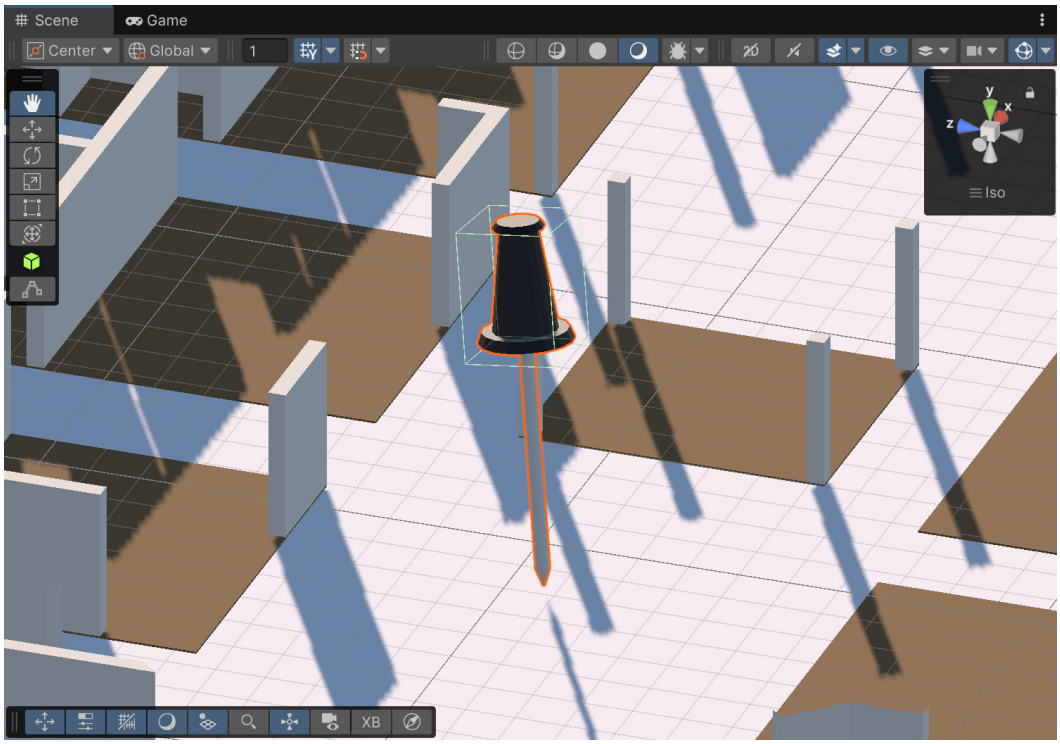

Figure 5.5: Adjustment of the pin collider size in Unity, limited to the upper half of the object to prevent conflicts with the player collider.

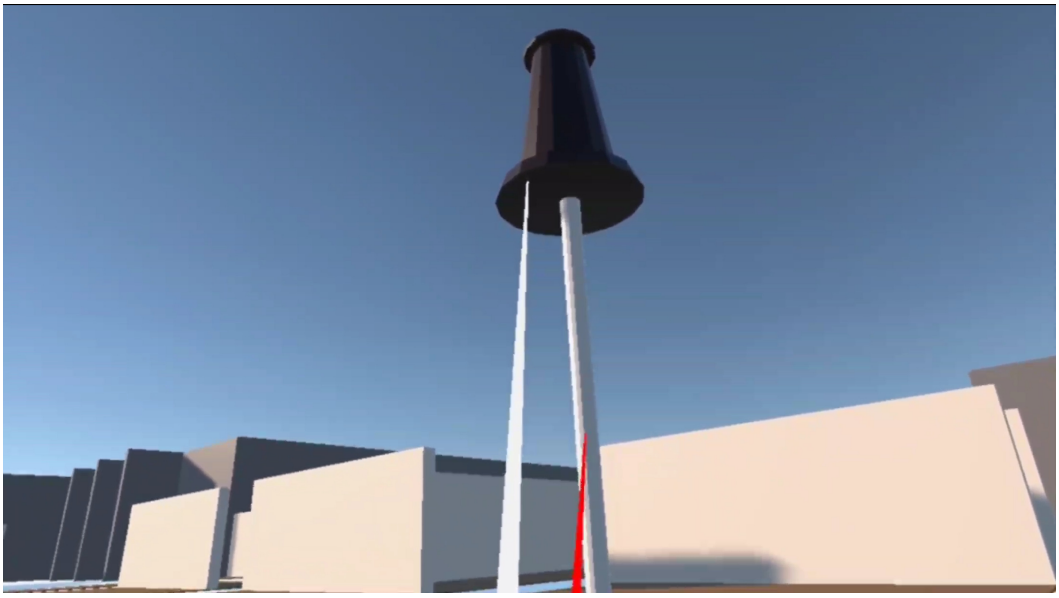

Figure 5.6: Interaction rays showing that only the upper part of the pin is interactable, while the lower part is not.

5.1.3 Corner HUD on a Monitor vs in Headset

A permanent HUD placed in the corner felt acceptable on a monitor, but inside the headset it soon proved annoying and made interaction difficult. In VR, the eyes naturally return to the center of view, which acts as the primary point of concentration. Looking at a corner without moving the head requires the eyes to rotate to the limits of their range. This shifts the gaze outside the lenses' sweet spot, which increases eye strain and often produces blur or ghosting that resembles double vision.

To address this issue, the design was modified so that panels appear on demand near the center of view and then fade out automatically after a few seconds, following the recommendations in [14]. However, after testing, I refined the approach. Instead of panels that appear for a few seconds and then fade out, I implemented a design allowing users to press and hold a button to summon the panel only when needed. This solution resembles checking the rear-view mirror while driving, where attention is intentionally shifted for a moment and then returned to the main focus. In this way, the panel functions as a purposeful tool rather than a constant source of annoyance.

In this design the panel is anchored at a fixed distance in front of the user, so it moves together with the user while staying stable in their view (see Figure 5.7). The drawback of this approach is the possibility of occlusion (see Figure 5.8). Whenever

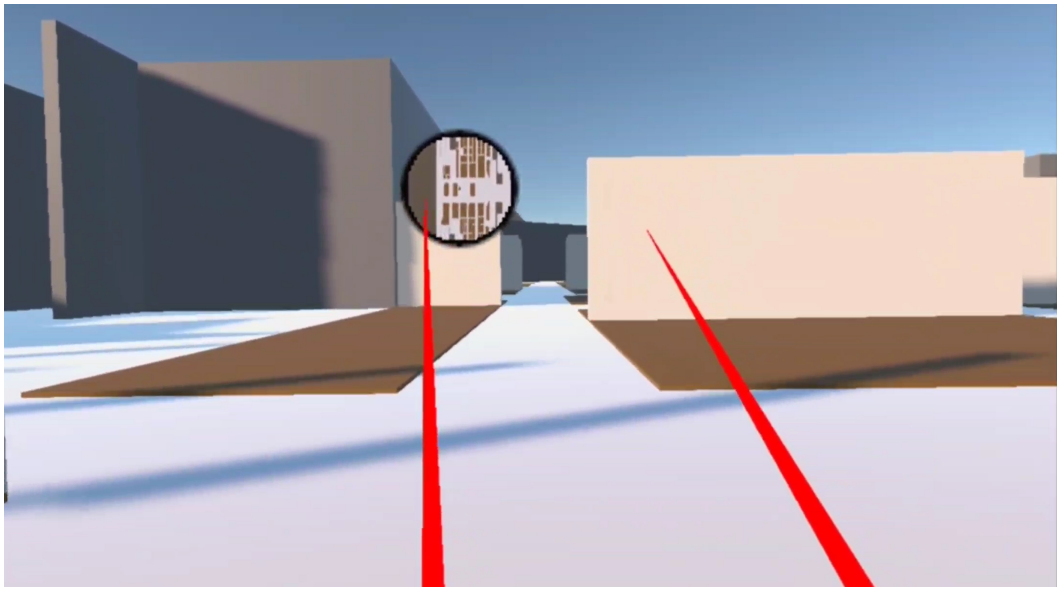

Figure 5.7: The minimap anchored to the user's view, staying fixed at a set distance in front of the player.

another object comes between the user and the panel, the panel may be partially or completely hidden from view. To reduce this risk, I shortened the distance between the user and the panel, which minimized the chance of occlusion while keeping the content readable.

5.1.4 Heatmap on a 2D Canvas

When the heatmap was first drawn on a 2D screen-space canvas, the result looked correct on the laptop monitor but not inside the headset. Each lens produces a slightly different image for the two eyes, and because I had only checked the flat 2D view, I mistakenly assumed it was working. In practice this led to a double-vision effect that made the heatmap unreadable. The problem was resolved by adopting the same approach for mini-map (see Figure 5.9). Therefore, Like mini-map, this panel remains vulnerable to occlusion when objects block the line of sight (see Figure 5.10).

5.2 Scope and Test Settings

At this stage, the goal is to support the claims with numerical data collected directly from the headset. The main focus is on assessing both the functionality and the performance of the scene while walking and using the different navigation aids inside

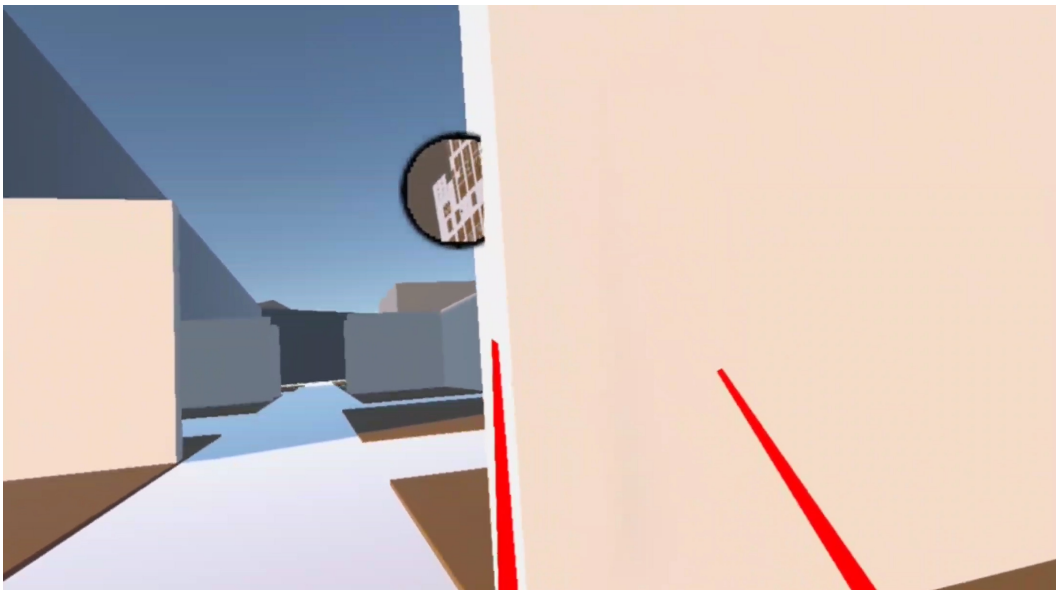

Figure 5.8: An example of minimap occlusion: when another object comes between the user and the anchored minimap, part of the visualization is blocked from view.

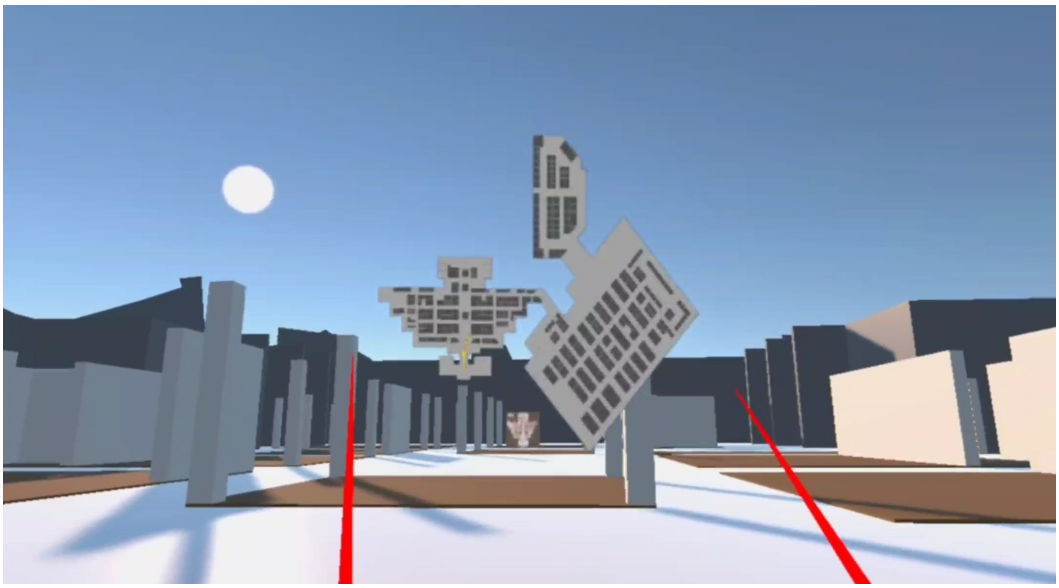

Figure 5.9: The heatmap implemented using the same world-space panel approach as the minimap, anchored in front of the user at a fixed distance.

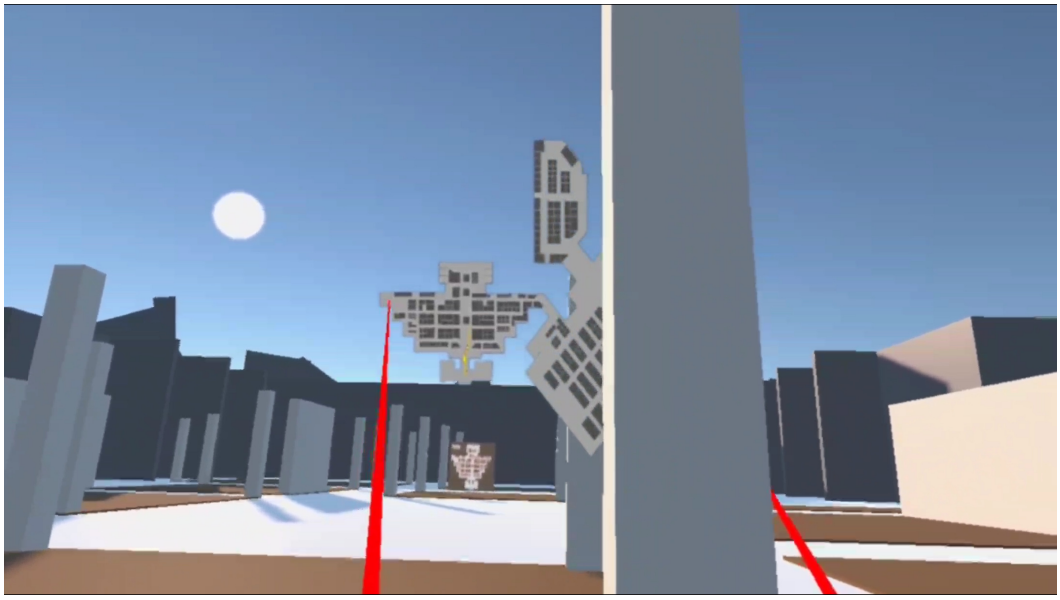

Figure 5.10: An example of heatmap occlusion: when another object comes between the user and the heatmap panel, the visualization is partially blocked from view.

the device.

5.2.1 Performance Evaluation Methodology

To evaluate performance, I first wrote a small Unity script that displayed the frame rate inside the headset (see Listing 5.1). The purpose of this overlay was to reveal any frame-rate drops during navigation, making it possible to identify and correct areas where rendering became demanding. The script computed the instantaneous frame rate as well as the session's minimum and maximum values, and displayed them on a simple `Canvas` with a `Text` element fixed to the view, so the values remained visible throughout navigation.

After setting up the overlay, I started the navigation tests on the standalone headset. Each run lasted about two to three minutes and covered a random route along the corridors. During testing, I changed the starting points and followed completely random routes through the corridors. Throughout navigation I used the different aids in varied ways, sometimes activating several of them at the same time. As the development process itself already required running the scene many additional times, these ten structured runs were considered sufficient for validation.

Listing 5.1: ShowFPSTMP.cs - Frame-rate overlay used in tests

```
using UnityEngine;
using TMPro;

public class ShowFPSTMP : MonoBehaviour
{
        public TMP_Text fpsText;
        float smooth;

        int frames, lastFPS, minFPS = int.MaxValue, maxFPS = 0;
        float t;

        void Reset() { fpsText = GetComponent<TMP_Text>(); }

        void Update()
        {
         smooth += (Time.unscaledDeltaTime - smooth) * 0.1f;
         int fpsNow = Mathf.Clamp(Mathf.RoundToInt(1f / smooth), 1, 999);

         frames++;
         t += Time.unscaledDeltaTime;
         if (t >= 1f)
         {
          lastFPS = fpsNow;
          minFPS = Mathf.Min(minFPS, lastFPS);
          maxFPS = Mathf.Max(maxFPS, lastFPS);
          fpsText.text = $"FPS:␣{lastFPS}\nMin:␣{minFPS}␣␣Max:␣{maxFPS}";
          t = 0f;
          frames = 0;
                }
        }
}
```

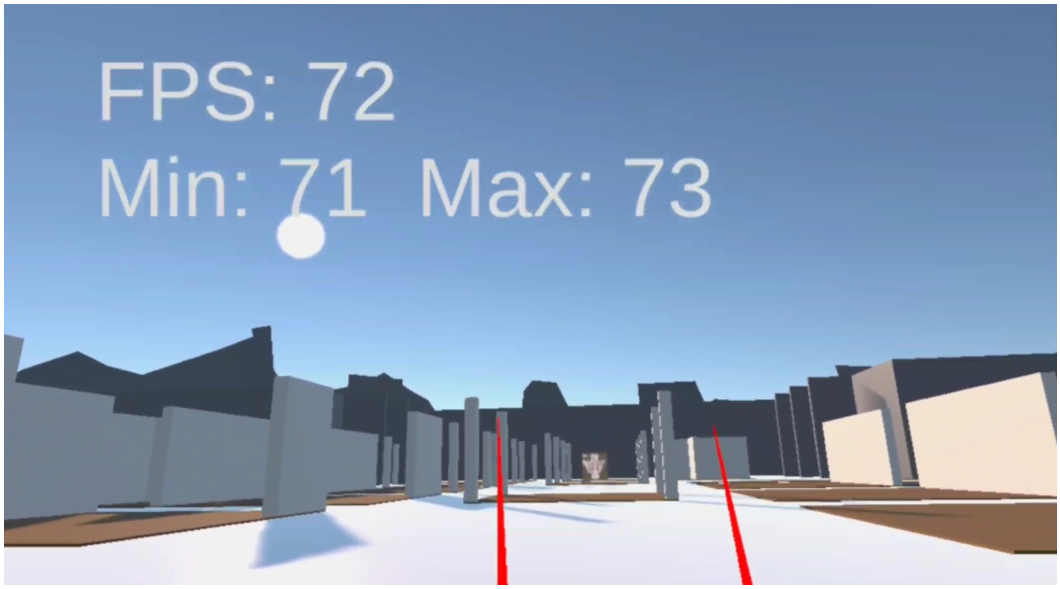

Figure 5.11: Observed frame rate values during testing, showing the current FPS along with minimum and maximum values recorded in the session.

5.2.2 Frame-Rate Results and System Stability

According to Figure 5.11 the results were consistent across runs:

- **Frame cap:** the frame rate was limited to around 72 fps, matching the default 72 Hz refresh rate of the headset in Unity builds.
- **Start dip:** at the beginning of each session the frame rate briefly dropped to around 71 fps while the scene completed its initial loading.
- **Stability:** after the first loading the frame rate remained steady at 72 fps, with no noticeable variation during walking or interaction.

These measurements suggest that the current scene fits comfortably within the render budget of the device. The test acted as a lightweight health check to confirm stable functionality of the navigation aids under real use, not as a full thermal or long-duration profile.

5.3 Summary of Findings

The first outcome is that, despite the challenges encountered, developing this prototype proved feasible. All four navigation aids - the mini-map, the heatmap, pins, and YAH signs - were fully built and tested on the standalone headset. Each tool

draws correctly on screen, responds to inputs, and keeps its state. Tests while walking showed that I can bring each tool into view only when needed, so there is no constant HUD taking space. The issues I saw during development were resolved. The heatmap no longer shows a double image because it now sits on a 3D panel, and pins no longer push the player upward after I adjusted the colliders. I ran ten sessions along the corridors. The headset's default 72 hertz refresh rate capped the frame rate at 72 frames per second; after a small drop at the start, it stayed steady. This means the scene fits the device, and the prototype is ready for further work and for comparing design options in real VR use.

A second key finding is that design expectations often diverge from what actually works in practice. YAH signs helped only at short distance; beyond one or two meters they lost clarity. A corner mini-map that felt fine on a desktop became uncomfortable in the headset because it pulled the gaze away from the center. These results show that ideas that seem reasonable in theory are not necessarily the most usable in VR, so on-device tests should guide the final design choices.

Chapter 6

Conclusions

This final chapter brings together the main threads of the thesis. It begins by restating the core goal that motivated this work, then highlights the key contributions and how they differ from existing platforms. After presenting the limitations and outlining future work, it concludes with a brief reflection on the broader impact of these findings.

6.1 Goals and Motivation

The primary goal of this thesis, inspired by my own experience of feeling lost in a real exhibition, was to support users in finding their way easily and without anxiety about getting disoriented, so that they could focus on and enjoy the visit itself. At the very beginning, the aim was to create a practical system that integrated tools like a mini-map, a visit-trace heatmap, simple pins, and YAH signs directly into a standalone VR headset, ensuring that these aids remained easy to use and did not overwhelm the user.

6.2 Key Contributions

In this section, we outline the main contributions of this thesis in detail. Each contribution represents a concrete outcome of the design, implementation, and evaluation process.

First and foremost, the thesis presents a headset prototype that runs directly on a standalone VR device. The work demonstrates that orientation aids can be implemented and verified under the constraints of an all-in-one headset, where rendering and interaction must perform reliably. By focusing on continuous locomotion rather

than teleportation, the prototype demonstrates that it is possible to provide orientation support that could help user's navigation experience. This contribution is significant because it shows that a standalone VR headset can host a fully functional navigation aid system tailored for exhibition-like environments.

Furthermore, in this thesis, the focus was not only on implementing orientation aids but also on documenting the problems that arose in practice and the simple solutions that proved effective. For example, always-on corner HUDs looked convenient in theory, but during trials they caused eye strain and broke immersion; the fix was to replace them with short, on-demand panels placed at the center of view. Rendering heatmaps on a 2D canvas seemed straightforward, but it created binocular mismatch and double vision; switching to world-space rendering solved this issue. Simple pins worked well for marking points of interest, but their colliders conflicted with the player body; reducing the collider size and enabling single-action removal made them reliable. Finally, YAH signs were added at corridor entries: they proved useful at close range but quickly lost clarity at greater distances. This was not treated as a strong advantage or disadvantage, but rather reported neutrally as part of the overall experience. In this way, the project highlights both what failed and what worked, turning each challenge into a clear design rule or fix that can guide future VR exhibition systems.

6.3 Distinction from Existing Platforms

Platforms such as **Spatial** and **Frame** illustrate this point. Spatial offers virtual galleries and creative hubs where art, media, and social events can be showcased, but its core emphasis remains on content sharing and collaboration. Frame, by contrast, is a lightweight, web-based platform that makes it easy to set up online meetings, classes, or events directly in the browser. Both can be adapted to host exhibition-like sessions, yet neither was originally designed as a dedicated exhibition platform.

The present thesis differs in its focus. Rather than maximizing content sharing or social interaction, the prototype treats *navigation* as a first-class concern. It implements and evaluates a lightweight bundle of aids, explicitly designed to work with continuous joystick locomotion in VR. The result is a set of rules for keeping users oriented without overwhelming them. Therefore, the work could also express the ambition to propose a new platform in the future-one designed to host large-scale exhibition events, support shopping in big supermarkets, and provide orientation in other complex indoor environments.

6.4 Limitations and Threats to Validity

This section summarizes the main limitations encountered during the project and explains how they shaped the outcomes.

6.4.1 No Controlled User Study

A key limitation of this work is that no controlled user study was carried out. The system was tested only through developer-led sessions, which helped to identify technical issues and refine the features but did not provide systematic feedback from independent participants. A controlled study with multiple users, proper task design, and statistical analysis would have been necessary to evaluate usability and effectiveness in a reliable way. The absence of such a study means that the results presented here should be seen as preliminary, and future work will need to include structured experiments with participants to validate the findings

6.4.2 Task and Scene Scope

The scope of tasks and scenes in this project was intentionally kept limited. The main goal was to test the navigation aids in a controlled and manageable setup, rather than to build a large and complex virtual world. Tasks focused on basic actions such as walking through corridors, opening and closing the mini-map or heatmap, placing and removing pins, and checking visibility. Scenes were designed to be simple exhibition-like layouts, large enough to require orientation. This restricted scope allowed for faster iteration and debugging, but it also means that the findings may not directly transfer to more complex or content-heavy environments

6.4.3 Method Mismatch with Prior Work

Most of the prior literature on navigation in VR has assumed *teleportation* as the default locomotion method [20, 6, 12, 22]. In contrast, this project required *continuous locomotion*, where the user moves smoothly with the joystick. Because of this difference, many findings and design recommendations reported under teleportation are not directly reusable for continuous motion; for example, comfort or orientation results measured with teleportation do not necessarily carry over to joystick-based movement [17]. This mismatch limits direct comparability and underlines the need for studies that focus specifically on continuous locomotion.

6.4.4 Single Device

All development and testing were performed on a single standalone VR headset. This limited setup was sufficient to verify that the system worked and that the navigation aids could be activated and used in practice. However, it did not allow for testing with multiple devices at the same time, which would have been necessary for collaborative scenarios or for gathering data from a broader participant pool. Relying on a single device also means that potential differences across hardware models, display qualities, or tracking systems were not explored. As a result, the findings should be considered device-specific and may not generalize to other platforms.

6.5 Future Work

Several directions can extend and deepen the present work:

- **Serious-game wayfinding protocol.**

  A standard serious-game protocol for wayfinding assessment is a structured procedure that uses game-like tasks to evaluate how well users can navigate in a virtual environment. In this approach, the virtual scenario is designed as a serious game, meaning that it has playful elements but the main purpose is research or training rather than entertainment. The protocol usually defines the starting point, the target locations, and the rules for interaction, so that all participants face the same conditions (see Figure 6.1). This makes it possible to compare their performance in a fair and repeatable way. Typical measures include the time needed to reach a goal, the number of wrong turns, and the efficiency of the chosen path. By using a standardized game-like protocol, researchers can collect reliable data about spatial orientation and navigation strategies [25].

- **Remote studies at scale.**

  Large-scale remote studies refer to research activities where data collection and interaction with participants are carried out without requiring them to be in the same physical place. In this approach, participants can join from their own location, often through the internet and devices such as VR headsets or online software platforms. The main goal is to include a larger and more diverse group of users, sometimes even from different parts of the world, in order to obtain a broader and more reliable sample. The main advantage of this method is

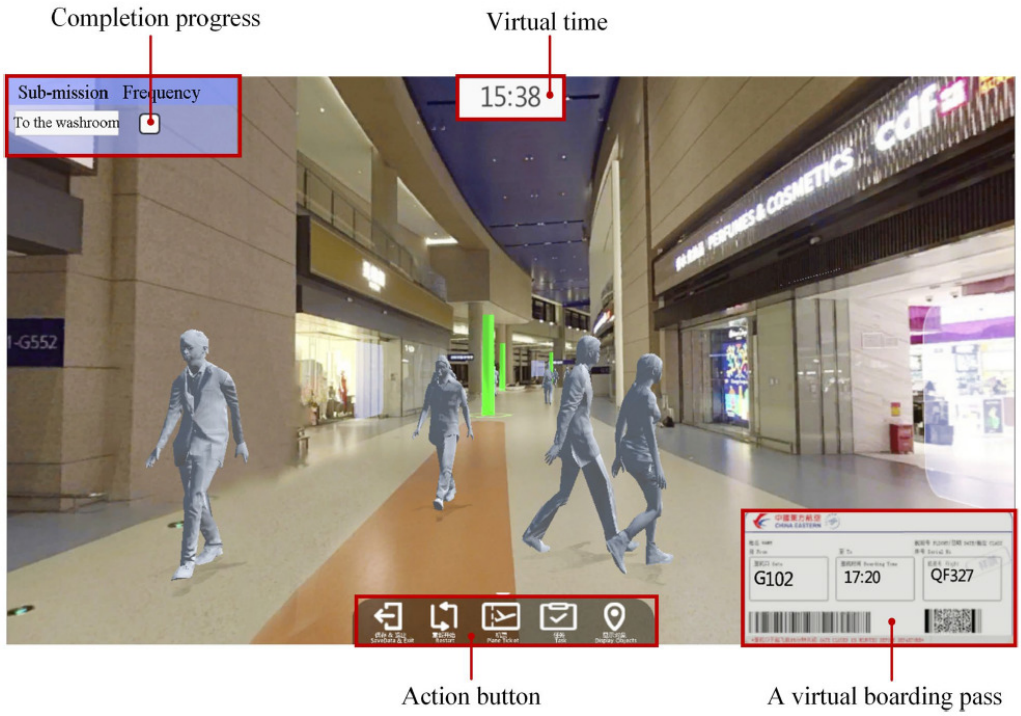

Figure 6.1: Illustration of a serious-game wayfinding protocol used for evaluating airport design, including progress tracking, virtual time, action buttons, and a virtual boarding pass [25].

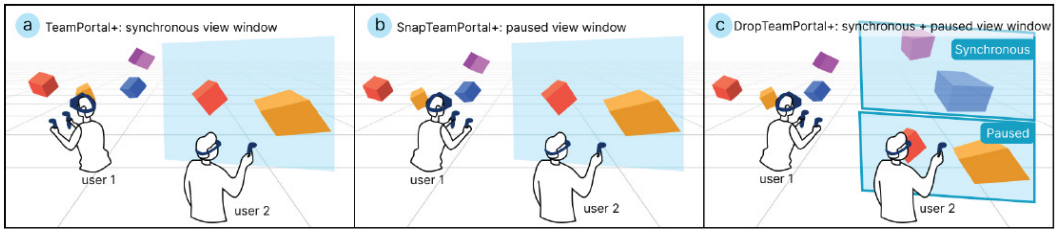

Figure 6.2: An illustration of collaborative navigation using TeamPortal. Different setups are shown: (a) synchronous shared view, (b) paused view, and (c) a combination of synchronous and paused windows. These modes allow multiple users to share or compare perspectives, enabling richer teamwork and decision-making in VR environments [19].

that it reduces infrastructure costs and removes the need for travel or local laboratories. However, the key limitation is that it is harder to control the environment of each participant, and it cannot always be guaranteed that all of them experience the study in exactly the same conditions [2].

- **Collaboration.**

  Multi-user collaboration with shared and parallel views refers to settings where more than one participant takes part in the same virtual environment at the same time. In the shared-view approach, all users see the same scene from a common perspective, which helps them coordinate actions and build a joint understanding of the space. In the parallel-view approach, instead, each user keeps their own individual viewpoint but can still interact with others in real time. The goal of both methods is to study how people work together, share information, and solve navigation tasks when they are not alone in the environment. Shared views are useful for achieving a common focus, while parallel views give participants more freedom and reflect real-world collaboration where everyone has a different angle. Such protocols can provide valuable insights into teamwork, communication, and joint decision-making in [19].

- **Extending pins to lightweight notes.**

  Extending pins to lightweight notes means that the simple markers used in the virtual environment are enriched with short pieces of information. Instead of being only a visual flag that shows a position, each pin can also carry a short text or symbol, like a label or reminder. This makes the interaction more

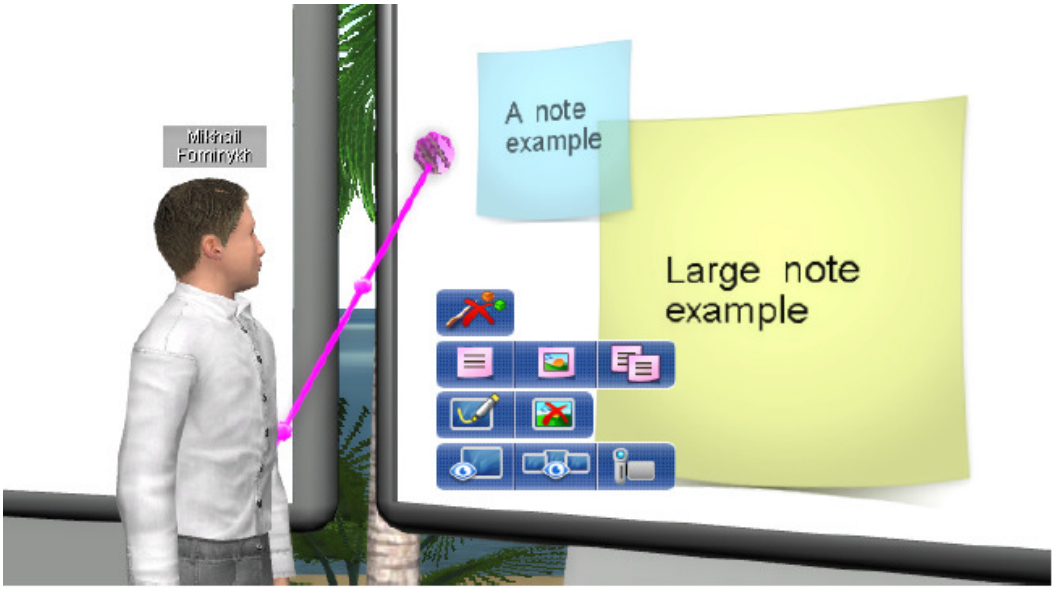

Figure 6.3: Example of a sticky-notes method in a virtual environment, where users can attach short comments or annotations alongside pinned locations. This can extend simple pin markers with lightweight textual feedback [16].

flexible, since users can leave quick annotations directly in the virtual space, similar to sticky notes in the real world. The main goal is to support memory, communication, and organization without making the interface heavy or distracting. Such lightweight notes can be useful for design reviews, educational tasks, or exhibition scenarios, where users may want to highlight objects, leave comments, or mark areas of interest [16].

- **Adaptive mini-map and heatmap design**

  In future work, the mini-map and heatmap could be designed in an adaptive way, where their size and distance from the user are adjustable rather than fixed. This would give users or developers more control to avoid situations where occlusion blocks important parts of the scene. By allowing the interface elements to be repositioned or scaled, it becomes possible to find layouts that balance visibility with comfort. After initial testing in realistic conditions with many users, statistical analysis could then be used to identify which default configuration works best on average. In this way, adaptive controls would not only improve flexibility during development but also help define data-driven defaults for large-scale deployments with potentially millions of visitors.

- **Locomotion comparison.**

Figure 6.4: An educational VR room illustrating a study where physical walking and teleportation were compared for learning scientific content. The findings showed no significant difference in declarative knowledge retention, although continuous joystick locomotion again was not specifically tested [17].

A controlled comparison between continuous locomotion and teleportation refers to experimental studies where both navigation methods are tested under the same conditions, so that their strengths and weaknesses can be observed fairly. In continuous locomotion, the user moves smoothly through the virtual scene, usually by using the joystick on the controller. In teleportation, instead, the user points to a distant location and instantly jumps there. The goal of such studies is to understand how each method affects factors such as user comfort, sense of presence, and spatial orientation. The advantage of continuous locomotion is that it provides a more natural and immersive experience, but it may cause motion sickness for some users. Teleportation, on the other hand, reduces discomfort and is easier for beginners, but it can break immersion and make spatial learning less accurate [17].

Pursuing these directions would transform the present prototype from a proof of feasibility into a broader platform for studying and supporting navigation in complex VR exhibitions.

6.6 Final Remarks

This thesis set out to explore how lightweight orientation aids can support navigation in VR exhibition spaces without overwhelming the user. A headset prototype was built and tested, showing that a compact bundle of aids can be paired with

continuous joystick locomotion in a technically stable way. From these trials, clear design rules and anti-patterns were distilled, offering guidance that goes beyond the goals of existing platforms.

The work is modest in scope, yet it demonstrates feasibility and contributes a practical rule set that others can extend, whether toward controlled comparisons of locomotion, remote large-scale studies, serious-game protocols, or collaborative navigation features. In this sense, the project closes with a simple claim: carefully chosen, short-lived, and centered aids make virtual navigation more grounded, and they provide a foundation for broader research and application in VR exhibitions.

Bibliography

[1] Huizhong Cao, Francisco Garcia Rivera, Henrik Söderlund, Cecilia Berlin, Johan Stahre, and Björn Johansson. Human-centered design of vr interface features to support mental workload and spatial cognition during collaboration tasks in manufacturing. *Cognition, Technology & Work*, Jun 2025.

[2] Evan Cesanek, Sabyasachi Shivkumar, James N. Ingram, and Daniel M. Wolpert. Ouvrai opens access to remote virtual reality studies of human behavioural neuroscience. *Nature Human Behaviour*, 8(6):1209–1224, Jun 2024.

[3] Yi-Ting Chen, Chi-Hsuan Hsu, Chih-Han Chung, Yu-Shuen Wang, and Sabarish V. Babu. ivrnote: Design, creation and evaluation of an interactive note-taking interface for study and reflection in vr learning environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 172–180, 2019.

[4] Seung Jin Chung, So Yeon Kim, and Ki Han Kim. Comparison of visitor experiences of virtual reality exhibitions by spatial environment. *International Journal of Human-Computer Studies*, 181:103145, 2024.

[5] Dimitrios Darzentas, Harriet Cameron, Hanne Wagner, Peter Craigon, Edgar Bodiaj, Jocelyn Spence, Paul Tennent, and Steve Benford. Data-inspired co-design for museum and gallery visitor experiences. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 36:e3, 2022. e3.

[6] Gabriel Di Domenico, Felipe Da Cas Becker, Bento Borges Schirmer, Natan Luiz Paetzhold Berwaldt, Gustavo Machado de Freitas, Alfredo Cossetin Neto, Cesar Tadeu Pozzer, Lisandra Manzoni Fontoura, and Raul Ceretta Nunes. Evaluation of navigation in immersive large-scale vr environments using minimap-assisted teleportation. In *Proceedings of the 26th Symposium on Virtual and Augmented Reality*, SVR '24, page 270–274, New York, NY, USA, 2024. Association for Computing Machinery.

[7] Cheng-Wei Fan, Sen-Zhe Xu, Peng Yu, Fang-Lue Zhang, and Song-Hai Zhang. Redirected walking based on historical user walking data. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 53–62, 2023.

[8] Rubén Grande, Javier A. Albusac, David Vallejo, Carlos González-Morcillo, Santiago Sánchez-Sobrino, and José J. Castro-Schez. Virtual reality shopping-insights: A data-driven framework to assist the design and development of virtual reality shopping environments. *SoftwareX*, 27:101874, 2024.

[9] Pei-Chin Hsu, Sabarish V. Babu, and Jung-Hong Chuang. Comparative evaluation of differing levels of information presentation in 3d mini-maps on spatial knowledge acquisition in vr. In *2025 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 526–536, 2025.

[10] Shinnu Jangra, Gurjinder Singh, Archana Mantri, Zeeshan Ahmed, Tze Wei Liew, and Faizan Ahmad. Exploring the impact of virtual reality on museum experiences: visitor immersion and experience consequences. *Virtual Reality*, 29(2):84, May 2025.

[11] Derya Karadağ. Pre-occupancy evaluation of wayfinding signage using immersive virtual reality. *Design Studies*, 99:101330, 2025.

[12] Matthias Kraus, Hanna Schäfer, Philipp Meschenmoser, Daniel Schweitzer, Daniel A. Keim, Michael Sedlmair, and Johannes Fuchs. A comparative study of orientation support tools in virtual reality environments with virtual teleportation. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 227–238, 2020.

[13] Jong-In Lee, Paul Asente, and Wolfgang Stuerzlinger. Designing viewpoint transition techniques in multiscale virtual environments. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 680–690, 2023.

[14] Imran Mahalil, Azmi Mohd Yusof, Nazrita Ibrahim, Eze Manzura Mohd Mahidin, and Mohd Ezanee Rusli. Virtual reality mini map presentation techniques: Lessons and experience learned. In *2019 IEEE Conference on Graphics and Media (GAME)*, pages 26–31, 2019.

[15] Daniel R. Montello. You are where? the function and frustration of you-are-here (yah) maps. *Spatial Cognition & Computation*, 10(2-3):94–104, 2010.

[16] Mikhail Morozov, Andrey Smorkalov, and Mikhail Fominykh. Sticky notes – a tool for supporting collaborative activities in a 3d virtual world. In *2014 IEEE 14th International Conference on Advanced Learning Technologies*, pages 683–687, 2014.

[17] Michael Rihs, Rahel A. Steuri, Sarah A. Aeschlimann, Fred W. Mast, and Martin Dobricki. Comparison of teleportation and walking in virtual reality in a declarative learning task. *Frontiers in Virtual Reality*, Volume 5 - 2024, 2024.

[18] Arco van Beek, Dorine C. Duives, Yan Feng, and Serge P. Hoogendoorn. Comparison of pedestrian wayfinding behavior between a real and a virtual multi-story building – a validation study. *Transportation Research Part C: Emerging Technologies*, 163:104650, 2024.

[19] Xian Wang, Luyao Shen, Lei Chen, Mingming Fan, and Lik–Hang Lee. Teamportal: Exploring virtual reality collaboration through shared and manipulating parallel views. *IEEE Transactions on Visualization and Computer Graphics*, 31(5):3314–3324, 2025.

[20] Xueqi Wang, Yue Li, and Hai-Ning Liang. Magicmap: Enhancing indoor navigation experience in vr museums. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 881–891, 2024.

[21] Haojun Xu, Yuzhi Li, and Feng Tian. Contrasting physical and virtual museum experiences: A study of audience behavior in replica-based environments. *Sensors*, 25(13), 2025.

[22] Krzysztof Zagata, Jacek Gulij, Łukasz Halik, and Beata Medyńska-Gulij. Mini-map for gamers who walk and teleport in a virtual stronghold. *ISPRS International Journal of Geo-Information*, 10(2), 2021.

[23] Krzysztof Zagata and Beata Medyńska-Gulij. Mini-map design features as a navigation aid in the virtual geographical space based on video games. *ISPRS International Journal of Geo-Information*, 12(2), 2023.

[24] Yuxin Zhang, Boning Zhang, Wansok Jang, and Younghwan Pan. Enhancing spatial cognition in online virtual museum environments: Integrating game-based navigation strategies for improved user experience. *Applied Sciences*, 14(10), 2024.

[25] Mingyan Zou, Shuyang Li, and Chengyu Sun. Optimized and cost-effective behavior studies via vr: a serious game of wayfinding at pvg. *Architectural Intelligence*, 4(1):7, Apr 2025.