Virtual and Augmented Environments

Title: World In Miniature

01/08/2025

SUPERVISED BY
Prof. Hartmut Seichter, PhD

SUBMITTED BY
Roy, Pranto Protim
Matriculation Nr.: 315570
pp.roy@stud.fh-sm.de

Declaration

I, Pranto Protim Roy, declare that this Virtual and Augmented Environments project is my own work, created specially for this course. I have not submitted any materials outside of this course.

During the course of this project, I hardly used AI tools to solve some technical difficulties and exchange ideas about development techniques. This report mentions the help I received from AI. The main ideas, design, and work are my own.

Project Goals

Initially, my goal was to implement World In Miniature (WIM) through virtual reality using the Godot Game Engine.

I wanted to see how two-way syncing works - that is, how it feels when a **RigidBody3D** collides with something in the real world, and how it works in the miniature world as a mirror. Similarly, if the interaction takes place in the mini world, how is that interaction reflected in the real world?

Also I wanted to use as many exported variables as possible, so that this project could be easily maintained in the future.

Theoretical Foundations

The "World In Miniature" (WIM) concept is a well-established virtual reality interaction technique, where a scaled-down version of the entire VR environment (i.e. a miniature copy of the real world/virtual world) is displayed.

Due to significant complexity, WIM is currently focused on the following two features:

1. Full scene visualization
2. Allowing real time object manipulation like moving a book and syncing between both worlds

Inspirations

When I first heard about raycasting and VR from Prof. Seichter, and he demoed them on slides, I immediately became interested in working with VR.

A few days later, the professor demonstrated practical work on VR in the lab and also showed how the miniature system duplicated and worked - which further triggered my interest.

Although the professor explained many things clearly, due to time constraints and lack of skills, the project had to be limited to just one book movement.

Individual milestones

While practicing in the lab, the professor showed me how to create a world setup structure that can be easily duplicated as a Miniature World. Accordingly, I created the structure as described below:

- Static Scenes:
    - Wall & Ceiling: In the real-world environment, I utilized **StaticBody3D** nodes representing walls and a ceiling to simulate the feeling of being inside a room. These nodes are excluded from the Miniature World for clarity. The **.tres** resource files for both the wall and ceiling were sourced from online 3D asset repositories. References to these resources are documented in the reference section of the report.
- Dynamic Scenes: All nodes under the dynamic scene are visible and interactable in the Miniature World. These include objects that require syncing and visual consistency across both worlds.
    - Floor & Table: The floor and table were implemented using **StaticBody3D** nodes to ensure they behave as non-movable physical surfaces.
    - Book: The book is the only **RigidBody3D** used in the scene. It is designed to be movable and interactable in both the Real World and the Miniature World. Its transform is dynamically updated to reflect interactions from either world, supporting bidirectional syncing.

      (By the way, I added Label3D nodes for the both Table and Book to identify)

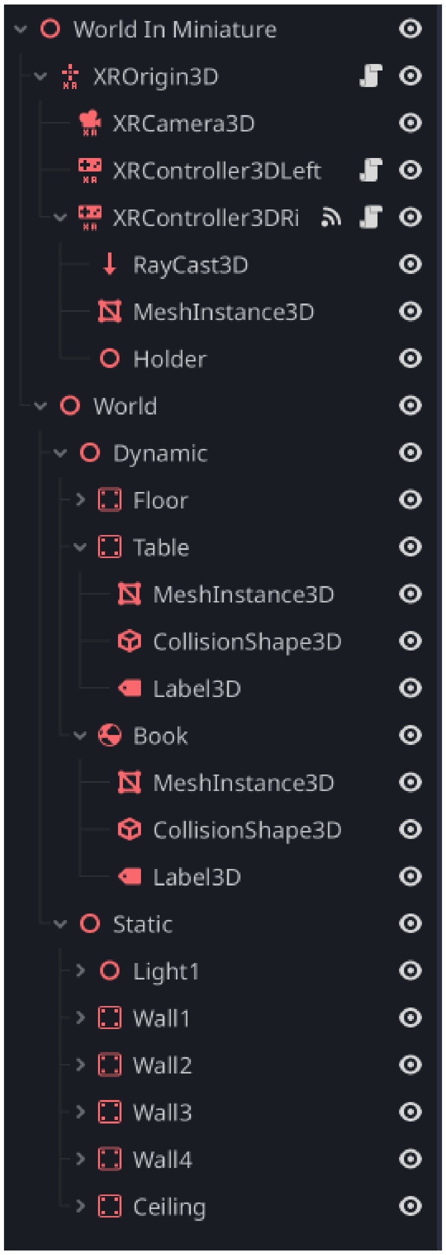


Figure 1: WIM Screenshot in Godot

Basically, the **Right Controller** is used to handle raycasting. If the raycast collides with a **CollisionObject3D**, that object can be grabbed. However, in this project, grab and release functions have been implemented only for the Book object. The book can be grabbed and moved to another location by pressing the button on the controller. Upon release, the book's new position is updated immediately.

This location update is done in real time, and the user can see it directly in the Real World. After release, the updated location of the book is also reflected in the Miniature World.

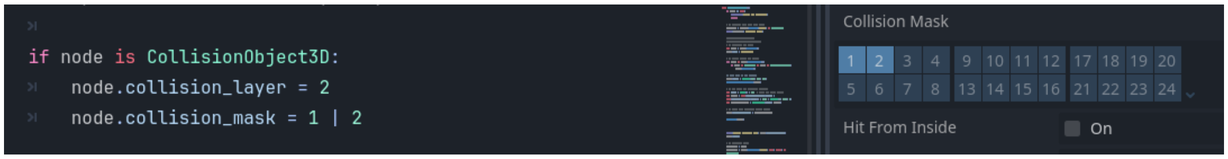
Since the project focuses on bidirectional interaction, I decided to add a visual indication in both worlds when hovering.

- When hovering over the book in the Real World, it turns yellow, and the same change is seen in the book in Miniature World.
- Again, when hovered over in Miniature World, the book turns cyan and that color is also reflected in the book in the Real World.

This color interaction was extremely helpful in debugging, as it showed exactly which objects the raycasting was colliding with.

However, a major problem arose when creating the Miniature World—its objects were physically colliding with objects in the Real World (e.g., books in the Real World were colliding with the Mini World). To solve this problem, I disabled the Miniature World's physics completely after duplicating.

The biggest challenge was to make objects collide in the Miniature World through raycasting. Initially, raycast was not hitting objects in the Mini World. After trying multiple strategies and failing, I took help from AI. I got a great suggestion from AI - it was about the concept of layering.



```
if node is CollisionObject3D:
    node.collision_layer = 2
    node.collision_mask = 1 | 2
```

Collision Mask: 1, 2 selected

Hit From Inside: On

Figure 2: Collision layer and mask configuration for Miniature World objects (xr_controller_3d_left.gd)

The image above shows:

- collision_layer = 2, indicates that this object is part of the Miniature World.
- collision_mask = 1 | 2, indicates that it can accept raycasts from two layers - Layer 1 (Real World) and Layer 2 (Miniature World).

As a result, bidirectional syncing between the two worlds has been successfully implemented.

Limitations & Future Works

Due to my lack of skills and time constraints, there are still two major problems with the current version of my work:

1. Miniature World Boundary: If I grab the book in Miniature World and release it somewhere outside the world, the book falls down and disappears.
2. Book movement in Miniature World: I've noticed many times that when I grab the book in Miniature World and release it, it goes back to its previous position. Though this is not regular, I've tried a few ways to fix this, and unfortunately none of them worked.

I plan to solve these problems in the future and expand the whole project into a Building Game. There will be bricks of different sizes, which can be used to build a building. Most importantly, there will be a specific pattern for building and the user will be notified whether that pattern has been followed or not.

References

Papers:

1. Bowman, Doug A., and Larry F. Hodges. 1997. “An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments.” In Proceedings of the 1997 Symposium on Interactive 3D Graphics - SI3D ’97, 35–ff. Providence, Rhode Island, United States: ACM Press. *https://doi.org/10.1145/253284.253301.*
2. Stoakley, Richard, Matthew J. Conway, and Randy Pausch. 1995. “Virtual Reality on a WIM: Interactive Worlds in Miniature.” In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI ’95, 265–72. Denver, Colorado, United States: ACM Press. *https://doi.org/10.1145/223904.223938*.

Textures:

1. *https://polyhaven.com/a/wood_planks*
2. *https://polyhaven.com/a/stone_tiles_02*
3. *https://polyhaven.com/a/painted_brick*