Unity VR Development Test Project for Interactive Realities Lab

Jai Sai Naga Venkat Guttikonda
jsnvg@iastate.edu

Abstract

This report documents the design and implementation of a virtual reality (VR) escape-room style application, completed as part of the Interactive Realities Lab's *Basic Unity VR Challenge* under Dr. Kopper. The challenge was intended to test students' ability to set up Unity XR projects, implement interactive 3D elements, and deliver a polished, user-friendly VR application within a two week timeline. The project simulates an escape-room scenario where players must solve three interconnected puzzles: entering a keypad code, rotating a valve with two hands, and inserting a key into a socket. These interactions were chosen to highlight different aspects of XR development, such as user interface design, physics-based manipulation, and object validation through sockets.

1 Introduction

For this project, I selected an **escape room** theme because it naturally combines multiple types of puzzles and encourages varied forms of interaction. This decision allowed me to experiment with UI-based input (a keypad), physics-driven mechanics (a rotating valve), and socket-based validation (a key insertion). The environment was deliberately kept small: a single room with a locked exit, so that focus could remain on the interactions themselves. As a first attempt at XR development, the project not only delivered the required features but also taught me valuable lessons about workflow in Unity: structuring scripts, testing frequently in VR hardware, and debugging issues such as falling rigidbodies and misaligned socket anchors. These lessons shaped the final outcome as much as the design of the puzzles themselves.

2 Methods

2.1 Unity Setup

Development was carried out in Unity 6 with the XR Interaction Toolkit, using a Meta Quest 2 headset provided by the IR Lab for testing. This was my first time setting up an XR project, so a significant part of the process involved exploring Unity's XR configuration workflow.

2.2 Puzzle Design

The escape-room environment was built around three core puzzles, each designed to highlight a different type of XR interaction:

- **Keypad Puzzle:** Implemented as a world-space UI canvas with interactive numeric buttons. Entering the correct four-digit code (4271) unlocked the keypad lock. A small display and

audio cues were added to make the interaction intuitive and to reinforce successful or failed attempts.

- **Valve Puzzle:** Designed as a rotating valve object that required two-hand manipulation. This puzzle was challenging to implement because I had to track the rotation angle while ensuring that only bi-manual grabs would activate the unlocking logic. A custom script monitored both hands and confirmed a turn of at least 90° before signaling the lock as solved.
- **Key Puzzle:** A golden key was modeled as a grabbable object (XR Grab Interactable) and validated by insertion into a socket (XR Socket Interactor). The socket was configured to accept only objects tagged as "Key." Upon insertion, an audio clip played and the lock was flagged as solved. This was the most technically demanding puzzle, as it required careful tuning of socket attach points, collider alignment, and interaction layers. It was also the most rewarding because it demonstrated the concept of object validation and selective acceptance.

2.3 Door and Lock Management

The escape-room door acted as the central focus of the project. It was modeled as a rigidbody constrained by a `HingeJoint`, allowing it to swing naturally once unlocked. A custom script, `LockManager.cs`, coordinated the three puzzles: it tracked the state of each lock (Keypad, Key, Valve), updated the indicator lights above the door (red → green), and toggled the rigidbody's kinematic state. Initially, the door was kept locked and immovable. When all three puzzles were solved, the lock manager released the constraints, played an unlocking audio cue, and allowed the door to swing open on its hinges. Building this system taught me how to centralize game state logic and connect multiple puzzle scripts into one coherent user flow. Although the DoorLeaf seem to be locked. It was sucessfully made and unlocked.

3 Results

During testing, the following outcomes were observed:

1. **Puzzle Interactions:** Each puzzle responded to natural hand interactions. The keypad buttons were pressed directly with controllers, the valve required grabbing with both hands to rotate, and the key could be picked up and inserted into the socket.
2. **Feedback Systems:** Visual, auditory, and physical feedback reinforced progress. LEDs above the door changed from red to green as each lock was solved, audio clips confirmed correct and incorrect actions, and physics constraints ensured objects behaved believably.
3. **Door Unlocking:** When all three puzzles were solved, the door transitioned from locked to free-moving.

A short demonstration video was recorded to capture this demo project.

4 Discussion and Evaluation

4.1 Technical Implementation

The project demonstrates correct use of the Unity XR Interaction Toolkit, with puzzles designed as XR Interactables and validated through custom scripts. A centralized `LockManager` script maintained the overall state of the escape-room, simplifying coordination between puzzles and the door.

This modular approach not only reduced complexity but also mirrored real development practices where multiple systems feed into a shared manager. Implementing the valve and key socket required careful handling of physics and anchors.

4.2 User Experience and Usability

Each puzzle provided multimodal feedback: LEDs indicated progress, audio cues confirmed correct or incorrect actions, and the physical resistance of colliders and hinge joints reinforced realism. Scene composition was designed intentionally so that the puzzles were easy to find but not overwhelming. For example, the keypad was placed directly in view, the bookshelf invited exploration, and the door itself served as a constant visual reminder of the goal.

4.3 Completion of Core Tasks

The challenge requirements were successfully met. They are a hinged door constrained by a `HingeJoint`, locked until all conditions were satisfied. Multiple grabbable and socketable objects implemented using XR Interactables. A bi-manual valve puzzle demonstrating physics-based constraints. A keypad puzzle integrating world-space UI with XR input. Together, these created a self-contained escape sequence that highlighted a variety of XR interaction styles. Some small tasks such as highlighting an object on hover or bringing a code paper to lamp to make the code appear clear, material binding etc., were also made as per the project requirements.

4.4 Innovation and Learning

The project went beyond isolated demos by combining all puzzles into a cohesive escape-room scenario. This added a layer of narrative and motivation, which improved the player's sense of progression. More importantly, as my first XR project, the work was a valuable learning experience. I encountered and overcame practical issues such as rigidbodies falling through the floor, sockets refusing to accept objects, and colliders misaligned with models. Each challenge required reading Unity's XR documentation, testing in VR hardware, and iterating on design.

5 Conclusion

The VR escape room project successfully fulfilled the objectives of the Interactive Realities Lab challenge by integrating Unity XR fundamentals, interactive puzzle design, and multimodal feedback into a cohesive application. The final experience allowed players to move naturally in VR, interact with objects in intuitive ways, and receive clear feedback as they progressed toward unlocking the exit door. Beyond achieving the technical goals, the project served as an introduction to the full workflow of XR development. I learned how to configure Unity for VR, build physics-driven puzzles, and design around comfort and usability. The iterative process of testing, debugging, and refining each puzzle was as educational as the final implementation itself.

Links

Project Repository: GitHub Link
Demonstration Video: OneDrive Link