Bizarre Museum: A Mixed Reality Dream Curator

Technical Documentation & Interaction Report

Nouraldin Gaser Yousri Algendi
Debaj Shady Elmaghraby
German International University (GIU)
Course: Extended Realities (Winter 2025/2026)

January 11, 2026

Abstract

Bizarre Museum is a mixed-reality prototype that bridges the gap between Mobile AR and VR. The experience gamifies the collection of "dream fragments" in the real world (AR) and allows users to curate and experiment with these fragments in a surreal virtual museum (VR). This report documents the system architecture, specifically the "Seed/DNA" data model, the interaction design implemented using Unity's UI Toolkit, and the technical implementation of the cross-platform experience.

Contents

1 Project Overview

Bizarre Museum explores the concept of object permanence and transformation across different realities. The core loop consists of two distinct phases:

1. **The Collector (AR):** Users capture floating "dream bubbles" in their physical environment using a mobile device.
2. **The Curator (VR):** Users utilize an access code to retrieve their collected items in a virtual space, where they can manipulate the items' parameters to create surreal environmental effects.

2 System Architecture

The project utilizes a "Client-Authority" architecture pattern to handle the procedural generation of item behaviors.

2.1 The "DNA Bank" Philosophy

To allow for complex, randomized behaviors (gravity distortion, color shifting, audio synthesis) without heavy server-side computation, we implemented a **Seed-Based Architecture**.

- **The Server (Node.js/Express + MongoDB):** Acts as a passive state container. It does not store physical parameters (e.g., "Gravity Force = 9.8"). Instead, it assigns a random floating-point `seed` $(0.0 - 1.0)$ to every item collected.
- **The Client (Unity):** Acts as the "DNA Unpacker." It receives the seed and uses it to initialize a deterministic Random Number Generator (RNG). This ensures that a specific seed always produces the exact same set of complex behaviors on any device.

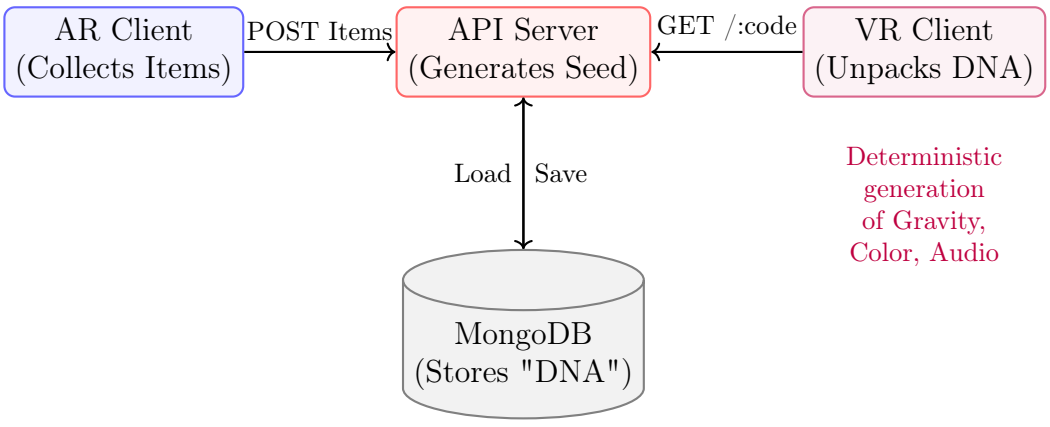

Figure 1: Data Flow: From AR Collection to VR Generation

3 Interaction Design & Implementation

3.1 Phase 1: AR Interaction (The Collector)

The AR phase focuses on physical movement and discovery.

3.1.1 Bubbles & Raycasting

Floating "Dream Bubbles" are instantiated around the user in world space. We implemented a Raycast system that translates 2D screen taps into 3D world interactions.

- **Interaction:** Tapping a bubble "pops" it, dropping the primitive item to the floor.
- **Collection:** Proximity triggers (Colliders) detect when the user walks over an item, adding it to the inventory.

3.1.2 Inventory System (UI Toolkit)

We utilized Unity's **UI Toolkit (UXML/USS)** for a performant, resolution-independent HUD.

- The inventory is a flex-container that dynamically fills as items are collected.
- Upon collecting 3 items, the "Upload Machine" appears.

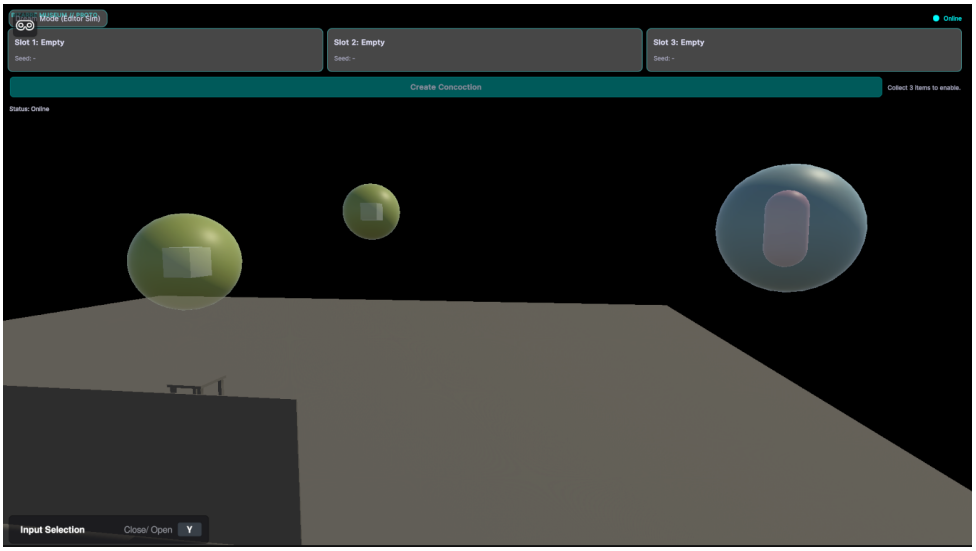

Figure 2: AR Interface: Dream Bubbles and Inventory System

3.2 Phase 2: VR Interaction (The Curator)

The VR phase shifts focus to experimentation and environment manipulation.

3.2.1 The Login Terminal

Upon entering the scene, the user is presented with a holographic keypad created with UI Toolkit.

- The user enters the 6-character code generated in the AR session.

- An API call retrieves the specific configuration of items.

3.2.2 Pedestals & Auras

The room contains three interactive pedestals. Placing an item on a pedestal triggers its `Configure(seed)` method.

- **Gravity Anomaly:** Uses the seed to determine force magnitude and direction, physically rotating and lifting nearby objects.
- **Chromatic Shifter:** Uses the seed to select a color palette (Gradient) and cycle speed, affecting the materials of the room.
- **Humming Relic:** Uses the seed to generate audio pitch and modulation speed.

Figure 3: VR Museum: Items activated on pedestals showing aura effects

4 Implementation Details

4.1 The "Unpacker" Pattern

To ensure the same "random" behavior occurs on both the developer's machine and the user's device, we use `System.Random` initialized with the downloaded seed.

```
public override void Configure(float seed) {
    // Convert normalized float (0.0-1.0) to integer seed
    int numericSeed = (int)(seed * 100000);
    System.Random rng = new System.Random(numericSeed);

    // Deterministically generate parameters
    float intensity = (float)rng.NextDouble(); // Always same for this
    seed
    float range = (float)rng.NextDouble() * 5.0f;

    // Apply to Aura
    auraController.SetIntensity(intensity);
    auraController.SetRange(range);
```

```
}
```

Listing 1: C# Implementation of Seed Unpacking

4.2 Input Abstraction

To facilitate rapid testing in the Unity Editor without deploying to devices constantly, we implemented an `InputAdapter` class. This wrapper intercepts input calls, automatically switching between:

- **Editor Profile:** Mouse clicks for taps, WASD + Right Click for movement/look.
- **Device Profile:** Touch inputs (AR) and XR Controller inputs (VR).

5 Conclusion & Future Work

The prototype successfully demonstrates a continuous data loop between AR and VR environments. The "Seed" architecture proved highly effective for synchronizing complex procedural behaviors with minimal data transfer. Future iterations will focus on replacing primitive shapes with scanned 3D assets and implementing spatial audio layers for the Humming Relic.