Software Design and Architecture

Introduction

This document outlines the architectural and technological background for an Augmented Reality Physics Game utilizing the Quest 3.

Architectural Goals and Principles

The goals of this architecture is to:

- Ensure quality / scalability
- Minimal coupling
- Very little stress
- Separation of concerns

System Overview

In the context of a Meta Quest application, the system is streamlined and self-contained, with a focus on delivering an immersive mixed reality experience directly on the Quest device.

Unreal Engine / Application Logic: Unreal Engine, the powerful game engine, allows for the creation of the immersive 3D environment inside the clear box and high-quality graphics for virtual bouncing balls. The core logic of the mixed reality application is developed using Blueprint scripting and C++ for the physics.

User Interface: Unreal Engine will render holographic user interfaces like floating sliders for game variables like gravity, ball bounciness among other features enabling a seamless interaction between the user and the mixed reality environment.

Architectural Patterns

Our final product is going to be a single executable, and this will not have any external dependencies. Everything the executable does will be done by itself, so we do not have a specific pattern.

Component Descriptions

- Quest 3 Device: The core hardware component is the user's gateway to the mixed reality environment. Its responsibilities include tracking user movements, capturing audio, and processing user inputs such as gestures and voice commands.
- User Interface: Handles user interactions and allows them to interact with the environment
- Database: Stores variable data entered by the user for the simulation

Data Management

User's data will be kept track of, and will be compared to other user data to show how well people did in the game. We will also need to keep track of whether someone wins or loses the game.

Interface Design

No interactions within our program will rely on outside information. There may be some interactions a user can make that change values but they will all be local to the application and not need any thorough backend design.

Considerations

Security

We will not need to worry about security since our game will not be connected to the internet.

Performance

Performance requirements for our project would include something like how quickly/smoothly the application actually runs. It needs to run smoothly enough to provide the user a comfortable experience while using it, this tends to be fairly critical for AR/VR applications.

Maintenance and Support

There are currently no plans to maintain or support this application after we have finished developing it.

Deployment Strategy

We will be developing our game by writing code, presumably with Unreal Engine, and adding it all together through GitHub and Pull Requests. However, we won't have an architecture for our AR game, so there isn't going to be much else to this deployment strategy.

Testing Strategy

Most of our testing can be done through playtests within our team, in a simulation environment, or through public playtests (likely roommates, friends, etc.). Our goal is to make sure that everything runs smoothly when playing, so playing the game more and trying to break it is our best method for finding errors.

Glossary

AR - Augmented Reality
VR - Virtual Reality