Motion: Virtual Physiotherapy Technical Documentation

by

Amine El Hammdaoui, Hoemeirra Abdoelrahman, Gavin Tjin-A-Soe,
Patrick Wals, Ilias Morabet, Susant Budhathoki

Amsterdam University of Applied Sciences
Developed using Unity 6 (6000.0.40f1)
June 2025

Table of Contents

Preface

This technical documentation was created to accompany the development of Motion: Virtual Physiotherapy, a VR application designed to assist users with lower back rehabilitation. It is intended for developers, academic evaluators, and technical stakeholders involved in the design, implementation, and future expansion of the project. The document outlines the system's architecture, development environment, core features, and implementation details to support both maintenance and continued development.

Chapter 1 | Project Overview

Motion: VR Physiotherapy is an immersive virtual reality application developed to assist users suffering from lower back pain. By guiding users through physiotherapist-approved exercises in an engaging VR environment, the app aims to improve compliance, form accuracy, and rehabilitation outcomes.

1.1 | Target Audience

- Adults aged 18-35 with mild to moderate lower back pain
- Physiotherapists seeking tools to remotely monitor and guide their patients

1.2 | Primary Goals

- Provide real-time guidance and feedback during physiotherapy routines
- Collect movement data for performance tracking
- Enable physiotherapists to tailor routines based on patient needs

1.3 | System Requirements

Hardware:

- Meta Quest 2 or newer VR headset
- 2.0m x 2.0m minimum play space
- Wi-Fi connection for data sync (optional but recommended)

Software:

- Unity 6 (6000.0.40f1)
- XR Interaction Toolkit (v3.0+)
- OpenXR Plugin
- Custom-built backend (optional, for data sync)

Rendering Pipeline:

- Unity Universal Render Pipeline (URP)

Input System:

- Unity Input System Package (v1.7+)

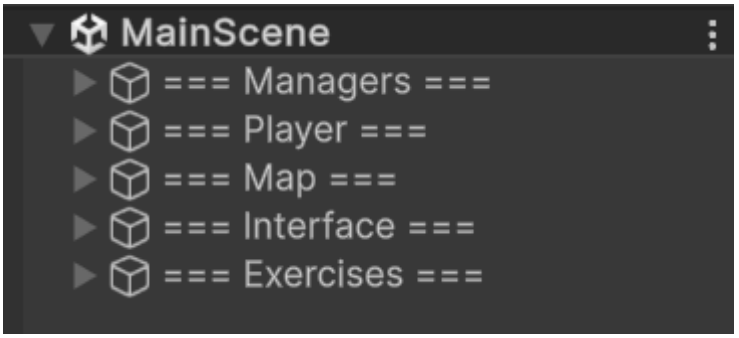
1.4 | User Flow

1. **Launch App**
2. **Intro & Safety Instructions**
3. **Balance Test** (required every day)
4. **Exercise Selection** (if balance test completed)
5. **Guided Exercise Session**
6. **Session Summary**
7. **Manual Exit**

2 | Application Architecture

2.1 | Scene Hierarchy:

Everything happens in the same scene. The hierarchy consists of the following 5 objects:



2.1.1 | === Managers ===

Contains global information and systems like the ExerciseManager, SoundManager or the MapManager. Children of this object are mostly singleton-style objects or objects that control core logic and game flow.

2.1.2 | === Player ===

Holds the camera and the VR system.

2.1.3 | === Map ===

Contains 4 different maps. Each of these maps are operated by the MapManager.

2.1.4 | === Interface ===

Contains all of the UI.

2.1.5 | === Exercises ===

Contains very specific exercise references like the NPC, affordances or footprint images.

2.2 | Code design

2.2.1 | Front-end UML

Documentations/Unity/MotionUML.pdf

2.2.2 | Back-end UML

Documentations/API/Technical document.pdf

2.3 | Key classes

Exercises

- Exercise system (ExerciseManager, Exercise & GenericExercise)
  Documentations/Unity/Exercises.md
- DifficultyManager
  Documentations/Unity/DifficultyManager.pdf
- FeedbackManager
  Documentations/Unity/DifficultyManager.pdf
- PositionChecker
  Documentations/Unity/Posture.pdf
- Firefly
  Documentations/Unity/FireFly.pdf

Back-end

- API
  Documentations/Website/VR Balance technical document.pdf
- Deployment
  Documentations/Website/VR Balance deployment.pdf
- Website
  Documentations/Website/VR Balance technical document.pdf

General

- AudioManager
  Documentations/Unity/SoundSystem.pdf
- MapManager
  Documentations/Unity/MapManager.md

Chapter 3 | Back-end

3.1 | Data Logging & Integration

What is Logged:

- Balance test results
- Time spent on each exercise
- User posture deviations (basic tracking)
- Routine completion status

Data Handling:

- Stored locally with optional upload to central server
- Physios access data via dedicated web portal
- Based on results, physios can assign new routines or review progress

3.2 | Store and fetch data using the API

To store or fetch data from the server, navigate to the service/ folder. Each service class in this folder contains the logic for making structured API calls to the backend. For local development, make sure to update the baseUrl in the ApiClient to: http://localhost:8080/api and that the backend is running.

Important: When sending data to the server, ensure that your **Data Transfer Object (DTO)** in the frontend (Unity) exactly matches the corresponding DTO in the backend.
You can refer to the existing methods in the service files as examples.

3.3 | Authentication

To track your progress, you must log in. In the Unity app, users can log in using their username and pincode. After successful authentication, the JWT token returned by the server is stored in PlayerPrefs. Additionally, the username is stored locally in a JSON file to simplify the login process the next time the app is launched. Navigate to the files **LoginManager.cs** and **User.cs** to see how this functionality has been implemented.

Chapter 4 | Development

4.1 | Feature Roadmap

- Either recentering the World/UI, or not requiring the user to be exactly at 0, 0, 0 when setting the target positions during generic exercises.
- Progress UI contains things like a headway graph, achievements and a streak calendar..
- City map.
- Each exercise has a personal leaderboard (NOT GLOBAL, because the app shouldn't be competitive). Some exercises have special stats like "amount of fireflies caught" for the firefly game.
- The NPC explains the movements as if it's an interactive exercise.
- Subtitles.
- Expand the exercise library with more gamified exercises that challenge the user's balance.
- Add physiotherapist-side app for in-VR consultations.
- Voice command integration.
- Colorblind and contrast-friendly modes.

4.2 | Known Issues

- The calibration isn't implemented properly, nor is it saved in the back-end. Exercises assume that the user is 1.75m.
- The NPC and its animations aren't implemented properly and don't correspond with the actual required movements.
- The HeightThresholdBehavior crashes the game if a movement has it. This behaviour is required for specific exercises to function properly like the Squat, Lunge or Glute Bridge.
- Occasional floor alignment bug on Quest standalone mode

4.3 | Debugging

The application is built with the Unity editor in mind. You can test most of the functionality without a VR headset, but exercises that rely on controller input won't work properly without one.

To test in VR, you can simply connect your headset with a cable. This allows you to run the scene in Play Mode with the same functionality as a built version. The only exception is passthrough, which only works in an actual build.

The UI can be interacted with directly in the editor. Pressing the spacebar also lets you skip certain movements and timers, which is useful during testing.

If any further assistance is needed, contact Amine El Hammdaoui on Teams and expect a fast response.