VRMTraining Guide

> Dissertation Repository

Table of Contents

Virtual Reality Mirror Training-based System

Unity Project Setup

To successfully open the VRMTraining project in Unity, ensure you are using Unity Editor version 2022.3.13f1 in the Unity Hub. If not, install it.

> Safety Mode Message: While opening the Unity project, if you receive a message asking you to "Enter Safety Mode" or "Ignore," choose "Ignore." This will not cause any issues.

> Unity Console: Once the project is opened, if red errors in the Console tab persist after clicking the "clear" button, you need to fix them. Otherwise, it can be safely ignored and click Play anyway.

> Missing Assets: In the hierarchy, you may see several objects in red, which indicates that some assets are missing from the project. You'll need to add these assets from the Unity Asset Store:

- **MVN Live Animation**: Follow this **tutorial** to connect Xsens with Unity.
- **Ready Player Me**: Follow this **tutorial** to connect RPM with Unity.
- **Free emojis pixel art**.
- **Apartment Kit**.
- **Minimalist ArchViz Bedroom**.
- **HDRP Furniture Pack**.
- **Chair and sofa set**.
- **Office Room Furniture**.
- **Morgue Room PBR**.
- **Lamp Model**.
- **Free Pack Woden Planks**.
- **SteamVR Plugin**.
- **PBR Folding Chairs**.

Activity Identifier

In the experimental protocol, four (4) distinct motor activities are performed, based on the UPDRS III scale. Each activity is assigned an acronym and identifier for easy reference:

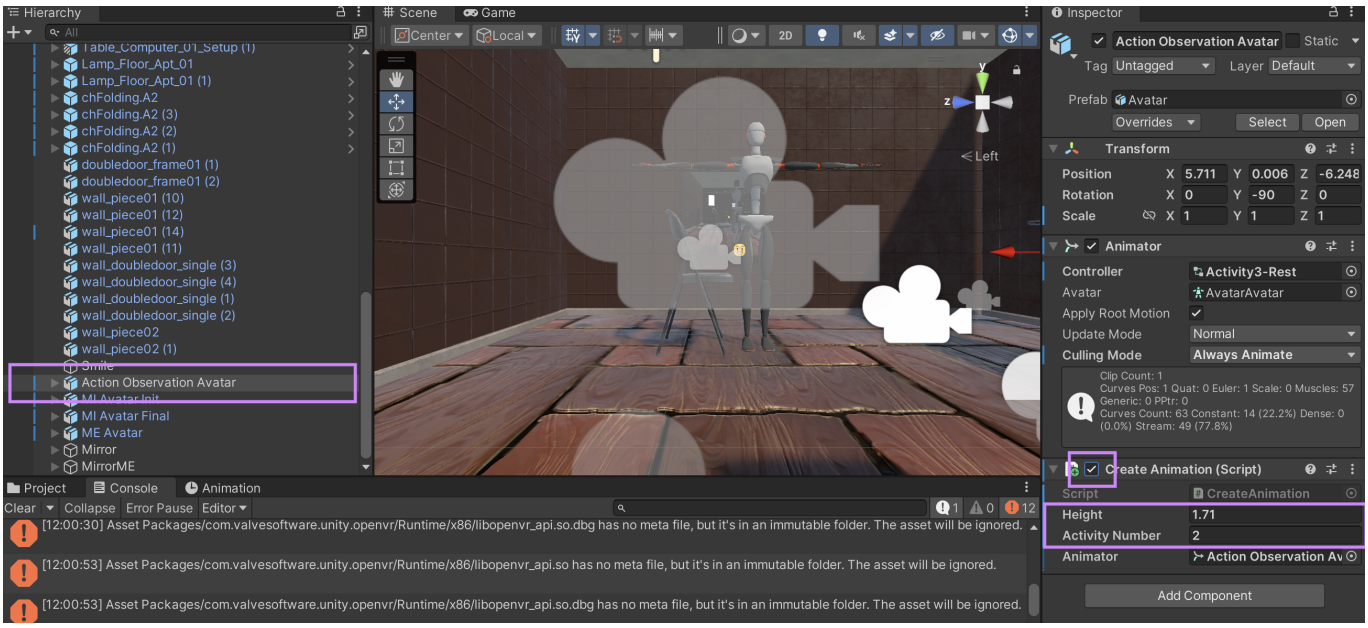
```
- Activity of "Lifting The Leg" = ACT1 (ID: 1)
- Activity of "Arising From A Chair" = ACT2 (ID: 2)
- Activity of "Extending The Arms Forward" = ACT31 (ID: 31)
- Activity of "Touching The Nose" = ACT32 (ID: 32)
```

These acronyms are used throughout the guide and script documentation for simplified reference, while the identifiers serve as inputs in various Unity components, as explained in the following sections.

Creating Animation Clips

```
- Which script creates the animation clips for each activity?
+ CreateAnimation.cs (Folder: VRMTraining/Assets/Samples)
```

To generate animation clips for a specific activity, you can add the ***CreateAnimation*** component to the AO Xsens avatar object (Figure below). This component requires two inputs: the `participant's height` and the corresponding `activity identifier`. If you no longer need to create animations, you can easily disable the component by unchecking its box, as illustrated below.



After entering these inputs, press Play in Unity, and the animation clip will be automatically generated and saved in the following folder: VRMTraining/Assets/Resources/Animations. The reference learning datasets used to generate these animation clips are located in the main directory: VRMTraining.

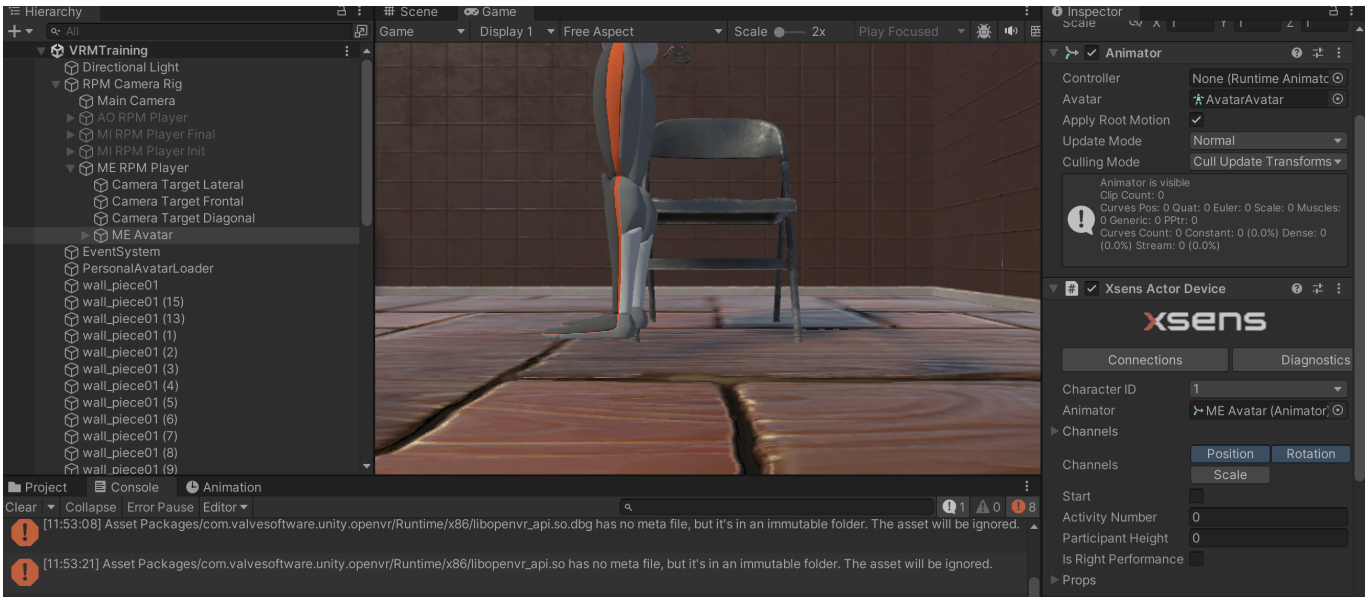
RPM Scripts

From all the Ready Player Me (RPM) scripts (located in the folder: VRMTraining/Assets/Samples/Ready Player Me Core/6.3.1/QuickStart/Scripts), only ***ThirdPersonLoader.cs*** was modified. Specifically, the

following functions were updated: *SetupAvatar*, *LoadAvatar*, *SetupController*, *SetupAvatarLink*. These functions were modified to enable the customization of the avatar's appearance, controller, and its position and rotation, being the position and rotation tailored to a specific activity.

The only script created from scratch is ***ConfigureActivityDisplay.cs***, which needs to be added as a component to the GameObject RPM Camera Rig and take as input the `avatar url` (from the RPM avatar), `the activity identifier` (either 1, 2, 31 or 32) and the `task acronym` (either "ao", "mi" or "me"). This component is responsable for **(i)** personalizing the avatar's appearance based on the url provided, **(ii)** configuring the camera's perspective according to the activity identifier, and **(iii)** displaying only the avatars associated with the selected task acronym.

When selecting the task acronym "ME" and pressing Play, an additional step is required. You have to add the ***Xsens Actor Device*** component to the object ME Avatar, generated inside the GameObject ME RPM Player within RPM Camera Rig (Figure below). This Xsens component enables the avatar to replicate the participant's movements in real time, and requires these inputs: the `activity identifier` and the `participant's height`. Additionally, the component provides two checkboxes: `start`, to begin providing feedback, and `isRightPerformance`, which should be checked when the right limb performs the activity (e.g., ACT1 and ACT32), and unchecked for left limb performance. Once these inputs are added, the avatar will be configured to move in real-time based on the participant's movements. **Note**: currently, an issue that still needs to be resolved is the alignment of the avatar with the virtual chair. The avatar's seated position needs to be fixed to ensure proper alignment within the virtual environment.



Xsens Scripts

To enable the streaming of multiple data packets from Xsens to Unity, rather than just a single packet, several Xsens scripts were modified. The following scripts and their respective functions were updated:

- ***XsDataPacket.cs***: This script contains the class *XsDataPacket*, which handles the decoding and parsing of data packets based on their type.
- ***XsQuaternionPacket.cs***: The *parsePayload* function in this script was updated to enhance the parsing of data packets based on their type.
- ***XsensClient.cs***: The cases `StreamingProtocol.SPLinearSegment` and `StreamingProtocol.SPCOM` were added in the *ListenForDataAsync* function, allowing the system

to receive these specific data packets. Additionally, the *ParsePacket* function was modified to accommodate these changes.
- ***XsensTypes.cs***: Two new variables, `Velocities` and `PositionsCOM`, were added to the *FrameData* structure.
- ***XsensDevice.cs***: Modifications were made to the *UpdateMinMaxValues* and *ProvideFeedback* functions. All changes within the section marked `[START/END] Our code` pertain to our custom modifications.

Each of these scripts is thoroughly documented, providing explanations on their functions and purposes. For optimal understanding and reference, the scripts should be consulted in the following order: **(1)** *XsDataPacket.cs*, **(2)** *XsQuaternionPacket.cs*, **(3)** *XsensClient.cs*, and finally **(4)** *XsensDevice.cs*. This sequence will help you follow the changes made and better comprehend the data streaming process involving multiple packets.

Other Scripts

Another script created from scratch is ***MirrorFlipCamera.cs*** (folder: VRMTraining/Assets/Samples), designed to flip the camera horizontally (with an option for vertical flip as well), providing a mirrored perspective of the avatar. This mirrored view is especially useful for the observation of activities ACT31 and ACT32, with a clear frontal view of the upper limbs. The component is meant to be activated on the GameObject corresponding to the AO and ME RPM avatars. When the horizontal flip is applied, however, an issue occurs: the background becomes distorted, and the avatar is no longer clearly visible. **This issue still needs to be resolved to optimize the VRMT.**

Biomechanical Results

```
- Which script implements processing methods for biomechanical outcomes?
+ process-data.py
```

This file contains the data processing methods used to **segment**, normalize, interpolate, **calculate the mean and standard deviation curves**, **plot** data and **extract the features** for both healthy and PD patients. The results are available on the shared drive in the folder plots/healthy-participants for healthy participants and in plots/pd-patient for the PD patient.

Questionnaire Results

```
- Which script implements ploting methods for questionnaire outcomes?
+ plot_questionnaire_results.ipynb
```

The ***plot_questionnaire_results.ipynb*** file contains the ploting methods used to visualize the results from the IPQ, IMI and KVIQ questionnaires. This includes both the mean responses of all healthy participants and the individual responses of the PD patient. All generated plots and visual representations are stored in the plots folder on the shared drive.