**Developer’s Guide for the VR Biotech Lab Simulation**

**Overview**

This guide provides comprehensive instructions for setting up, extending, and maintaining the VR Biotech Lab Simulation. It targets developers who intend to update the development environment, integrate new features, or expand existing functionalities. Please follow all steps carefully, as even minor omissions may lead to issues in setup or development.

**Development Environment Setup**

**Prerequisites**

* **Operating System:** Windows 10/11 (recommended), MacOS compatibility is limited
* **Hardware:** VR-ready PC with a Meta Quest 3 headset (or equivalent)
* **Software Requirements:**
  + Unity Hub
  + Unity 2022.3.5f
    - Required for compatibility with XR Plugin features.
  + Meta Quest Developer Hub (MQDH)
  + Android SDK & NDK
  + Visual Studio 2022 with Unity workload (for C scripting)
  + Git for version control

**Installation Steps**

1. **Install Unity**
   1. Download Unity Hub: (<https://unity.com/download>).
   2. Install Unity Editor version 2022.3.5f1 or later with Android Build Support.
2. **Install Meta Quest Developer Hub**
   1. Download MQDH: (<https://developer.oculus.com/downloads/>).
   2. Use MQDH to set up your VR headset for development.
3. **Install Android SDK & NDK**
   1. Navigate to Unity Preferences > External Tools and install the required Android SDK, NDK, and OpenJDK through Unity.
4. **Clone the Repository**
   1. git clone <[engr-csc-sdc/2024Fall-Team33-Srougi](https://github.ncsu.edu/engr-csc-sdc/2024Fall-Team33-Srougi)>
   2. cd <[engr-csc-sdc/2024Fall-Team33-Srougi](https://github.ncsu.edu/engr-csc-sdc/2024Fall-Team33-Srougi)>
5. **Open the Project in Unity**
   1. Open Unity Hub.
   2. Click on "Add," navigate to the project folder, and select it.
6. **Install Dependencies**
   1. The project uses Unity's XR Interaction Toolkit:
      1. Open `Window > Package Manager`.
      2. Search for and install "XR Interaction Toolkit" and "XR Plugin Management."
   2. Set up OpenXR as the XR plugin under `Project Settings > XR Plugin Management`.
7. **Run the Project**
   1. Connect the VR headset via USB.
   2. Use the Play button to start the simulation in tethered mode.

**Custom Controller Development Setup**

**Install Arduino IDE**

You can use either the Arduino IDE Version 2 or the Legacy IDE Version 1.8.X - the Version 2 is more of a traditional IDE while the Legacy version is simpler and a bit easier to use. The instructions in this guide are based on the Legacy IDE version 1.8.19.

* Download and install the Arduino IDE from this link: <https://www.arduino.cc/en/software>
* Instructions on installing the IDE are listed here: <https://docs.arduino.cc/software/ide/#ide-v2>

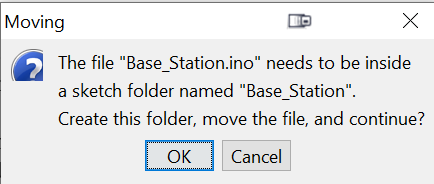
**Set Up Seeed Studio XIAO ESP32S3 in Arduino**

The base Arduino installation will not recognize the ESP32S3 board, so you will need to update the boards the Arduino IDE recognizes by following the guide from Seeed Studio here: <https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/>

* Scroll down to the “Software Preparation” section and follow the instructions.
* Once you have followed the instructions, you should be able to upload code to the boards.

**Uploading the Code**

The code for the pipette aid and base station can be found in the [Github repository](https://github.ncsu.edu/engr-csc-sdc/2024Fall-Team33-Srougi/tree/main) in “controller\_files > code”, along with the library for the IMU used for orientation sensing (not actually used in-game). Download the Arduino files - Base\_Station.ino and Pipette\_Aid.ino - and open them in the Arduino IDE. You should be prompted with the following prompt:



*Figure 1: Prompt when downloading Arduino files*

Hit “OK” to create new folders for the code in your Arduino folder on your computer - this is required. Next, move the entire “Adafruit\_LSM6DS” folder inside the “libraries” folder in the Arduino folder on your computer, e.g. “C:\Users\xxx\Documents\Arduino\libraries”.

To upload the code to the microcontroller:

* Connect the MCU (microcontroller) to your computer via USB.
* Open the desired file - e.g.Base\_Station.ino or Pipette\_Aid.ino - in the Arduino IDE.
* Navigate to “Tools > Board > ESP32 Arduino” and scroll down to find “XIAO\_ESP32S3” (it’s very far down”), and select it.
* Navigate to “Tools > Port” and select the COM port the board is connected to, e.g. “COM12 (ESP32 Family Device)”
* Hit the check mark in the top left corner to compile and verify that the code is correct, then hit the arrow next to the check mark to upload the code to the MCU. Upload progress can be watched in the output window at the bottom of the IDE.

*Figure 2: Checkmark to Upload Code to the MCU*



*Figure 3: Upload progress*

* The code is now uploaded to the board and automatically begins running.

**Testing**

**Running Tests**

1. **Unit Tests**
   1. Open the Test Runner via `Window > General > Test Runner`.
   2. Run all unit tests in the "PlayMode" and "EditMode" categories.
      1. Unit
2. **Manual Testing**
   1. Use the Meta Quest 3 headset to manually test interactions, including:
      1. Movement and teleportation
      2. Object interactions
      3. UI functionalities like the Lab Journal and Angle Alerts

**Directory Structure**

* **Assets/Scripts:** Contains all scripts, including:
  + **`HandInteractionManager.cs`:** Manages object interactions for both hands.
    - Specifically contains logic for detaching serologicals, deleting used serologicals
  + **`HoverLabelUIt.cs`:** Displays hover labels and highlights objects.
  + **`AngleAlertNotificationUI.cs`:** Displays alerts when pipetting angles are incorrect.
  + **`PipetteAid.cs`(Contains class SerologicalAttachment):** Manages logic specific to the pipette aid and acts to control access to which hand’s buttons are used for aspiration/dispense, manages attaching serologicals to the pipette aid (Thus its name)
  + `**Bottle.cs`:** Handles adjusting and turning off and on the mesh renderers for the bottle liquids, Also handles the math part of the drawing and dispensing of liquid and detecting if a serological is in a bottle.
  + `**Liquid.cs`:** handles detection of a serological in the bottles liquid specifically
  + `**Task.cs`:** Handles generating updating and scoring a task
  + **`LabJournalUI.cs`**: Manages the VR Lab Journal interface, including task, rules, and feedback pages.
  + ‘**Serologicals.cs’:**  Contains logic for turning on and off the mesh renders for the liquid in the serologicals and the math for adjusting the serological liquid mesh’s correctly.
* **Assets/Prefabs:** Includes reusable 3D models and UI prefabs.
* **Assets/Scenes:** Contains Unity scenes, such as the main VR Lab.
* **Assets/Materials:** Contains materials and textures for objects.
* **Assets/Resources:** Stores configuration files for dynamic loading.

**Key Components and Extensibility**

**Adding New Interactable Objects**

1. **Create Object Prefab**
   1. Import or create a 3D model in Unity
   2. Drag the model into the scene to configure it
   3. Add the `XRGrabInteractable` component to the GameObject to enable interaction.
   4. Adjust the component settings, such as interaction layer masks or movement types, based on the behavior you want.
2. **Set or Add Valid Tags**
   1. If the object requires a new tag:
      1. Go to `Edit > Project Settings > Tags and Layers`.
      2. Under the `Tags` section, add a new tag.
      3. Check that the object tag matches the `validTags` list in `HoverLabelAndHighlight`.
   2. Update the list of valid tags:
      1. Open the `HoverLabelAndHighlight.cs` script.
      2. Locate the `validTags` list
      3. Add your new tag to the list
3. **Add Hover Functionality**
   1. Attach the `HoverLabelAndHighlight` script to the GameObject to enable hover label functionality.

**Adding New UI Elements**

1. **Create New UI**
   1. Use Unity's Canvas system in "World Space."
   2. Position the canvas relative to the VR environment.
2. **Link to Scripts**
   1. Update scripts like `HandInteractionManager` or `AngleAlertNotificationUI` to integrate new UI functionalities.
3. **Testing UI Integration**
   1. Use Unity’s VR simulator to test new UI elements without deploying to the headset.

**API Reference and Key Scripts**

**AngleAlertNotificationUI.cs**

* **Purpose:** Displays angle alerts when the pipette is held incorrectly during aspirating/dispensing.
* **Extending:** Modify `ShowAngleAlert` to display additional angle data or adjust notification placement.

**HandInteractionManager.cs**

* **Purpose:** Tracks objects held by each hand and manages interactions like picking up/dropping objects.
* **Extending:** Add new interaction logic under `UpdateHeldObjectStatus`.