Contents

1 Local Anesthesia Haptics

1.1 Overview

This document provides comprehensive technical documentation for the Local Anesthesia VR Simulation with Haptics project. Developed by the NYU College of Dentistry, this VR simulation has been enhanced through a collaboration with the Applied Interactive Multimedia Laboratory to integrate haptic feedback.

1.2 Video Demo

Watch the full simulation on YouTube at https://www.youtube.com/watch?v=Ga___IgOu5bE.

1.3 Controls

- **Y**: Calibrate syringe.

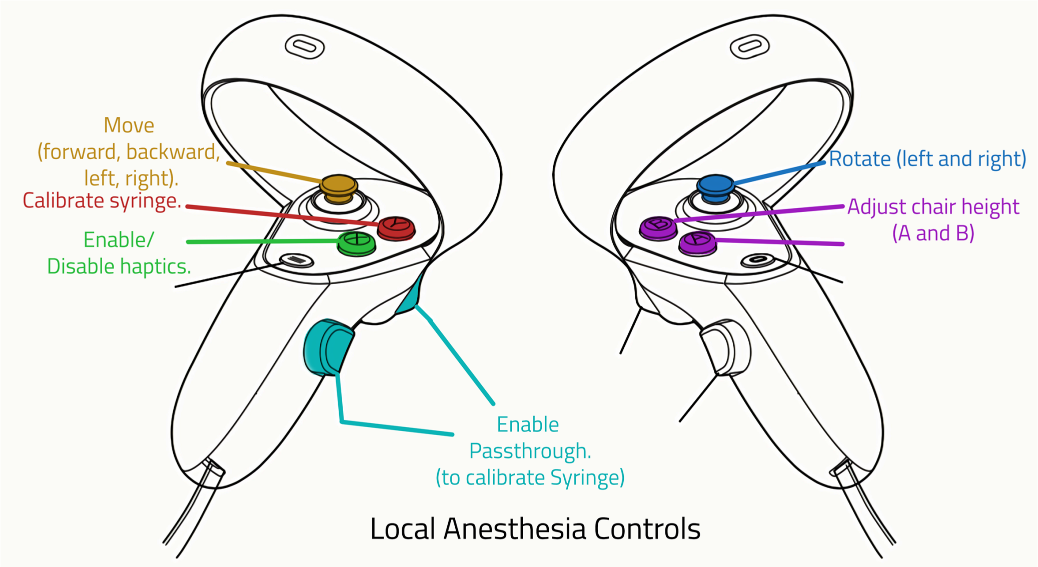


Figure 1: Controller Keys for VR Simulation

- **X**: Enable/Disable haptics.
- **A and B**: Adjust chair height.
- **Left Joystick**: Move (forward, backward, left, right).
- **Right Joystick**: Rotate (left and right).
- **Left Grip Trigger**: Enable Passthrough.
- **Touch the Syringe**: Disable Passthrough.

1.4 Project Contents

- **Google Drive Link for Project Files**: https://drive.google.com/drive/folders/1DM7zRw2hjEjneNvTwJHMj4V-7whAdX9G?usp=sharing
  - `Demo-bundle.zip`: Contains demos for device calibration.
  - `HapleyTest.zip`: Main Unity simulation project.
  - `HaplyInverseComponentsInstaller-2.0.0.exe`: Device drivers for Haply hardware.
  - `HaplyInverseComponentsInstaller-2.1.1.exe`: Device drivers for Haply hardware.
  - `serialmanager.py`: Python script for auto-detecting syringe modules.

1.5 Setting Up for the First Time

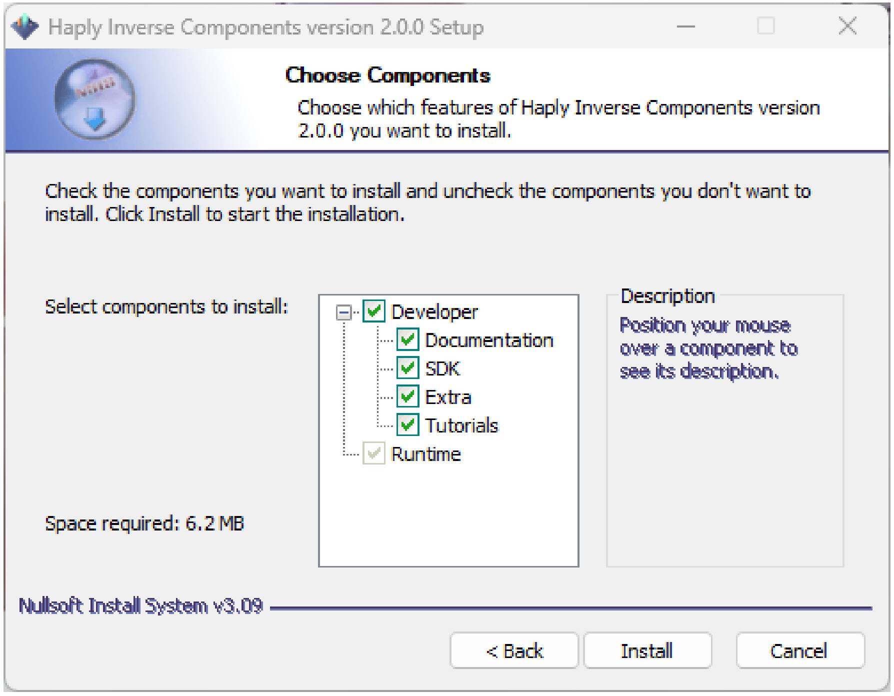
1.5.1 Hardware Requirements

- Oculus Quest 3 VR Headset (connected to a computer via Quest Link)
- Hand controllers
- High-performance computer with a capable graphics card
- Haply Inverse 3 Haptics hardware
- Custom-made syringe modules

1.5.2 Software Setup

1.5.2.1 Initial Setup

1. **Unity Installation**:
   - Install Unity version 2022.3.33f1 from the Unity Archives Unity Archive
2. **Driver Installation**:
   - Install `HaplyInverseComponentsInstaller-2.1.1.exe`. If incompatible, fallback to version `2.0.0`. When installing the Inverse Components, we have to check everything.



Haply Inverse Components version 2.0.0 Setup

Choose Components

Choose which features of Haply Inverse Components version 2.0.0 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.

Select components to install:

- ☑ Developer
  - ☑ Documentation
  - ☑ SDK
  - ☑ Extra
  - ☑ Tutorials
- ☑ Runtime

Space required: 6.2 MB

Description

Position your mouse over a component to see its description.

Nullsoft Install System v3.09

3. **Project Setup**:
   - Unzip `Demo-bundle.zip` and `HapleyTest.zip`.
   - Open the unzipped Unity project in Unity Editor.

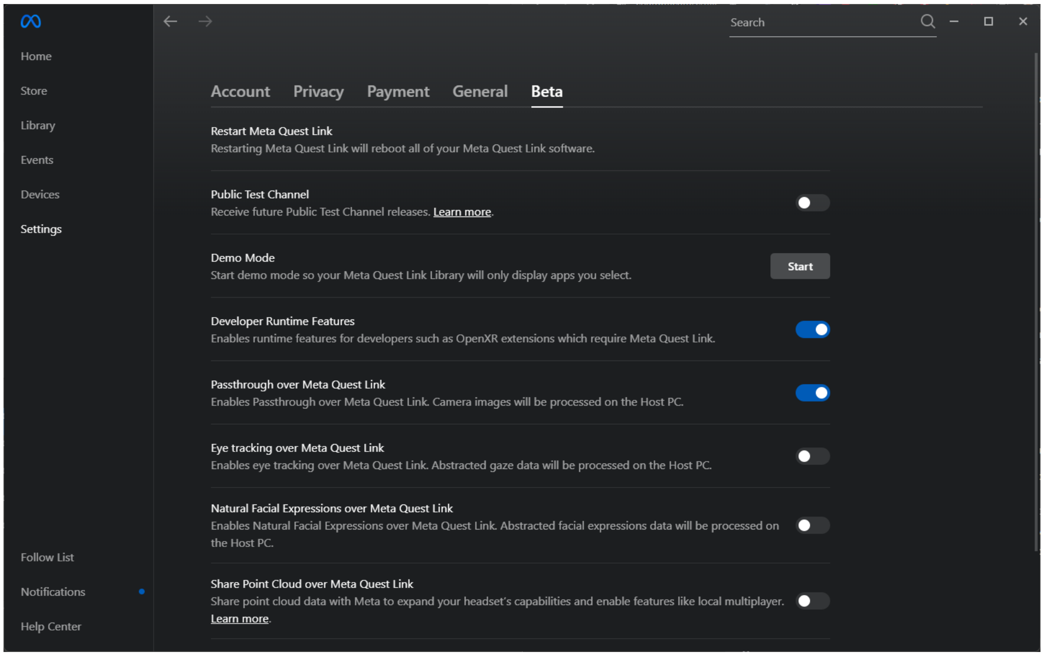
1.5.2.2 Python Environment Setup

- Ensure Python is installed and configure the environment to include `pyserial`. Use the following command to install `pyserial`:

```
pip install pyserial
```

1.5.2.3 Enable Passthrough

Note that in the Meta Quest Link Application, we need to explicitly allow the Passthrough option.



Restart Meta Quest Link
Restarting Meta Quest Link will reboot all of your Meta Quest Link software.

Public Test Channel ○
Receive future Public Test Channel releases. Learn more.

Demo Mode
Start demo mode so your Meta Quest Link Library will only display apps you select.

Developer Runtime Features ●
Enables runtime features for developers such as OpenXR extensions which require Meta Quest Link.

Passthrough over Meta Quest Link ●
Enables Passthrough over Meta Quest Link. Camera images will be processed on the Host PC.

Eye tracking over Meta Quest Link ○
Enables eye tracking over Meta Quest Link. Abstracted gaze data will be processed on the Host PC.

Natural Facial Expressions over Meta Quest Link ○
Enables Natural Facial Expressions over Meta Quest Link. Abstracted facial expressions data will be processed on the Host PC.

Share Point Cloud over Meta Quest Link ○
Share point cloud data with Meta to expand your headset's capabilities and enable features like local multiplayer. Learn more.

Figure 2: Need to enable Passthrough in Meta Quest Link

Once these are set up, we should be able to run smoothly.

1.6 Running Procedure

Once initial software is installed, you can run the simulation as follows for day to day usage.

1.6.1 Initial Calibration

Before using the simulation, it is essential to calibrate the Haply Inverse 3 device: 1. Navigate to the **Demo-bundle** folder and open the launcher application. 2. Select the **Device Dashboard** app to begin the calibration process for the Haply Inverse 3. 3. Choose your device port from the device list. 4. Select **"Shaded"** to view the device in the dashboard. 5. Manipulate the device arm to ensure it is aligned properly. 6. If the Haply Inverse 3 is not calibrated, refer to the Haply Inverse 3 manual to place the device arm in the calibration position by attaching the magnet to the top of the device and then press the **Calibrate** buttons.

1.6.2 Serial Manager Setup

Next, set up the serial manager to detect syringe modules: 1. Ensure Python and `pyserial` are installed on your system. If not, install `pyserial` using pip:

```
pip install pyserial
```

2. Launch the serial manager script through the command prompt:

```
python serialmanager.py
```

3. Leave this command prompt window open as it will continue to detect the syringe modules automatically.

1.6.3 Running the Simulation

- Put on the VR headset.
- Click the **Play** button in Unity to launch the simulation.
- Click **Scan** in the Python Serial Manager window to automatically connect the syringe module to Unity.

1.6.4 Using the VR Controls

Below are all the controls for the simulation.

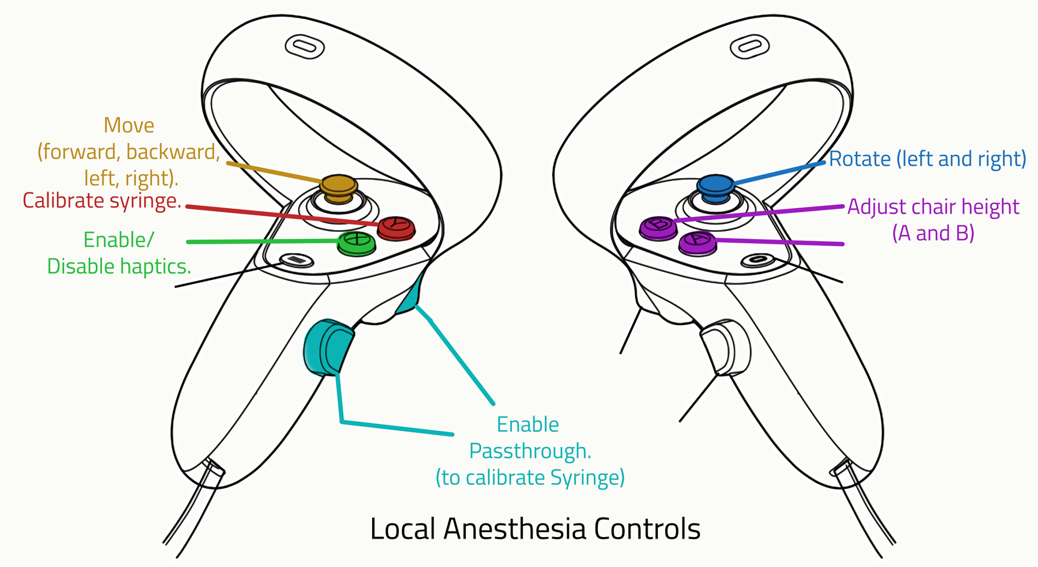


Figure 3: Controller Keys for this simulation

- **Y**: Calibrate syringe.
- **X**: Enable/Disable haptics.
- **A and B**: Adjust chair height.
- **Left Joystick**: Move (forward, backward, left, right).
- **Right Joystick**: Rotate (left and right).
- **Left Grip Trigger**: Enable Passthrough.
- **Touch the Syringe**: Disable Passthrough.

1.6.4.1 Control Overview

Pick up the VR controllers. Use the following controls: - **Left Joystick**: Move (forward, backward, left, right). - **Right Joystick**: Rotate (left and right). - **A and B buttons**: Adjust chair height.

1.6.4.2 Calibration and Interaction

- Press the **Left Grip Trigger** to enable passthrough for calibration

of the virtual syringe with the real-world syringe, adjusting both its position and rotation.
- Use the **Y**, **A**, **B** buttons and the joysticks to approximately align the virtual syringe with the real-world syringe.
- Rotate the syringe to any orientation and use the plunger. Ensure the **Scan** button on the Python Serial Manager has been clicked to sync the syringe rotation data.
- Once aligned to your satisfaction, drop the controllers and manually handle the syringe, which will revert to the VR simulation with the correct alignment.

1.6.5 Enabling Haptic Feedback

1.6.5.1 Haptic Control

- To enable haptic feedback, pick up the left controller with your free hand and press the **X** button. This will toggle the haptic feedback allowing you to feel different textures such as skin and teeth.
- If the syringe goes out of alignment, press the **Y** button to recalibrate.

Note: All buttons to press are on hand controllers, not on keyboard.

1.7 Project Structure

Currently, when we open the Unity Project, we can see a well organized folder structure.

The simulation design is modular. All the scripts are in the Scripts Folder and is heavily documented. Below are a summary of all the scripts used.

- `updateMeshCollider_New`
  - **Purpose**: Dynamically updates the mesh collider
  - **Functionality Description**: Syncs mesh collider with blend shape deformations to accurately reflect the mesh's current visual state.
- `VRLocationAdjuster`
  - **Purpose**: Controls Player movement using Oculus inputs
  - **Functionality Description**: Facilitates the syringe calibration VR space.
- `RemoveCollidersEditor`

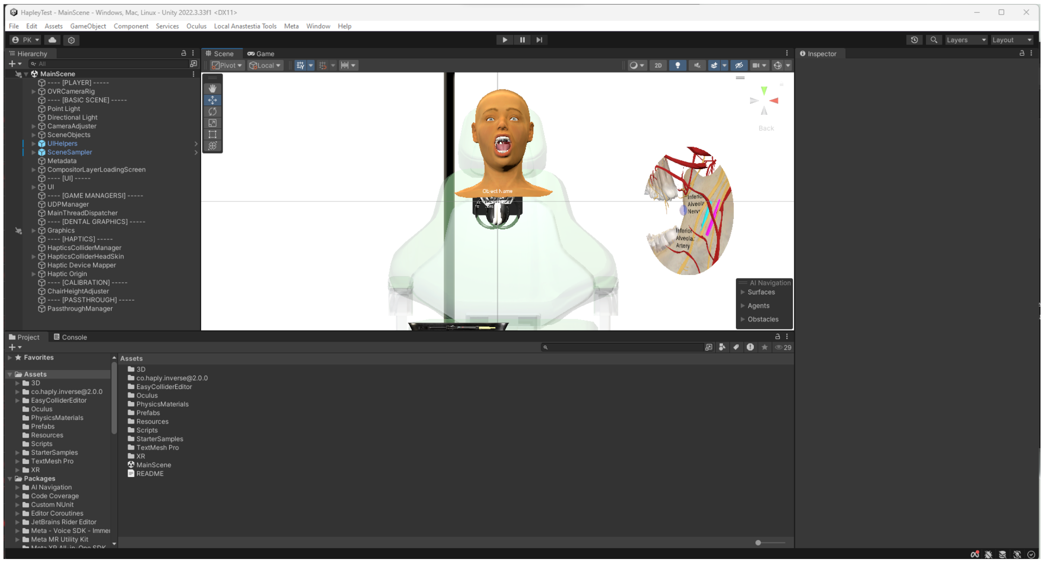


Figure 4: Organized Unity Project

  - **Purpose**: Provides a tool for removing colliders
  - **Functionality Description**: Adds a tool in the Unity Editor for managing collider components during development, useful in haptic feedback adjustments.
- `PersistentTrail`
  - **Purpose**: Visualizes the trajectory of the Haply Device
  - **Functionality Description**: Enhances debugging by displaying and saving the motion trajectory of Haply arm.
- `ChairHeightAdjuster`
  - **Purpose**: Adjusts height of Player in VR World
  - **Functionality Description**: Allows ergonomic alignment adjustments within the VR setting on how high the OVRCamera component is.
- `AssignPhysicsMaterialEditor`
  - **Purpose**: Assigns physics materials to colliders
  - **Functionality Description**: Automates the setup of Hard and Soft physical materials for haptic integration.
- `PassThroughController`
  - **Purpose**: Manages Oculus Passthrough feature
  - **Functionality Description**: Toggles passthrough visibility dy-

namically, used for syringe calibration.

- `MainThreadDispatcher`
  - **Purpose**: Ensures thread-safe execution on main thread
  - **Functionality Description**: Maintains stability in networked or complex simulations by safely managing actions affecting game objects. This is required for syringe to Unity communication.
- `CombinedController`
  - **Purpose**: Processes sensor data for syringe control
  - **Functionality Description**: Syncs the orientation of the virtual syringe with its physical counterpart.
- `LaserPosition`
  - **Purpose**: Manages a VR laser pointer
  - **Functionality Description**: Provides a visual tool for pointing and selection, enhancing interaction within the VR environment.
- `UDPSerialReceiver`
  - **Purpose**: Handles UDP data reception
  - **Functionality Description**: Connects Unity with external haptic devices (syringe, cheek retraction module) for real-time interaction control.
- `FaceControl`
  - **Purpose**: Animates facial expressions
  - **Functionality Description**: Adjusts facial blend shapes based on inputs, namely the cheek retraction module.
- `DrawLine`
  - **Purpose**: Draws a line between two points
  - **Functionality Description**: Used for visualizations like indicating paths or connections for debugging purposes.
- `HapticsToggleManager`
  - **Purpose**: Manages haptic feedback colliders
  - **Functionality Description**: It goes through all the Haptic enabled game objects and toggles the mesh colliders.

1.8 Implementing Haptics

Implementing Haptics is straightforward. Attach any Unity Collider component (Box, Sphere or Mesh) to any game Object and assign a Physics Material (either Soft or Hard). If the mesh is complex, we can use the Easy Collider Editor plugin (already installed in the project).

IMPORTANT: We have to assign either “Soft” or “Hard” Physics material to the colliders, or the device will not apply any force. This is a safety mechanism.

If there are too many colliders to assign the physics materials to, we have created some tools to help you automate the process.

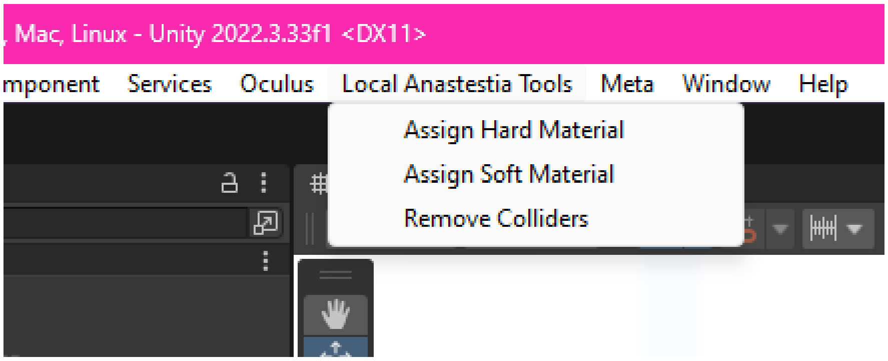
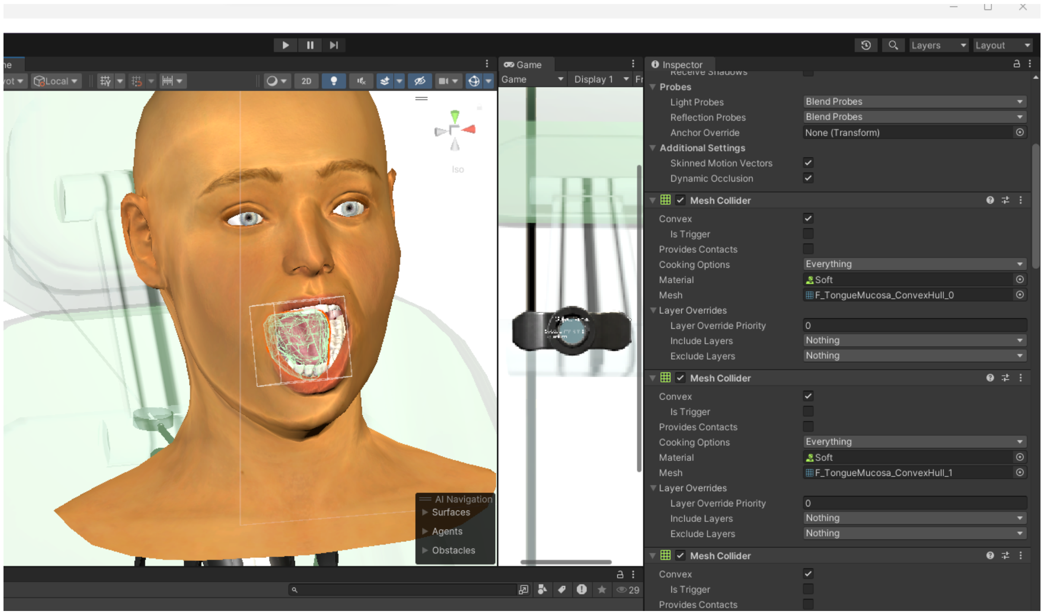


Figure 5: Custom Tools for Local Anesthesia

A correctly working haptics implementation should look like this.

1.9 Contact

If you have any questions or comments, please email Pi Ko at pi.ko@nyu.edu for this simulation. Thank you for reading.



Probes
Light Probes: Blend Probes
Reflection Probes: Blend Probes
Anchor Override: None (Transform)
Additional Settings
Skinned Motion Vectors ☑
Dynamic Occlusion ☑

Mesh Collider
Convex ☑
Is Trigger □
Provides Contacts □
Cooking Options: Everything
Material: Soft
Mesh: F_TongueMucosa_ConvexHull_0
Layer Overrides
Layer Override Priority: 0
Include Layers: Nothing
Exclude Layers: Nothing

Mesh Collider
Convex ☑
Is Trigger □
Provides Contacts □
Cooking Options: Everything
Material: Soft
Mesh: F_TongueMucosa_ConvexHull_1
Layer Overrides
Layer Override Priority: 0
Include Layers: Nothing
Exclude Layers: Nothing

Mesh Collider
Convex ☑
Is Trigger □
Provides Contacts □

Figure 6: Correct Haptics Implementation Example