README & User Manual
SCE VR Training Program

Brailey Gonzalez-Oxlaj, Derek Rosales ,Kevin Guerrero

December 10, 2025

1 Project Overview

1.1 Formal Objective

The SCE VR Training Program is a virtual reality training application designed for Southern California Edison (SCE). The objective of the project is to provide an immersive, step-by-step training experience that simulates real-world overhead maintenance scenarios on electric poles. Trainees can safely practice installation and maintenance procedures for Power Delivery Products (PDP) such as smart navigators and related equipment without risking injury or equipment damage.

1.2 Why This Project Is Needed

SCE frequently works with both internal field workers and third-party contractors. Contractors often do not receive the same depth of training as full-time employees. Traditional training methods can be expensive, constrained by physical resources, and potentially hazardous.

The VR Training Program:

- Provides a safe, controlled training environment.
- Closely mimics real-world tools, equipment, and procedures.
- Reduces costs associated with in-person training and equipment wear.
- Increases trainee confidence and familiarity before performing field work.
- Helps standardize training across internal staff and contractors.

1.3 Key Features

- Immersive VR simulation using Meta Quest 2.
- Guided, step-by-step training flow for overhead installation tasks.
- In-world UI with “Next”, “Previous”, and audio playback controls.
- Inventory system that allows users to carry tools between scenes.
- Audio feedback and 3D sound for immersion.
- Haptic feedback when interacting with correct objects.
- LED sequence simulation for Pole Master device states.

2 Repository and Project Management

2.1 Git Repository

The project source code and assets are hosted in a Git repository:

`https://github.com/bgonza214/CS3338-Group9`

Clone the repository using:

```
git clone https://github.com/bgonza214/CS3338-Group9.git
```

2.2 Jira Board

Project tasks, sprints, and user stories are tracked on Jira:

`https://calstatela-team-group9.atlassian.net/jira/software/projects/SCRUM/boards/1`

Refer to the Jira board for current sprint status, open issues, and backlog items.

3 System Requirements

3.1 Hardware Requirements

- **VR Headset:** Meta Quest 2.
- **Controllers:** Meta Quest 2 left and right controllers.
- **Development PC (for building and deploying):**
  - 8 GB RAM or higher.
  - Processor: Intel i5-4590 / AMD Ryzen 5 1500X or greater.
  - At least 1x USB port.
  - Stable internet connection (for package downloads and deployment).

3.2 Software Requirements

- Windows 10 or higher (Version 22H2 or later recommended).
- Unity Editor (2021.3.x LTS; originally 2021.3.5f1).
- Unity Hub.
- Meta Quest OS version 46.0 or higher on the headset.
- Android build support installed in Unity (including SDK, NDK, and OpenJDK).
- Visual Studio 2019 or later with C# support.

4 Downloading and Building the Project

4.1 Cloning the Repository

1. Open a terminal or command prompt.
2. Navigate to the directory where you want to store the project.
3. Run:

```
git clone https://github.com/bgonza214/CS3338-Group9.git
```

4. Open Unity Hub and add the cloned project folder as an existing project.

4.2 Unity Project Setup

1. Open the project in Unity using the correct Unity version (e.g., 2021.3.5f1).
2. Ensure the following packages are installed or enabled:
   - XR Plugin Management.
   - Oculus / OpenXR support for Meta Quest 2.
   - XR Interaction Toolkit.
3. Configure build settings:
   (a) Go to `File` $\rightarrow$ `Build Settings`.
   (b) Select **Android** as the platform and click **Switch Platform**.
   (c) Add the current scene(s) to the build by clicking **Add Open Scenes**.
4. Configure XR Plugin:
   (a) In `Project Settings` $\rightarrow$ `XR Plugin Management`, enable the Oculus or OpenXR plugin for Android.
5. Configure graphics and API level:
   (a) In `Project Settings` $\rightarrow$ `Player` $\rightarrow$ `Other Settings`:
      - Set minimum API level to at least **Android 6.0 (API level 23)**.
      - Adjust Graphics APIs order (e.g., Vulkan / OpenGLES3) as required for Quest.

4.3 Building the APK for Meta Quest 2

1. Connect the Meta Quest 2 to the PC via USB and enable developer mode.
2. In Unity, go to `File` $\rightarrow$ `Build Settings`.
3. Ensure the correct scenes are included.
4. Click **Build** or **Build and Run** and select an output folder.
5. Install the generated APK on the Meta Quest 2:
   - Using `adb install` from the Android SDK, or
   - Via SideQuest or other deployment tools.

5 Running the Training Simulation

5.1 Launching the Application

1. Put on the Meta Quest 2 headset.
2. Locate the installed training application in your app library.
3. Launch the application.

5.2 Main Menu and Navigation

On startup, the user is greeted with a simple UI explaining the training simulation.

Typical options include:

- **Next**: Proceed to the next area of training.
- **Play Audio**: Play a voice-over for the text instructions currently displayed.
- **Previous**: Go back to the previous training area (available after the first step).
- **Menu** / **Wrist Menu**: Bring up options such as restarting the simulation, jumping to specific scenes, or viewing remaining tasks.

5.3 Controller Basics

The simulation assumes the user is familiar with basic Meta Quest 2 controller usage:

- **Grab**: Hold and release objects.
- **Trigger** / **Select**: Interact with UI buttons and confirm actions.
- **Primary** / **Secondary Buttons**: May be mapped to actions such as opening the menu.
- **Joystick**: Used for snap turning or menu navigation if configured.

6 Using the Training Features

6.1 Step-by-Step Training UI

During training:

- Instructions for the current step are displayed on an in-world UI panel.
- The user completes the step following the on-screen description and any audio prompts.
- When the system detects that the step is complete, the **Next** button becomes available.
- The **Previous** button allows revisiting earlier steps when appropriate.

6.2 Inventory System

The Inventory System Module (ISM) allows the user to carry important tools between scenes:

- Each inventory slot holds one item and displays a label with the item name.
- If an item is dropped outside the allowed boundary, it is returned to a “dropped item” slot.
- Items can be grabbed from and placed into the inventory using the VR controllers.

6.3 Audio and Feedback

- Sound effects and voice-overs provide guidance and immersion.
- A dedicated audio toggle button allows turning audio on or off in the UI.
- Haptic feedback in the controllers indicates successful interactions (e.g., picking up the correct tool).

6.4 Pole Master LED Sequence Simulation

A dedicated module simulates the LED states on the Pole Master:

- Different LEDs (WAN, ERR, I/O, DC) display different colors and blink patterns depending on system state (communication mode, error mode, pairing mode, connected mode, power modes, etc.).
- Certain user actions (e.g., using a magnet tool, completing installation steps) trigger changes in the LED sequences.

7 Troubleshooting

7.1 Common Issues

The application does not appear on the headset.

- Confirm the APK installed successfully on the Meta Quest 2.
- Make sure developer mode is enabled when deploying.

The simulation runs slowly or stutters.

- Close other intensive applications on the headset.
- Ensure you are using the correct build target and graphics settings.
- Check that you are using supported API levels and XR plugins.

Controllers do not interact with UI or objects.

- Verify that XR Interaction Toolkit components are properly attached to controllers.
- Ensure that interaction layers and physics colliders are configured correctly.

8 Contributing and Development Workflow

1. Create a feature branch from the main branch.
2. Implement your feature in Unity and commit changes regularly.
3. Push your branch to the remote repository.
4. Open a pull request and link it to the relevant Jira ticket.
5. Request code review from teammates.
6. Once approved, merge into the main branch.

9 Credits and Contact

This project was developed by:

- Derek Rosales
- Brailey Gonzalez-Oxlaj
- Kevin Guerrero

For questions or support, please contact the project team or the course instructor.