Software Design Document

for

SCE VR Training Program

Version 2.0

Prepared by Cameron Cheng, Fernando Torres, Denise Tabilas, Martin Castorena, Manuel Guillen, Jaiden Holcomb, Seng Hei Lei, Matthew Mendoza, Kevin Truong and Han Cao

Southern California Edison

Table Of Contents

Revision History

Name	Date	Reason For Changes	Version
Martin Castorena	11-21-2022	Work on 2.1,2.2,5.1,5.2,5.3	1.0
Fernando Torres	12-8-22	working on all of part 1	1.1
Jaiden Holcomb	12-9-2022	Implementing User Interface Section (Section 8.1 - 8.3)	1.1
Cameron Cheng	3-8-2023	work on section 5 / 6	1.2
Matthew Mendoza	5-10-2023	updated sections	1.3
Multiple members	5-12-2023	double checked all sections	2.0

1. Introduction

1.1 Purpose

The main objective of this project is to develop and implement a training application in virtual reality specifically designed for Southern California Edison. This training platform will provide trainees with an immersive experience that simulates the real-world scenarios they will encounter while performing overhead tasks, including maintenance on the electric poles. By utilizing virtual reality, trainees will be able to perfect their skills in a safe and controlled environment, without the risk of injury or equipment damage. Moreover, the use of VR technology will enable Southern California Edison to streamline their training program, minimize costs, and improve the overall efficiency and effectiveness of their workforce.

1.2 Document Conventions

This document followed the details on the documents that were provided to us by the Edison leaders as well as the knowledge we gained from the tour of the Edison facility in Chino, California in November.

1.3 Intended Audience and Reading Suggestions

This document is intended for Edison leaders/managers, workers, and trainees or outsource trainees. The document details the full overview of the application's design including the methods used in designing this application and policies. Each reader should read section by section to get a clear picture of the application.

1.4 System Overview

Unity Game Engine - This real-time development platform allows users to create programs, 2D and 3D games, physics simulations, etc. We used Unity to create the virtual reality training application using models imported from a 3D modeling software.
Blender - This program allows users to create 3D models, simulation renders, or even animations. We used Blender to create 3D models of the various equipment that will be used in the virtual reality training application.

2. Design Considerations

2.1 Assumptions and Dependencies

- User has read the VR manual
- User has obtained a Meta Quest 2
- User has updated version of the Meta Quest 2
- User knows the general controls for Meta Quest 2 controllers such as the grab, select, primary, and secondary buttons
- User knows how to read English

2.2 General Constraints

User Experience

- Utility truck bucket cannot be moved
- User is not able to move around environment on their own, instead they are able to teleport from different areas of the map depending on what step of the process they are on
- User cannot interact with environment freely, since products are manually placed
- User will be guided through text on UI

System Constraints

- This application has only been tested with the Meta Quest 2 VR headset

2.3 Goals and Guidelines

The objective of this application is to offer trainees a straightforward, step-by-step guide on the installation process. This guide is based on the real-world training conducted at the Edison training facility, as well as the training materials provided by the Edison leadership. The equipment models in the application have been designed to closely resemble their real-world counterparts, providing trainees with a sense of familiarity with the equipment they will encounter in the field.

2.3.1 Provide user-friendly and straightforward guided training application

- Increase user engagement
- Decrease chances of frustration
- Increase overall satisfaction
- Increase productivity and efficiency

2.3.2 Base program off the real-world

- Provide students with a sense of familiarity with equipment
- Help students feel more confident and prepared when performing actual tasks
- Improve retention of learning

2.4 Development Methods

Our team utilized the agile development method, an iterative and flexible approach to software development that emphasizes collaboration, adaptability, and rapid delivery. Unlike the sequential phases of the waterfall development method, agile development involves continuous planning, development, testing, and delivery in short cycles known as sprints. This approach grants greater flexibility and responsiveness to changing requirements, as well as more frequent opportunities for feedback and course correction. Agile development typically involves close collaboration between team members and relies heavily on communication to ensure that everyone is working toward a common goal.

We chose this development approach because the complexity of our project required a methodical approach. The team had little to no prior experience with VR, so we needed to ensure that everyone was comfortable with the technology and the process. To accomplish this, we assigned small tasks to each team member, leveraging the collective expertise of the team. By working collaboratively to merge features after completing individual assignments, we ensured that the development process remained adaptable to changing requirements. This approach also enabled us to identify potential bottlenecks early on and address them in a timely manner, which helped keep the project on track. Overall, our goal was to create an efficient and effective development process, and this approach allowed us to achieve that goal while providing valuable learning opportunities for the team members who were new to VR development and 3D modeling.

3. Architectural Strategies

Prior to development, we establish our communication channels to ensure efficient and effective collaboration. We use various platforms for different purposes. For example, we use Zoom and Discord for meetings and discussions, and GitHub and Google Drive for sharing program and document files. By having multiple communication channels available, we can easily adapt to different situations and keep everyone in the loop.

When designing the program, we followed a strategy of working through the installation manual in chronological order, which allowed us to concentrate on one step at a time and make steady progress. However, we also made sure to be flexible and adaptable to any changes or unforeseen issues that came up during the process.

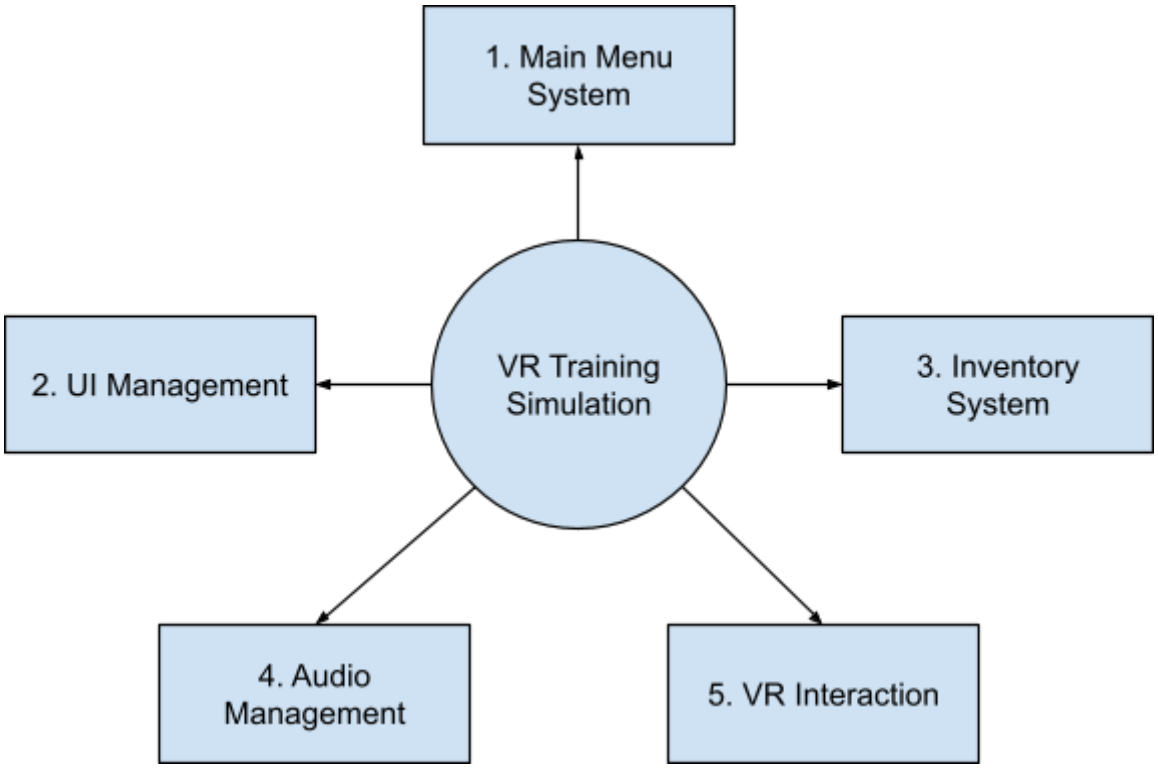
We use 3D modeling software such as Blender and Autodesk Maya while working concurrently within Unity. Due to the lack of free assets for download, we create our own custom assets when necessary.

In Unity, there is a package called 'XR Interaction Toolkit' that we utilize to manage VR controls, actions, and events. This package allows us to streamline VR development and achieve more efficient and effective results. We frequently use components from this package to ensure that our development process is as smooth and seamless as possible.

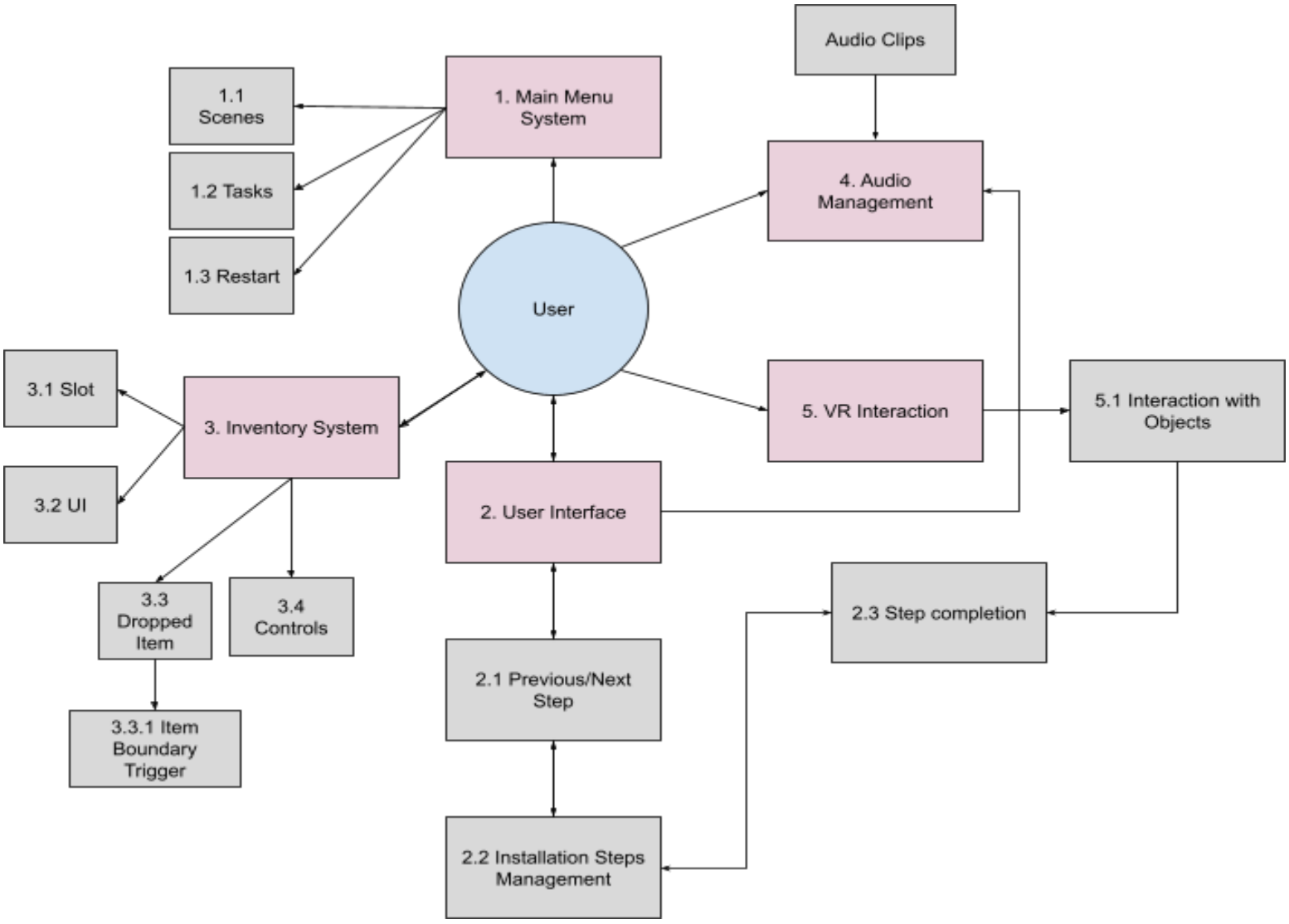
In addition to using the 'XR Interaction Toolkit', we also develop custom scripts for object components using C#. C# is a popular object-oriented language utilized in various industries, such as game development, web development, and enterprise software development. By using C#, we are able to create reusable code that can be used to expand and enhance the software, making it more robust and versatile. Overall, our approach to development and task delegation allows us to achieve a high level of quality while also being efficient and adaptable to any changes or issues that may arise.

4. System Architecture

Level 0 Data Flow Diagram



Level 1 Data Flow Diagram



5. Policies and Tactics

5.1 Choice of which specific products used

- Game Engine: Unity
- IDE: Microsoft Visual Studio 2019
- 3D Computer Graphics: Blender and Autodesk Maya
- Compiler: C# Compiler
- VR Headset: Meta Quest 2
- Unity Package: XR Interaction Toolkit

5.2 Plans for ensuring requirements traceability

Requirements were discussed at the beginning of the project. If requirements aren't clear, the team goes back to SRS documentation to see what requirements have not been met. If a requirement is not addressed in SRS, then it will be brought up during the meeting with SCE. Plans and meetings are also recorded for documentation. Requirements and design are discussed during our weekly meetings through Zoom and meeting notes are stored on Discord.

5.3 Plans for testing the software

5.3.1 Testing the features of the modules

- Sessions
  - Testing user experience as we are developing the application
  - Ensuring all objects behave and interact with each other as intended

5.3.2 Engineering trade-offs

- Sessions don't support different VR headsets
- The number of people who can try the VR experience is based on the number of available headsets.

5.3.3 Coding guidelines and conventions

- Coding Guidelines
  - Code must have comments explaining what the purpose of the code is
  - Comments at the head of a script explaining the purpose of the script

5.3.4 The protocol of one or more subsystems, modules, or subroutines

- All supporting packages, modules, and subsystems must be organized as dependencies.

5.3.5 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

- There is no set of particular algorithms or implemented.

5.3.6 Plans for maintaining the software

- Most textures and assets are used for aesthetic purposes and do not require frequent updating.
- XR plugin is maintained by Unity, thus is susceptible to becoming obsolete if Unity chooses.

- Oculus firmware is maintained by Meta, which is also a chance to become obsolete.
- Custom scripts may become depreciated based on the unity version that the methods used are from.

5.3.7 Interfaces for end-users, software, hardware, and communications

- Software Interfaces
  - Unity Editor
    - World Space UI, positioned in the world among other 2D or 3D objects.
    - In-World User Interfaces, designed to not cause discomfort or break immersion.
    - VR User Interface, designed specifically for VR applications.
  - Visual Studio 2019
    - simple and intuitive editor for custom C# script creation
    - A minimap of the code outline provides the user with a high-level view of your code, allowing for quick navigation of large files.
- Hardware Interfaces
  - QuestLink
    - Is used to be able to access Oculus developer mode and connect to the computer to view in Unity.

5.3.8 Hierarchical organization of the source code into its physical components (files and directories)

- Most original source code should be placed in the Assets folder where the Unity project file is located.
  - Scripts, audio, and models have designated folders in the Assets folder
  - Complex Assets will have deeper rooted directory path
- Scripts will be C# files

5.3.9 How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

- Installing XR Plugin
  - Go to File in a new/existing project then Build Settings
  - Click on Android in the Platform tab then press on Add Open Scenes
  - Click on Player Settings
  - Click on Other Settings tab and switch the order of Vulkan and OpenGLES3 in the Graphics APIs tab
  - Change the Minimum API Level to “Android 6.0 ‘Marshmallow’ (API level 23)”
  - On the Left tab, click on XR Plugin Management and install by clicking the only button present

- ➢ Once the initial download is finished click on the check box marked Oculus

❖ Importing Assets

- ➢ Make sure you are logged in Unity
- ➢ Click on the Window > Package > Manager located towards the top of the edition
- ➢ In the Packages tab towards the top click on “Package: My Assets”
  - ■ To add an Asset to your account you must go to the Unity Asset store and then click on the asset you want. Then click on “Add to my Asset.” This will add it to your account and make it importable in the Unity editor
- ➢ Click on the Import button based on what asset you want to import
- ➢ A new window will appear and ask what specifics you want from the asset. By default, you can import all of it. Click on Import.

6. Detailed System Design

6.1 Main Menu Module (MMM)

6.1.1 Responsibilities

This module is used to give the user different choices of how to go through the training simulation. The menu will display options that the user can select. The purpose of the menu is to provide the user with another way to go through the simulation.

6.1.2 Constraints

- Language is in English

6.1.3 Composition

- User is able to select the scene option to display the different steps that the user can jump to
  - User can use this option to go back or forward in steps for more practice and repetition
- User is able to select the ‘Tasks’ option to display a checklist of the different steps which indicate the user’s progress
- User is able to select the ‘Restart’ option to restart the training simulation to the beginning

6.1.4 Uses/Interactions

This component interacts with VR interactions that the user has on the environment. For example, the tasks option will update its checklist according to the steps that the user has completed.

6.2 UI Management Module (UIMM)

6.2.1 Responsibilities

This component is designed to provide the user with detailed and step-by-step instructions on what actions to take during their current step. By doing so, the user will be able to navigate the training simulation with ease, thus enabling them to fully engage with the learning material and achieve their desired learning outcomes. Additionally, these clear instructions not only facilitate the user's progress through the simulation but also help to ensure that the user comprehends the key concepts and skills being taught, leading to greater retention and application of the knowledge gained.

6.2.2 Constraints

This component, much like the main menu, requires the users to possess a certain level of familiarity with the English language, as well as a basic understanding of the functionalities and controls associated with the VR controllers.

6.2.3 Composition

This component is made up of a few sub-components:

- Text that describes the user's goal for the current step
- Next button to jump to the next step
  - User is required to complete the current step in order to go to the next
- Previous button to jump to the previous step, if applicable
- Audio button to toggle the audio effects

6.2.4 Uses/Interactions

Controllers play an important role in the system by interacting with other components and monitoring changes in the environment. To improve the user experience, we implemented a system that detects when a user has completed a step in the process, enabling them to transition smoothly to the next step. Moreover, controllers are designed to engage users through interactive buttons on the interface.

6.3 Inventory System Module (ISM)

6.3.1 Responsibilities

The ISM will handle the inventory functions. We decided to implement an inventory so that the user can bring objects that they need easily from scene to scene.

6.3.2 Constraints

The inventory has limited space. There are only a few items that need to be used with the inventory so we also took that to account when implementing this feature.

6.3.3 Composition

There are a few sub-components that make up the inventory.

- Slot that holds the item
- Text label that tells the user the name of the object that is populating the slot
- Dropped item slot
  - Item that is dropped outside of the boundary is transported back into this slot of the inventory

6.3.4 Uses/Interactions

This component interacts with the VR.Interaction module. The user uses the VR controllers and buttons to place and grab items to and from the inventory.

6.4 Audio Management Module (AMM)

6.4.1 Responsibilities

The AMM is used to organize audio files and queue them when triggered by a script or component. This module accomplishes both an organized and synchronized way for

audio to blend in the app. The module achieves creating user immersion and feedback with 3D sounds.

6.4.2 Constraints

Contains audio file format restrictions constrained to AIFF, WAV, MP3, and Ogg formats. Audio data is separated from the actual AudioClips in Unity 5.0 The Audio assets can be kept in memory at all times or loaded on demand.

6.4.3 Composition

AMM Subcomponents:

- ❖ Audio Manager
  - ➢ Allows you to tweak the maximum volume of all sounds playing in the scene.
- ❖ Audio Profiler
  - ➢ Supports mono, stereo and multichannel audio assets

6.4.4 Uses/Interactions

The AMM is a powerful module that can be used to manage audio triggers, play audio when needed, and keep everything organized and in line. It is used together with game objects and scene components. The interaction manager for each component is where adjustments of the audio are made. If not set right, such as 3D sound not enabled the user will only hear the audio in one direction.

6.4.5 Resources

The Oculus headset speakers output the curated sound from the AMM to immerse the user in a virtual world with 3D sound. Alternatively if the app is run on the unity program, the computer's audio would be utilized.

6.5 VR Interaction Module (UIMM)

6.5.1 Responsibilities

This module deals extensively with the user's interaction with the environment. This includes a variety of interactions with grabbable objects, such as picking them up and putting them down, as well as objects that can be pulled towards the user, pushed away from the user, and moved around in the environment. Additionally, this module covers interactions with objects that are snapped onto specific positions within the environment, such as buttons, switches, and levers, as well as the manipulation of virtual objects that mimic real-world physics. Overall, this module provides a comprehensive understanding of how users can interact with their digital environment in a way that feels intuitive and natural.

6.5.2 Constraints

The module has some constraints and restrictions that include performance optimization, hardware compatibility, user experience considerations, and development time planning. Performance for instance, app optimization is needed to ensure that it runs at a high frame rate. Compatibility with hardware that you are targeting such as the Oculus headset. Ensures that the app is comfortable and easy to use for the user. Planning your project accordingly for development is needed. User limitations, where the user can only pick up or interact with one object at a time for each hand.

6.5.3 Composition

- ❖ Interatable
  - ➢ A component that allows an object to be interacted with in VR
- ❖ Grabbable
  - ➢ A component that allows an object to be grabbed and moved around in VR.
- ❖ Teleportation
  - ➢ A component that allows the player to teleport to different locations in the virtual environment.
- ❖ UI Interactor
  - ➢ A component that allows the player to interact with U elements in VR.
- ❖ Snap Turn
  - ➢ A component that allows the player to turn in increments instead of smooth turning.

6.5.4 Uses/Interactions

Each of the components requires use of VR Interaction. For example, the UI can be interacted with by the use of rays that are casted from the controllers. Also, the user is given haptic

6.5.5 Resources

The resources that are affected by the module are Oculus controllers. User motion, controller vibration, and button response are all affected. Internal computer resources are also affected, this includes CPU, GPU, Memory, and Storage.

6.6 Pole Master LED Sequence Module (PMLSM)

6.6.1 Responsibilities

This module must accurately display the different LED sequences on the four LEDs of the Pole Master. These LEDs are referred to as WAN, ERR, I/O, and DC.

6.6.2 Constraints

The constraints of this module are that all LEDs must be in certain states depending on the state of the virtual training simulation such as pairing mode for the I/O LED.

6.6.3 Composition

- Communication Mode (WAN LED)
  - LED must have a constant blue color
- Error Mode (ERR LED)
  - LED must have a red flash once per second
- Pairing Mode (I/O LED)
  - LED must have a yellow flash twice per second
- Connected Mode (I/O LED)
  - LED must have a yellow flash once per second
- Stable Power Mode (DC LED)
  - LED must have a constant green color
- Backup Battery Mode (DC LED)
  - LED must have a green flash once per second

6.6.4 Uses/Interactions

The user may invoke certain LED sequences such as pairing mode on the I/O LED by placing a magnet tool on the pole master. The user may also complete smart navigator installation to enter connected mode on the I/O LED. All objects contain scripts which communicate with the LEDs and in turn adjust their LED sequence based on these different states.

6.5.5 Resources

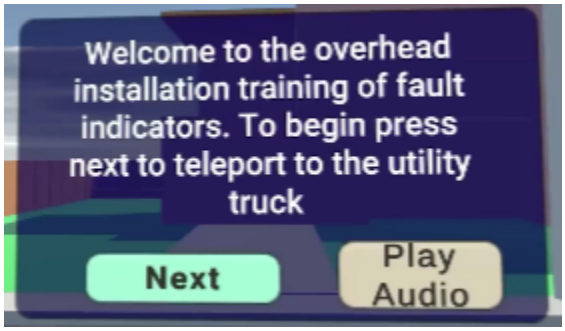
Resources affected by this module are the smart navigators as well as the electric pole on which the pole master is mounted. The magnet tool also contains scripts which enable pairing mode on the pole master.

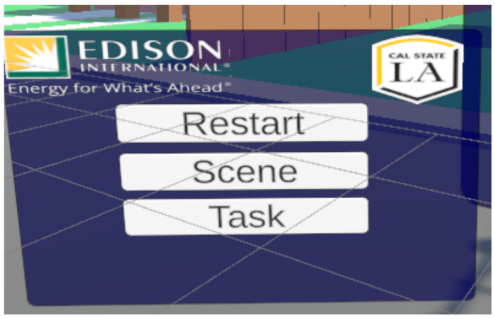
7. User Interface for Training Guidance

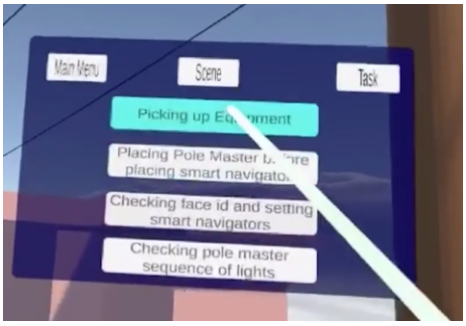
7.1 Overview of User Interface

The idea behind our user interface in our application was to keep it as simple as possible for the user. To ensure this upon loading the application, the user is greeted with a simple UI explaining the training simulation. There are 2 buttons that are displayed, the “next” and “play audio” button which takes the user to the next area of training and plays the audio version of the text that is displayed on screen. The following UI windows in the rest of the application have similar buttons but also add in a “previous” button to take the user to the last training area they were in. The user is also able to pull up a simple menu user interface which shows the options of selecting to restart the application, choose to go to different scenes or check their remaining tasks they need to complete.

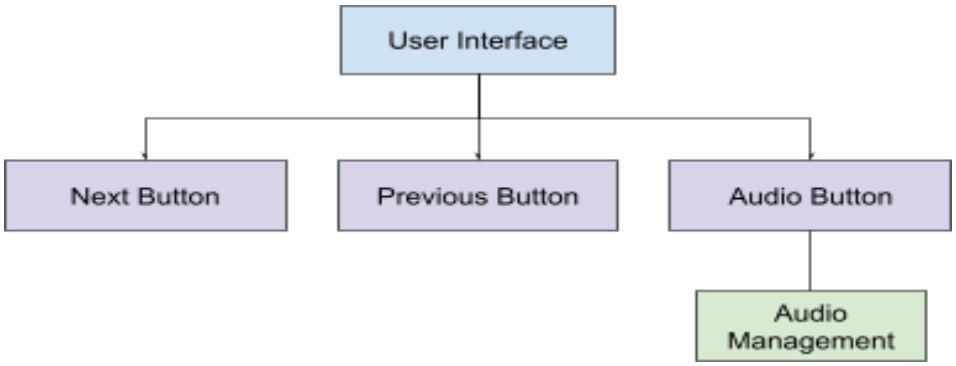
7.2 Screen Frameworks or Images







7.3 User Interface Flow Model



8. Requirements Validation and Verification

8.1 Functional Requirements

Functional Requirements	**Component Modules/UI Elements That Satisfy Requirements**
System shall run smoothly on hardware used	Component modules and UI elements will be light enough for hardware to handle
System should be free of system-breaking errors/bugs	Component modules and UI elements will work
System shall have sound effects and voice over	Sound component
System shall have options menu	Options UI element
System shall allow player to interact with objects in environment	Object module and components to allow interacting
System shall allow user to adjust volume	Volume component

9. Glossary

Blender - A modeling program used to create 3d models, simulations, and even animations.

C# (C Sharp) - A high-level programming language created by Microsoft that is utilized in many programs such as Unity.

Discord - A chatting program similar to Skype that allows users to chat with one another whether in servers or privately through direct messages. Also allows screen sharing with each other.

Github - A website that allows users to contribute to open source projects or create their own open source projects in which others can contribute to.

Google Drive - A cloud drive created by Google that allows users to store files and share them with others where those users can either view the file or even edit the file.

Meta Quest 2 VR headset - A virtual reality headset developed by Facebook that allows the users to run virtual reality programs such as games and simulations.

Southern California Edison (SCE) - The largest subsidiary of Edison International and the primary electricity utility for much of Southern California.

Unity - A program that utilizes assets and models to create 2d/3d programs like simulations and games.

Virtual Reality (VR) - A simulation of 3D objects or environments that can be interacted with by the user as if that user is there in the environment using special equipment like a virtual reality headset.

Zoom - A program that allows users to connect over video and voice chat with one another. This program is also used by schools world wide.

10. References

Bliz Studio. (2022c, September 26). Unity VR XR Interaction Toolkit Grabbable UI Canvas Tablet [Video]. YouTube. https://www.youtube.com/watch?v=TCixKyOGTRU

RealaryVR. (2022, February 17). Unity VR Development for Oculus Quest: Inventory [Video]. YouTube. https://www.youtube.com/watch?v=gAz_SeDUQBk

Justin P Barnett. (2021, December 15). How to use VR Sockets (and XR Interaction Toolkit) in Unity 2021.2 [Video]. YouTube. https://www.youtube.com/watch?v=rRNvq09Itdw

Unity. (2013, April 8). Colliders as Triggers - Unity Official Tutorials [Video]. YouTube. https://www.youtube.com/watch?v=m0fjrQkaES4

Valem. (2020, April 8). Introduction to VR in Unity - PART 1 : VR SETUP [Video]. YouTube. https://www.youtube.com/watch?v=gGYtahQjmWQ

Bliz Studio. (2022, August 23). Unity VR XR Interaction Toolkit - XR Socket Interactor [Video]. YouTube. https://www.youtube.com/watch?v=1viTbR_hTuQ

Bliz Studio. (2022b, September 14). Unity VR XR Interaction Toolkit - UI Menu and Interactions [Video]. YouTube. https://www.youtube.com/watch?v=Qb_8POkFwlk

Jayanam. (2018, May 5). Unity Beginner Tutorial : Triggers [Video]. YouTube. https://www.youtube.com/watch?v=JHZN_vMqqr4

2KMIXEDIT. (2017, November 23). Unity 5 - Showing Text on Trigger [Video]. YouTube. https://www.youtube.com/watch?v=CNNeD9oT4DY

Nima Jamalian. (2022, March 1). Unity VR Development Tutorial - Snap Zone/Socket Interaction - Part 14 [Video]. YouTube. https://www.youtube.com/watch?v=0TWVWuJi2LI

Ryan King Art. (2021, May 31). How to Use Curves in Blender (Tutorial) [Video]. YouTube. https://www.youtube.com/watch?v=Ve9h7-E8EuM

Fist Full of Shrimp. (2022, May 23). Unity VR Game Basics - PART 11 - Wrist Menu [Video]. YouTube. https://www.youtube.com/watch?v=YISa0PvQTGk

Valem Tutorials. (2022, July 24). How to Make a VR Game in Unity 2022 - PART 2 - INPUT and HAND PRESENCE [Video]. YouTube. https://www.youtube.com/watch?v=8PCNNro7Rt0

Valem Tutorials. (2022a, July 17). How to Make a VR Game in Unity - PART 1 [Video]. YouTube. https://www.youtube.com/watch?v=HhtTtvBF5bI