VR BioTech Minigame

Final Project Report

Requirements, Design, Implementation & Testing

Dr. Melissa Srougi

CSC 492 Team 33

Sam Gerstner, Satwika Kancharla, Jonathan Kolesar, Raven Midgett, Kayla Sanderson

North Carolina State University

Department of Computer Science

December 6, 2024

# Executive Summary

#### *Author(s):* Kayla Sanderson, Jonathan Kolesar

*Reviewer(s)/Editor(s):* Sam Gerstner, Raven Midgett

Dr. Srougi, an associate teaching professor at NC State’s Department of Molecular Biomedical Sciences, is the sponsor for this senior design project designed to enhance lab training and accessibility for students. Students typically only have access to labs during course time, once a week or less, and they often lack opportunities to practice critical skills like pipetting in a low-stakes environment. Dr. Srougi, known for her passion for inquiry-based learning and research collaboration with undergraduates, sees the potential of virtual reality (VR) simulations to provide students with a safe, flexible environment to refine these skills outside traditional lab hours. The games focus on foundational lab equipment skills, namely how to use a pipette aid and a micropipette.

The team was tasked with developing a VR simulation focused on foundation lab skills, particularly the use of a pipette aid, with the goal of allowing students to explore lab techniques, make decisions, and learn from mistakes without the risks or constraints associated with real-world equipment. The simulation provides an open-ended, fail-forward, learning experience, complete with timely feedback to help students recognize and correct errors, all within a realistic lab environment.

To date, significant progress has been made:

* Requirements: Detailed requirements were outlined for the first game.
* Design: The lab environment, including teleportation locomotion and object interaction, has been implemented. A realistic grad student character with animations from Mixamo has been added to guide players. Additionally, the JournalUI has been enhanced with a task-checking feature that provides feedback on completed tasks.
* Implementation: The custom controller for the pipette aid simulation has been finalized and integrated into the VR environment. This controller uses a 3D-printed case and mounted progressive triggers for realistic operation. Position tracking for the custom controller has been achieved as well.
* Testing: Comprehensive system and playtesting plans have been executed, and results analyzed to refine user interactions. The final improvements were focused on liquid dispensing mechanics, controller dynamics, and task feedback.

While initial development was slower due the team’s limited experience with tools like Unity and Blender and unfamiliarity with biology lab equipment, the team has adapted effectively. The decision was made to concentrate exclusively on the first game to maximize quality and functionality for the sponsor.

**Project Description**

*Author(s):* Jonathan Kolesar

*Reviewer(s)/Editor(s):* Kayla Sanderson, Sam Gerstner

**Sponsor Background**

Dr. Melissa Srougi is an Associate Teaching Professor at North Carolina State University, in the Department of Molecular Biomedical Sciences. Her teaching and research focus on fostering innovation in STEM education, with an emphasis on immersive and interactive approaches to biotechnology and biomedical sciences. By developing tools that prepare students for real-world lab environments, Dr. Srougi aims to enhance their learning experiences through inquiry-based techniques and technology.

Dr. Srougi has identified limited lab accessibility as a critical challenge in preparing students for future careers in biotechnology. This issue inspired her sponsorship of this project, which leverages virtual reality (VR), and the presence of VR devices in university and k-12 schools, as a novel solution to improve student engagement and training.

**Sponsor Contact Information:**

- Name: Dr. Melissa Srougi

- Title: Associate Teaching Professor

- Department: Molecular Biomedical Sciences, North Carolina State University

- Address: 1060 William Moore Drive, Raleigh, NC 27607

- Phone: (919) 515-9396 (Department)

- Email: msrougi@ncsu.edu

**Problem Description**

Dr. Srougi is most concerned that biotechnology students do not have sufficient access, if at all, to laboratory equipment and other resources. This creates a barrier for students to practice proper lab procedures, such as practicing pipetting, preparing solutions, and performing investigative analysis, which are critical to learning for their academic and professional development and success.

Currently, there are very few opportunities for students to practice hands-on skills due to:

* Limited lab time & equipment availability: Lab spaces and equipment are finite, while the number of students entering the field is continuously growing. This leads to scheduling conflicts, leaving students with not enough practice.
* High costs & maintenance: Lab equipment is usually quite expensive to purchase, let alone maintain. Furthermore, consumables such as pipette tips and reagents can lead to additional costs that add up quickly. These consumables also create some environmental concern as they must be thrown away after use and usually come with sterile packaging that must be disposed of as well.
* Safety Issues: New/inexperienced students may not be comfortable with handling some very delicate and sensitive lab equipment or dealing with chemicals, which potentially leads to safety risks.

As a consequence, these limitations heavily contribute to inconsistent, hence insufficient, and potentially hazardous/life-threatening situations. Without enough practice, students may lack confidence in their abilities and struggle to perform adequately in a real laboratory.

**Proposed Solution & Project Goals/Benefits**

The team developed a VR-based training simulation that focuses on teaching key lab techniques, starting with a minigame centered on using the pipette aid. The simulation provides students with a realistic and interactive environment to learn and practice their skills. A custom-designed controller further enhances immersion by mimicking the feel and functionality of a pipette aid, creating an experience as close to real-life practice as possible.

**Major Goals**:

* Immersive VR: Utilize true-to-life models of tools and 3D environments to replicate a biotechnology lab where students can remotely engage in following lab protocols and procedures.
* Simulate key lab techniques: Provide in-depth tutorials with real-time feedback on tasks such as using the Pipette Aid and adjusting micropipette volumes. This will enable students to engage in step-by-step procedures in order to reinforce good/proper laboratory practices.
* Enhance learning via repetition: Every lab technique should be practiced repeatedly by students as many times as necessary, supported with guided instructions for them to sufficiently learn and retain lab safety rules.
* Observe safety: Incorporate safety measures/precautions such as wearing PPE and following proper sanitization practices, to reinforce the practice so that students remember and properly adhere to lab safety rules.

**Benefits**:

* For the Sponsor: Dr. Srougi will gain a tool that will enhance the educational experiences students are offered in her courses, by enabling them to practice lab techniques more frequently and effectively. This may lead to better overall academic performance and real-world preparedness.
* For the Students: Opportunities for students to practice lab techniques in the safety of a virtual environment would be unlimited, allowing them to gain higher confidence and competence levels at their own pace. This VR experience bridges the gap between theoretical knowledge and practical skills.
* For the University & Community: This project could serve as a model for other courses and departments to integrate newer innovative learning tools across different disciplines. In addition, it might also attract those students who are looking towards cutting-edge learning methodologies.

This project therefore aims at improving learning outcomes for students working on their biotechnology skills. By developing interactive VR minigames we can address the problem brought on by the limited access to physical laboratory equipment.This will ultimately translate into better-equipped graduates who will take their increased confidence of performing lab techniques into future careers.

# Resources Needed

#### *Author(s):* Sam Gerstner

*Reviewer(s)/Editor(s):* Kayla Sanderson,Raven Midgett

***Table 1: List of Resources Needed***

| **Resource** | **Purpose** | **Version** | **Licensing Information** |
| --- | --- | --- | --- |
| Virtual Reality System | Headset & controller system that the game will be developed for and tested on. | Meta Quest 3 |  |
| Desktop Computer | There are specific hardware requirements for running VR games in Unity and our laptops do not meet the requirements. |  |  |
| Unity | The games are to be developed in Unity. | 2022.3.5f1 | Student license |
| Arduino IDE | IDE for developing controller software | 1.8.19 (Legacy) | GNU GPL |
| Altium PCB Designer | PCB design software, used to create schematics of the circuits for the controllers. | 23.0.1 |  |
| Blender | Working on, isolating, and correcting models. | 4.2 | GNU GPL |
| Mixamo | Create animations for in game. |  | [Mixamo licensing](https://community.adobe.com/t5/mixamo-discussions/mixamo-faq-licensing-royalties-ownership-eula-and-tos/td-p/13234775) |
| Figma | Create wireframes, user flows, and UML diagrams | Stable release124.4.7 / September 18, 2024 |  |
| Oculus XR Plugin Unity Package | Required for developing for VR in Unity. | 4.0.0 |  |
| OpenXR Plugin Unity Package | 1.10.0 |  |
| XR Core Utilities Unity Package | 2.2.3 |  |
| XR Interaction Toolkit Unity Package | 2.5.2 |  |
| XR Legacy Input Helpers Unity Package | 2.1.10 |  |
| XR Plugin Management Unity Package | 4.4.1 |  |
| Unity Test Framework Package | Unit testing framework for Unity. | 1.1.33 |  |
| TextMeshPro Unity Package | Displaying UI text. | 3.0.8 |  |
| Meta Quest Link Application | Connects VR headset to computer and allows it to interact with Unity | 69.0.0.501.353 (69.0.0.501.353) |  |

# Risks & Risk Mitigation

#### *Author(s):* Sam Gerstner

*Reviewer(s)/Editor(s):* Kayla Sanderson, Jonathan Kolesar

* **Delayed access to assets:** Our game is the successor to a previous browser-based game. The previous game contained a large amount of assets that are transferable over to the new game, the transfer of which would speed up the development for our game. Due to issues with communication between our team and the team that developed the previous game — and had access to the assets — we received the assets relatively late, slowing our overall process in bringing everything together. To mitigate this risk we coordinated with Dr. Srougi to get the GitHub repository containing the assets forked and copied to a repository we have access to, which we were successfully able to do.
* **Hardware constraints:** As our game is a VR game, we need a VR headset system to develop the game on and a graphically powerful computer to do the development. For the first weeks of the semester, we did not have a VR system to get familiar with, but our sponsor was able to provide us with a system. In addition, only one of our laptops is graphically powerful enough to do the development of the games as needed, and we had to have the university provide us with a desktop for development - which we also received.
* **Playtesting delays:** Due to the aforementioned issues, we have had to delay our first playtesting session that was originally scheduled for October 9th to the following week, which could have set back development due to not getting accurate feedback early on. To mitigate this risk we went over as many features as we could in the playtesting session, and feedback from the first session informed improvements which were validated in a second session later in the semester.
* **General lack of experience:** Going into this project, nobody on our team had any experience developing in Unity. In addition, nobody had significant experience with modeling in Blender or working with assets/models, meaning any work that had to be done to existing models would be tedious, and any new assets that were made would take significant amounts of time. To address this, the team used much of the downtime during initial setup of the project to work with Blender and other software in order to become more experienced and maximize the effectiveness of our time.
* **Development of custom controllers:** As development of custom controllers is difficult and time consuming, any issues encountered during the process could lead to delays and issues if the rest of development is waiting on the controllers to be made. In addition, the controllers could end up not working properly and lead to the end product being unsuccessful. To mitigate this, a significant amount of time was put towards developing the controllers and many iterations were done in order to fix problems during development.

# Development Methodology

#### *Author(s):* Sam Gerstner*,* Raven Midgett

*Reviewer(s)/Editor(s):* Jonathan Kolesar

Our team is primarily using feature-driven development along with other Agile principles and methodologies. We worked using iterative development and originally planned four iterations over the course of the semester but have adjusted this to two iterations to accommodate some of the manifested risks we encountered like delayed access to materials. Our first iteration, iteration zero, was exclusively for research, planning, and eliciting requirements. Iteration zero was two weeks. Iteration one included the entire process of developing our minigame. Because the minigame we developed had so many associated risks we ended up devoting the entire semester to the one minigame instead of our original plan to make 2 minigames. Iteration one began two weeks into the semester and lasted for the remaining 12 weeks. This is a bit unconventional, but we have found that it was a good way to minimize or at least work with some of these risks we encountered.

To describe how we are utilizing feature driven development it is helpful to discuss what our process was like in iteration one. Below are the steps we took during iteration one.

1. First, we establish an initial game model. This is done through a combination of user flow diagrams and by developing requirements and system tests.
2. From the documentation developed in the previous step we build a list of features we will need to implement to fulfill the requirements.
3. Next, we plan which features to implement first and who will work on these features. Sometimes features can be worked on simultaneously and sometimes they need to be done procedurally. We distribute development tasks on a weekly basis, typically immediately after our sponsor meetings on Fridays. By assigning tasks right after sponsor meetings we were able to better incorporate sponsor feedback into our decisions and task planning. This is not a formal standup or scrum but we assign tasks as action items in our meeting minutes documents.
4. We implement one or more features. We are using GitHub as our version control but due to the nature of game development in Unity we have to be very careful not to cause merge conflicts with our game scene. Resultantly, we generally take turns working on one feature branch and when the feature is completed, we merge back into our main branch. Sometimes during debugging we will make a branch off of main, resolve our issue, and then merge back.
5. We test the developed features. This is done initially by the developer(s) and then done more formally by following along with the system test plan and seeing if the new features fulfill more of our requirements and make more of our tests pass. When some set of features is completed, we schedule playtesting to get additional feedback.
6. Then we work on the next features we can implement. At this step we may need to reprioritize our planned features depending on the results of testing or sponsor feedback. We continued from step 3 over and over again until all the features for the iteration were completed. This cycle of planning, implementation, and testing worked as mini iterations.

The main reason that we have chosen to do iterative and feature-driven development is due to the risks that we have mentioned, primarily focusing on “Delayed access to assets” and “General lack of experience”. As we have had multiple delays outside of our control, focusing on feature-driven development has allowed us to work on the specific features and parts of the project that do not rely on the parts of the project that the delays are affecting; in addition, doing iterative development allows us to backtrack to fix issues that arise as the delayed parts of development are completed. With our other main risk during development being our general lack of experience developing in Unity — especially developing for VR in Unity — focusing on implementing feature-by-feature allows us to build up our knowledge on Unity by completing more basic features, leading to us being more competent in Unity when we begin working on the more complex requirements.

# System Requirements

#### *Author(s):* Kayla Sanderson

*Reviewer(s)/Editor(s):* Raven Midgett*,* Satwika Kancharla

## **Overall View**

Users for this project will be students, likely college students who have already taken a prerequisite course on lab safety. These students may be NC State students or eventually students at other institutions. This simulation will give these students the opportunity to practice using essential lab equipment such as the pipette aid through task-oriented tutorials. The simulation is designed to allow students a significant amount of free choice, allowing them to explore the lab at their own pace. This should allow them the opportunity to make mistakes and receive real time feedback.

The first game focuses on the pipette aid and offers a realistic and engaging way to understand how the tools work. In the pipette aid simulation, players will practice handling how to prepare a pipette aid as well as how to draw and expel liquids accurately and safely.

**Functional Requirements**

**Game 1: Pipette and Pipette Aid**

**[Genre]**

The Pipette and Pipette Aid game is a tutorial-style game being designed as a puzzle game.

**[Target Audience]**

The Pipette and Pipette Aid game is targeted towards biotechnology students who have already taken a prerequisite class about lab procedures.

**[Actors]**

A1: The playable character is a student in the lab.

**[Goals]**

G1: The goal of this game involves the student learning the correct procedure for using the serological pipette and pipette aid.

G1.1 The student opens the journal and checks the amount of liquid to be aspirated currently.

G1.2: The student selects the appropriate pipette for the volume of liquid mentioned in the scenario.

G1.3: The student picks up the pipette aid in one hand.

G1.4: The student attaches the chosen pipette to the pipette aid.

G1.5: The students hold the pipette aid vertically.

G1.6: The student reads the gradation lines.

G1.7 : The student inserts the serological attached to the pipette aid into a sample container

G1.8: The student depresses the top trigger to slowly aspirate the sample up to the appropriate volume indicated on the serological pipette.

G1.9: The student removes serological from the solution bottle.

G1. 10 : The student checks the amount of solution drawn into the pipette

G1.11: The student moves pipette aid to the destination vessel.

G1.12: The student depresses the bottom trigger to slowly dispense the liquid into the destination vessel.

G1.13: The student removes the serological from the destination vessel.

G1. 14: The student removes the serological from the pipette aid

G1.15: The student disposes of the pipette into the disposal bin.

G1.16: The student places the pipette aid down.

G1.19 The student opens the journal and receives feedback on how well the task was completed

G2: The student is able to interact with a journal to access dialogue and feedback

G2.1: The student interacts with the lab journal and can see a task

G2.2 The student can see any previous dialogue from the grad student in the lab journal

G2.3 The student can see feedback in the lab journal if they did something that was not safe or sterile.

**[Rules]**

R1: The student is able to navigate around a lab environment and interact with lab materials

R1.1: The student is able to turn their head using the controller

R1.2: The student is able to move around the room using the controller

R1.3: The student is able to pick up the following objects that they will need to use during the lab: pipette aid, serological tips and the lids of solution bottles.

R1.4: The student is able to set down the objects from G0.3.

**[Constraints]**

**Con**.1 The games will be developed in Unity.

**Con**.2 Project will be playable on a Meta Quest 3 (sponsor desires the game to be system agnostic, but minimum constraint is Meta Quest compatibility).

# Design

### *Author(s):* Raven Midgett - High Level Design, Jonathan Kolesar, Satwika Kancharla - Low Level Design

*Reviewer(s)/Editor(s):* Sam Gerstner, Kayla Sanderson

## **High-Level Design**

Our project is built on a modular architecture designed to simulate a biotechnology lab environment. The primary objective is to provide students with an immersive and interactive experience to practice essential lab techniques, such as pipetting, solution preparation, and equipment handling. This project is developed primarily to be a standalone VR experience where a Unity game is built and run on any VR device but there are multiple possible system configurations for interacting with the project's minigames at different levels of hardware integration.

### Technologies and Paradigms:

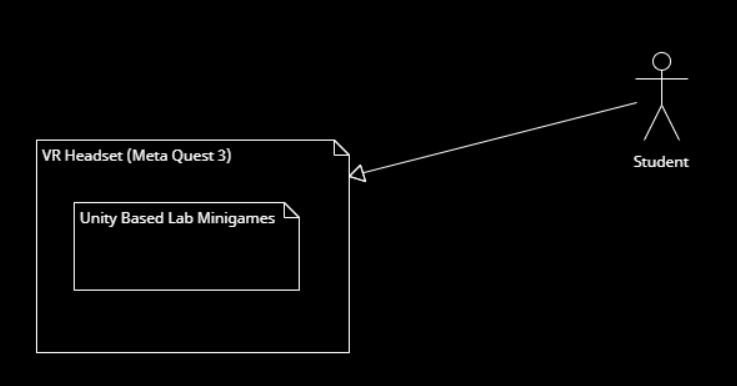
* Unity Game Engine: The project is developed in Unity, utilizing C# scripts for implementing gameplay logic and managing interactions.
* Object-Oriented Programming (OOP): Modular code structure for reusable components such as equipment classes and feedback mechanisms.
* Bluetooth Communication: Used for integrating custom controllers that mimic real lab tools (e.g., Pipette Aid).

### Core Components

* The VR Headset and standard controllers: In our case we developed with the Meta Quest 3 but the project should be transferable to any common VR system.
* The custom controllers: These controllers are models of the real life lab equipment: the pipette aid.
* A computer: This could be the students, the developers, or an institution. Requirements for the computer will be described further when each configuration is explained.
* VR Biotech Minigame: These are packaged as an android build for when the student is playing with the standalone VR configuration described below and as a WebGL build accessible through a web browser when playing in the web-based configuration described below as well.

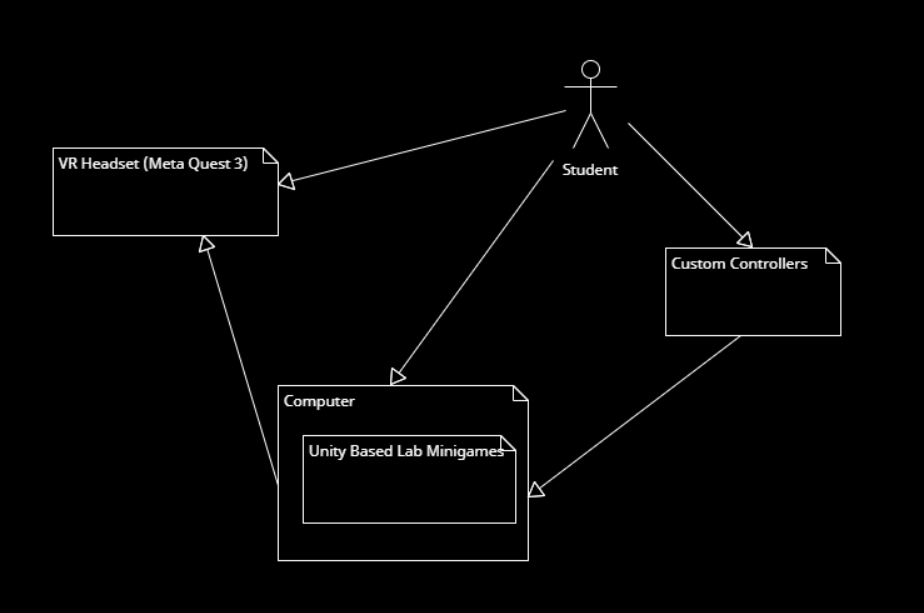
### Possible Configurations

To make our project accessible to as many students as possible while still exploring the potential that custom VR hardware allows, our minigames are playable in three different configurations that require different hardware and computer power.

* **The Standalone VR Configuration** is where the student interacts solely with a VR headset system that has a downloaded Android build of our minigames. The student does not need to utilize any other equipment or computer to have the lab minigame experiences. This design is displayed below in figure 1.

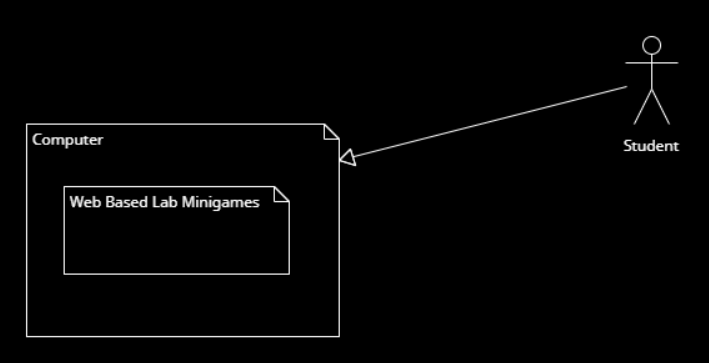
*Figure 1 : High level interaction of the student playing with the standalone configuration*

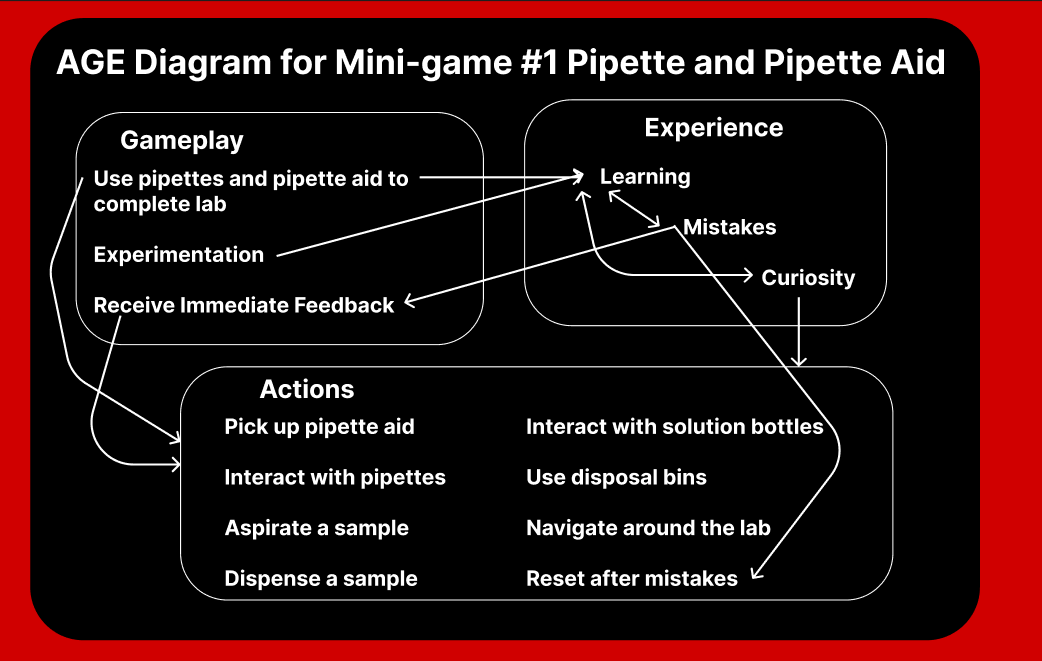
* **The Custom Hardware Configuration** is where the student is able to use the custom controllers associated with minigame 1 and 2. Due to technical limitations of the Meta Quest 3 that we worked with, the student must tether their VR headset system to a computer so that the custom controller can interface with the VR headset as well. In this configuration the student uses both the VR Headset and the custom controller while the Unity based minigames are executed by their computer. In this circumstance the student needs to use a computer compatible with their specific VR headset system. These relationships are described in figure 2.



*Figure 2 : High level interaction of the student with the custom hardware configuration*

* **The Browser Based Configuration** is where the student interacts solely with the web browser on their computer to access the minigames. The minigames will need to be built as a webgl build and hosted on a website. This is not something that has been implemented yet but is an end goal of the project. This is less of an immersive experience but allows interactions without the need for additional hardware. This configuration can be run on a system with very low graphical specifications. The relationship is described in the figure 3 below.

*Figure 3 : High level interaction of the student with the browser based configuration*

****

*Figure 4: AGE Diagram for Game 1*

### Gameplay Logic

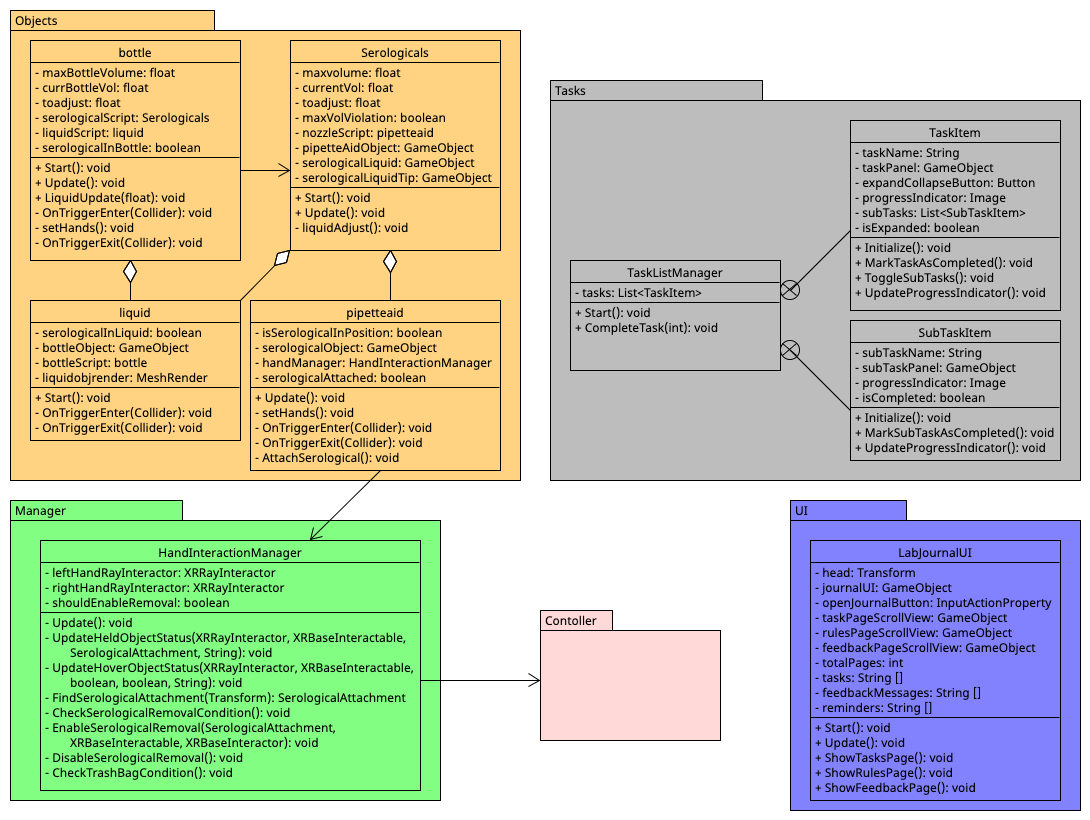
The above AGE diagram, Figure 4, for the Mini-game #1, Pipette and Pipette Aid, show how various elements of the game interact. It’s divided into three areas: Gameplay, Experience, and Actions.

* Gameplay
  + Players engage in lab activities using pipettes and pipette aids in an experimental nature, since it lets players try various methods to solve the task at hand, while immediate feedback is provided for every action taken to support learning.
* Experience
  + The mistakes underline the learning process players undergo while using the simulation. This drives curiosity, allowing the player to explore, manipulate, and try out different things to learn better. In this design, students can make mistakes and reset without the consequences of the real world.
* Actions
  + Picking up the pipette aid
  + Manipulation with pipettes and bottles of solutions
  + Aspiration and dispensing of samples
  + Using disposal bins
  + Navigating around the lab and resetting after making mistakes

## **Low-Level Design**

**UML Diagram**

Our project includes a variety of objects that are represented by assets that each have functionalities and are interconnected with each other as shown in our UML diagram.



*Figure 5: UML Diagram*

Our design for the game has been divided into 4 different parts as shown in Figure 5. The group of objects: bottle, serological, liquid, and pipette aid, each include scripts containing methods representing their functionalities. The bottle script and serological scripts allow for validity checks ensuring that serological is in the bottle and is attached to the pipette aid when aspirating and dispensing the liquid. The bottle uses the serological script to perform some of these checks. Both the bottle and the serological contain liquid which uses the amount to determine the presence and absence of liquid in each of these. The pipette aid handles the attaching to the serological to perform actions like dispensing and aspirating and detaching from it for disposal. It uses the HandInteractionManager for these functionalities specific to the ray interactors in VR. The customer controller also interacts with this for its functionality. All of these are managed my TaskManager that is used to relay feedback to the users through the Journal that is implemented using LabJournalUI.

### Custom Controller Integration

The custom controllers simulate real lab tools, with hardware components used to enhance the immersive experience.

Hardware Specifications

* 3D-Printed Shells: Create a realistic feel for the tools.
* Arduino-Based Microcontrollers: Enable Bluetooth communication\
* Mounting System for Meta Quest Controller: Enables motion and orientation tracking.

Integration Approach

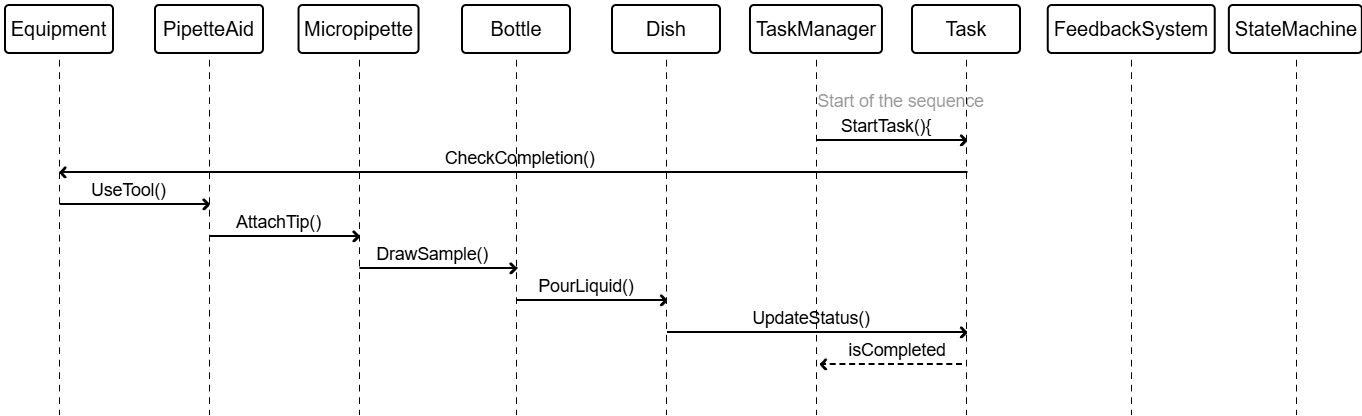
* Motion Data Handling: The headset tracks the Meta Quest controller attached to the custom controller.
* Bluetooth Communication: The controllers send sensor data to Unity in real time for accurate button inputs and pipette aid functionality. The controllers connect to a custom Bluetooth dongle that sends the data into Unity via a Serial connection.

### Feedback System

The feedback system is designed to assist students by guiding them through tasks and providing correction when necessary.

* Visual Feedback: The Lab Journal displays task instructions, error messages, and success alerts.

### Data Flow and UML Sequence Diagram

****

*Figure 6: Sequence Diagram for Game 1*

Example Interaction Sequence:

1. User Action: The player picks up the Pipette Aid.
2. Task State Update: The system checks if a pipette tip is attached. If not, then a visual alert is triggered.
3. Fluid Simulation: Once validated, the FluidSimulation class handles the liquid behavior during drawing or dispensing.
4. Feedback Update: The Lab Journal is updated based on the outcome, and the state machine transitions to the next phase.

UML Sequence Diagram

* Components Involved: Player Input, PipetteAid, FeedbackSystem, TaskManager, FluidSimulation
* Interactions: Illustrates how the system processes picking up the Pipette Aid, checking conditions, providing feedback, and updating the state.

Data Structures

* Task Class: Manages the current task's ID, description, status, and required equipment.
* Equipment Class: Base class for all lab tools with common attributes and methods.
* State Machine: Maintains a list of states and valid transitions between them.

Planned Future Enhancements

* Cloud-Based Progress Tracking: Integrate backend services for saving progress.
* Expanded Equipment Simulation: Add complex lab procedures.

### GUI Design

****

*Figure 7: GUI Design from Simulation*

We initially used the GUI design from the simulation as shown in Figure 7 as a starting point while planning to create our environment in Unity, along with inputs from visiting our sponsor’s lab.

# Implementation

#### *Author(s):* Raven Midgett, Kayla Sanderson

*Reviewer(s)/Editor(s):* Sam Gerstner

## **Iteration Definition & Current Status**

**Iteration 0 (Planning and Requirements Elicitation)**

Start Date: Aug 23th End Date: Sept 6th

This iteration was spent researching how to develop in unity, and for VR specifically. A significant amount of time was spent in attempting to learn Blender for asset development initially as it was unclear if the previous games assets would be able to be acquired.

**Iteration 1 (Minigame 1, Pipette Aid)**

Start Date: Sept 6th End Date: Dec 1st

\*This iteration references requirements for just Minigame 1 Pipette Aid.

Features implemented:

* Lab scene was implemented. This isn't necessarily a feature but the most basic part of our implementation that had to be created before features could be added.
* Locomotion features include point and click teleportation and snap turning. This was later updated following the playtesting feedback to use a system where we press the joystick down to trigger the movement and [R1.1 & R1.2]
* Grabbable objects implemented largely with XR interaction toolkit components [R1.03, G.04, G1.2, G1.3, G.10]
* Collision between objects, the walls, floors, and furniture. Additional hitboxes were added post playtesting to address issues[R1]
* Selecting and attaching correct serological tip [G1.2, G1.3, G1.4, G1.5, G1.6]
* Sensing if the pipette aid is held vertically and giving feedback to encourage correction [G1.5]
* Aspirating and dispensing liquid with the pipette aid [G1.7, G1.8,G1.9, G1.11, G1.12, G1.13 ]
* Removing and disposing of the serological tip [G1.14 , G1.15]
* The Lab Notebook [G2.1, G2.2]
  + Have a specific amount of liquid to be dispensed as a task that can be completed and feedback received [G2.16]

**Security Considerations**

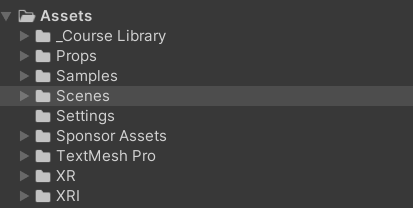
Generally, security concerns are outside the scope of our project but we will discuss how they would be considered in future iterations on this project by another team. Our current implementation of our system does not include any entering of sensitive information that would warrant security concerns but as our game is an educational tool it is likely that future iterations would include a system for tracking and monitoring student progress and outcomes. This could include collecting their names, educational institution, grades, and other information that could be at risk.

We plan to have our final game implementation available as a web-based browser game. This will mean our Unity game is converted to a WebGL build and hosted on a website. If some kind of user accounts were implemented or other data were collected, future teams could look into using the OpenGL extension GL\_ARB\_robustness. This would prevent denial of service attacks. There are also common methods for ensuring authentication in web applications. Future teams may want to investigate cookie authentication and API Key authentication.

## **Project Folder Structure**

When developing with Unity often hundreds of folders and files are created in the background and stored in the project folder hierarchy that are somewhat unimportant at least to the extent that they wouldn’t be relevant to any other teams working on this project in the future. In covering the project folder structure, only the major files and folders being used at this point will be discussed. For information about a lot of the auto-generated content that Unity creates in the project it may be useful to refer to Unity’s forums or knowledge articles.

**Assets Folder**

The assets folder is where all of the assets in the project are stored. This includes 3D models, textures, game scripts, Unity scenes, and prefabricated game objects. Figure 8 below shows the current folders within Assets.

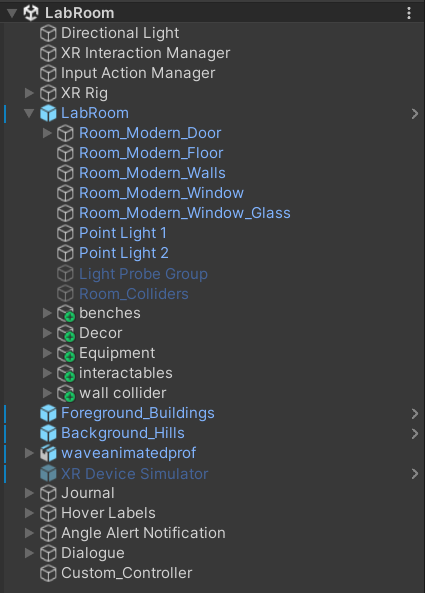
*Figure 8: Assets Folder from Unity*

Here is a brief description of what is in each Assets subfolder:

* Course Library: We based the initial implementation of our project off of a Unity Learn platform tutorial. Everything in this folder came with an empty base VR project from this tutorial. It includes things like VR game object prefabs, furniture prefabs, simple scripts that enable the VR integration, and more. Not every item in this folder is used in our current implementation but it has been a great resource and starting point.
* Props: This folder contains any asset that our team bought, developed, or altered ourselves. The 3D models for the lab benches, most of the lab equipment, and the grad student character are here.
* Samples: This folder contains an XR Device Simulator game object that is rigged to let you play the minigames within the Unity editor without using a VR headset. This game object can be dragged into any part of the Unity scene hierarchy to be used.
* Scenes: The scenes folder has the scene in which our game takes place. The scene is named LabRoom. There is further discussion of the hierarchy for this scene below. There is also currently another scene in this folder called LabRoomTestDialogue. Occasionally we create duplicate scenes during development to test new features without editing our working version of the scene.
* Settings, XRI, XR: These all have scripts and game objects that set up and give configurable settings for the VR components of the project.
* Sponsor Assets: This folder contains 3D models and textures of various lab equipment that was provided to us by our sponsor from previous work on a bio lab simulation game.
* TextMesh Pro: This folder is generated from a package we are using of the same name that allows us to create UI text elements.

**Lab Room Scene**

The Lab Room scene contains all of our game implementation and development thus far. Its components consist of the VR-enabled game objects, lighting, interaction managers, and all of the 3D assets that are within the lab room. Below is figure 9, that shows some of this hierarchy expanded. Important GameObjects are discussed below the figure.



*Figure 9: Hierarchy of Objects from Unity*

* XR Interaction Manager, Input Action Manager, and the XR Rig are all responsible for how the VR headset interfaces with the minigames. They each have various scripts and components attached to them that can be customized to tweak the experience when playing the minigames in VR.
* LabRoom has all of the 3-D models and assets that make up the room. This includes the walls and floors, all of the furniture in the room, lighting, and the lab equipment. This is broken down into some subfolders with fairly straightforward naming conventions.
* Journal is a game object that contains all of the UI elements for the games lab journal and the scripted behaviors associated with the journal.
* Hover Labels is a game object that contains the UI elements and scripting for the labels that appear over objects that are being pointed at.
* Angle Alert Notification is a game object that contains the UI elements and scripting for the alerts about the angle of the pipette aid.
* Dialogue is a game object that contains all of the UI text elements for in game dialogue and their scripted interactions.
* Custom Controller is a game object that controls the scripted interactions and integration of the custom controller.

## **Project Configuration/Settings**

Our project, because of the nature of using VR and Unity, has a lot of configuration settings. Below we will discuss the vital settings and configurations needed on the VR headset itself, the Meta Quest Link app, and Unity.

**VR Headset (Meta Quest 3)**

If using other VR headsets different configurations and considerations may need to be made. Our project has been developed to be cross platform between any common VR system but we will only provide specific settings for the Meta Quest 3 headset.

* The Meta Quest must be put in developer mode, this is done with the Meta Horizon Mobile Phone App. Instructions to do so can be found here from Meta themselves, done with the Meta Quest Link phone app.
* All of these developer setting must be activated on the headset settings
  + Enable Custom Settings
  + MTP Notification
  + Physical space features
  + Quest Link Auto Connect
* Under Quest lLink Settings, Quest Link must be enabled

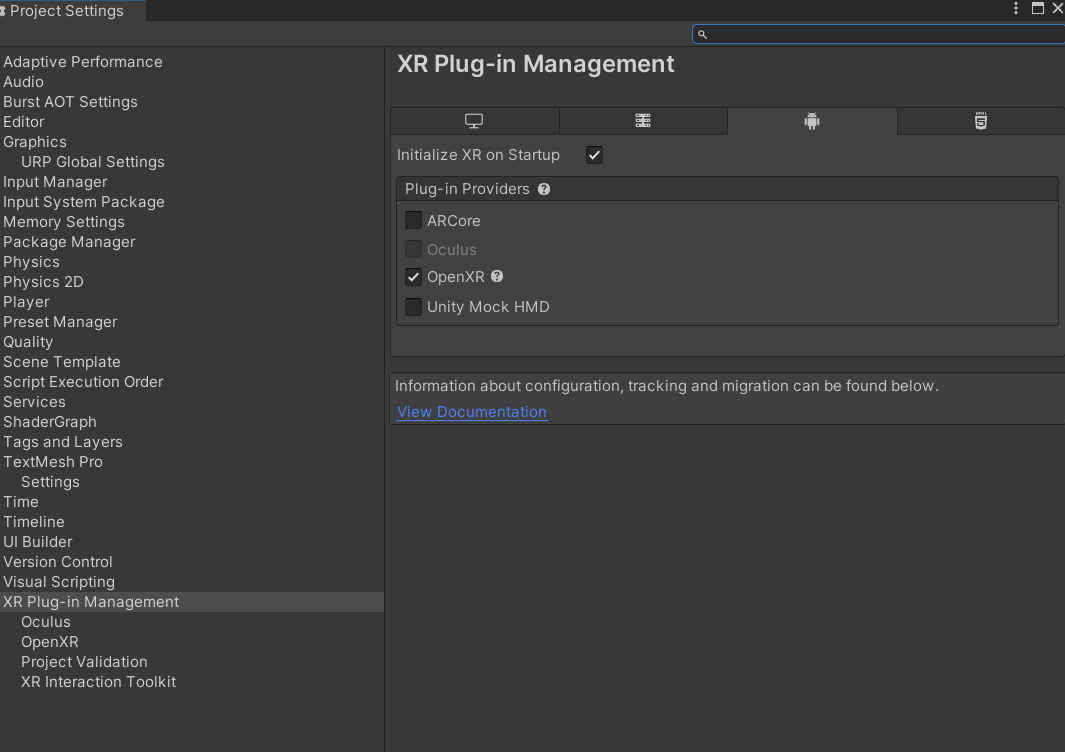
**Meta Quest Link App**

Again, if using a different VR headset this section may not apply. Other applications may be necessary to connect other headsets to your computer.

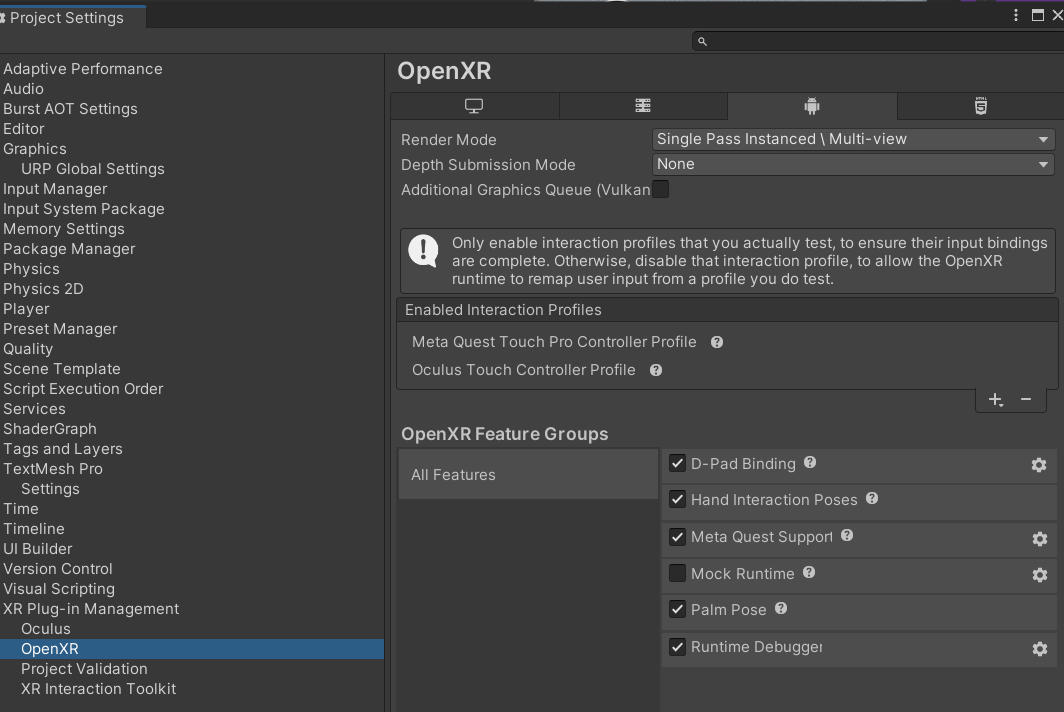
* Your VR Headset must be registered and connected to the Meta Quest Link app
* It is recommended to change the graphics preferences to the lowest refresh rate and the lowest render resolution
* Under General > Settings enable unknown sources

**Unity**

* Under Edit > Project Settings > XR-Plugin Management on the android tab OpenXR should be selected. This is shown in Figure 10 below.



*Figure 10: Unity Project Configuration Step 1*

* Under Edit> Project Settings > XR-Plugin Management > OpenXR on the android tab these interaction profiles should be enabled and these feature groups should be selected. This is shown in Figure 11 below.

*Figure 11: Unity Project Configuration Step 2*

# Testing

#### *Author(s):* Satwika Kancharla

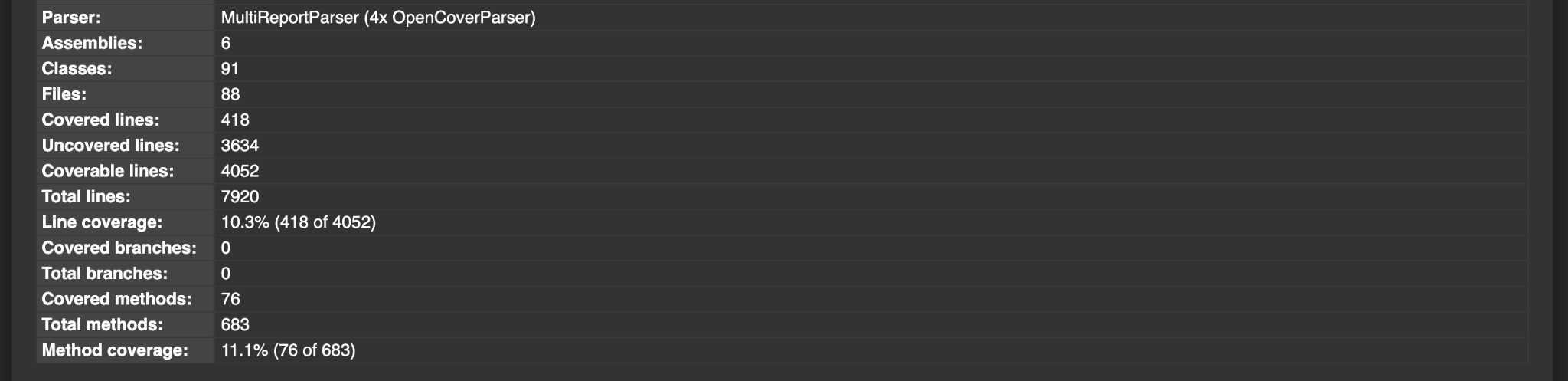
*Reviewer(s)/Editor(s):* Kayla Sanderson, Sam Gerstner

## **Overall View**

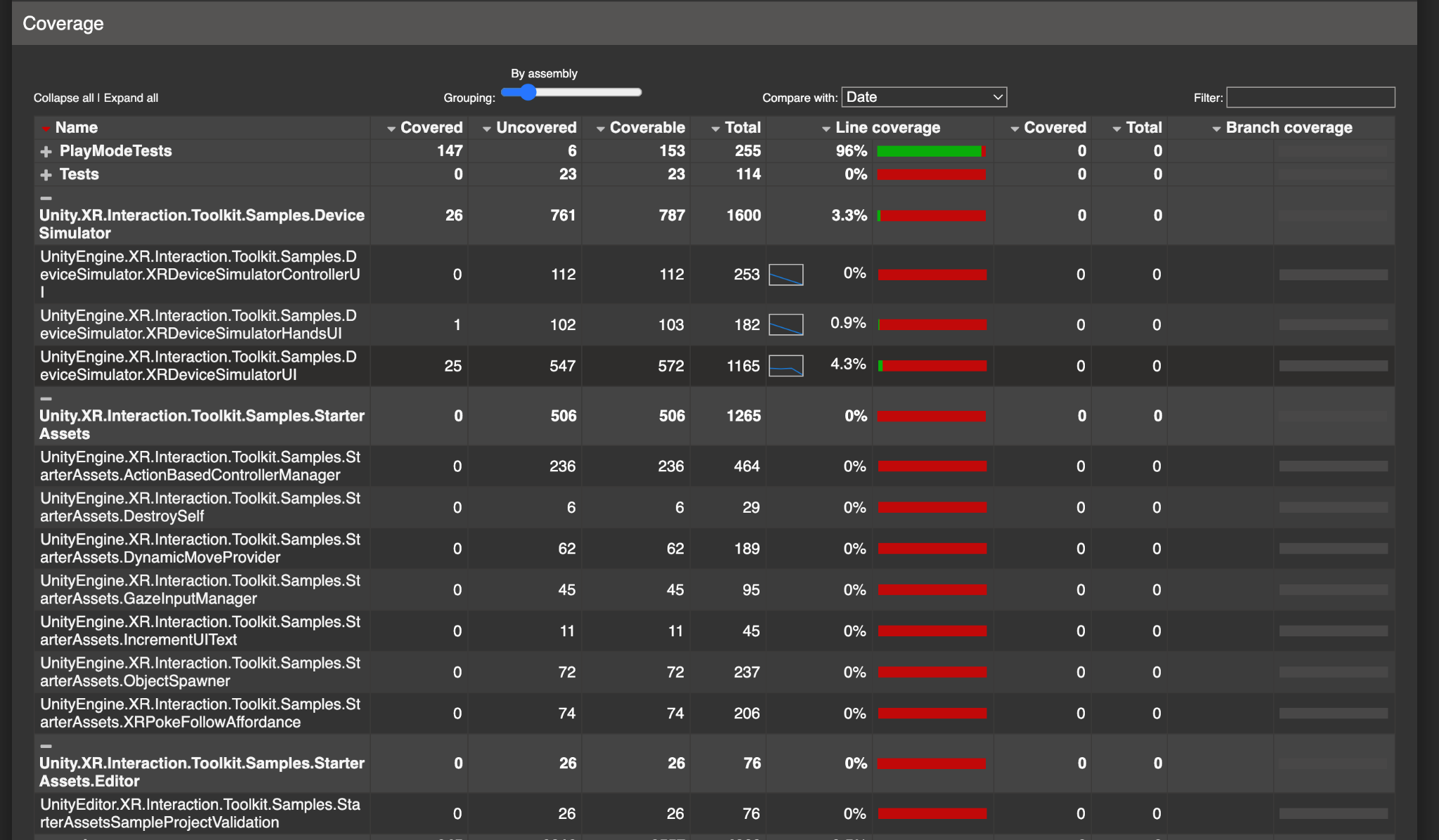
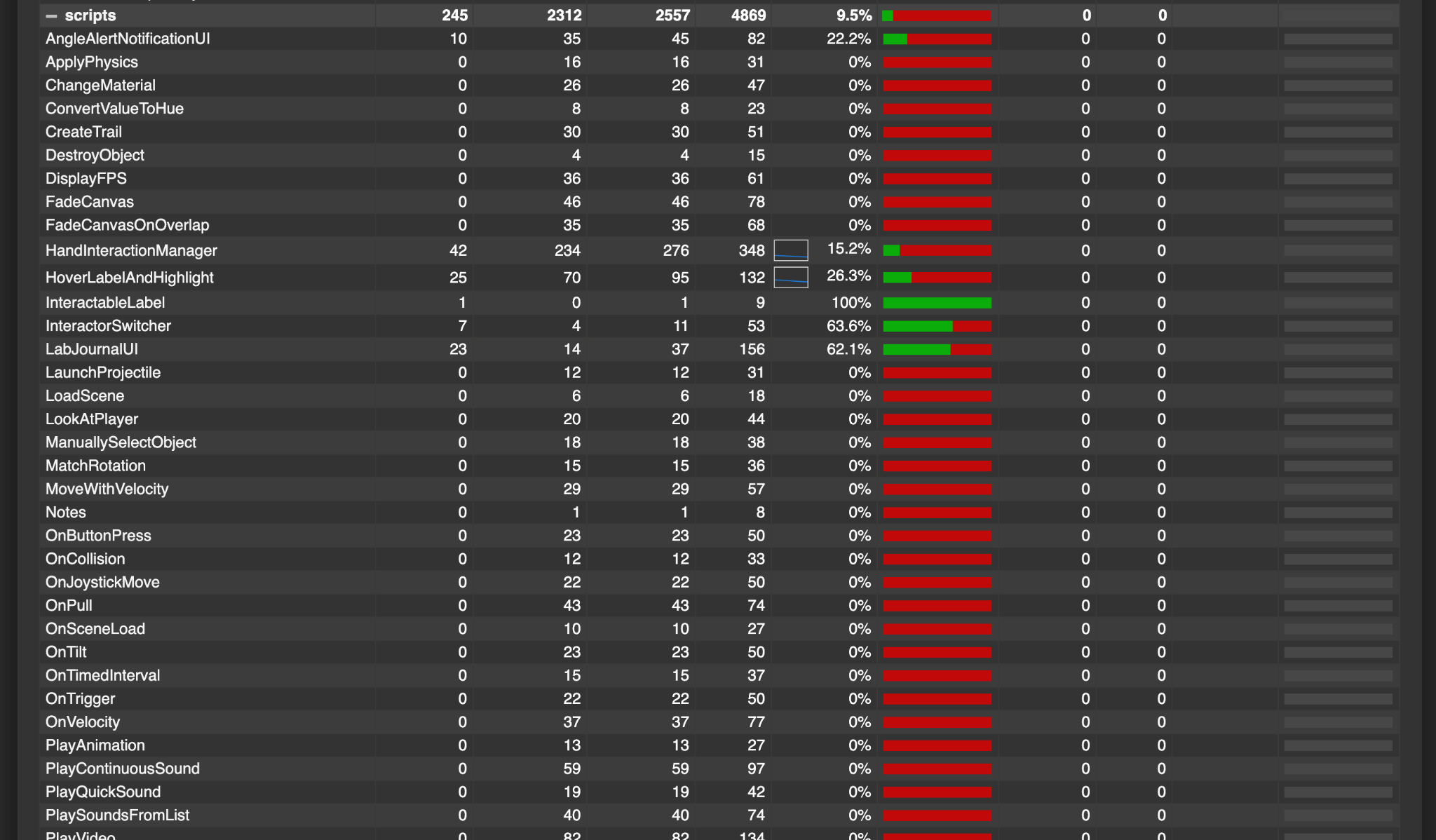
For our project, we plan on performing three major types of tests: System Testing, Play Testing, and Unit Testing. Since our project involves creating a game using Unity, system testing helps ensure that multiple aspects of the game (both assets and interactions) work correctly in the overall story. At the same time, this also helps test our environment and how individual components interact with it as a whole. In relation to this, to test the individual components, we are using Unit Testing. Playtesting helps receive feedback from our sponsor and users to validate our project, ensuring it is what they are looking for, or making changes to it if any.

## **Unit Testing**

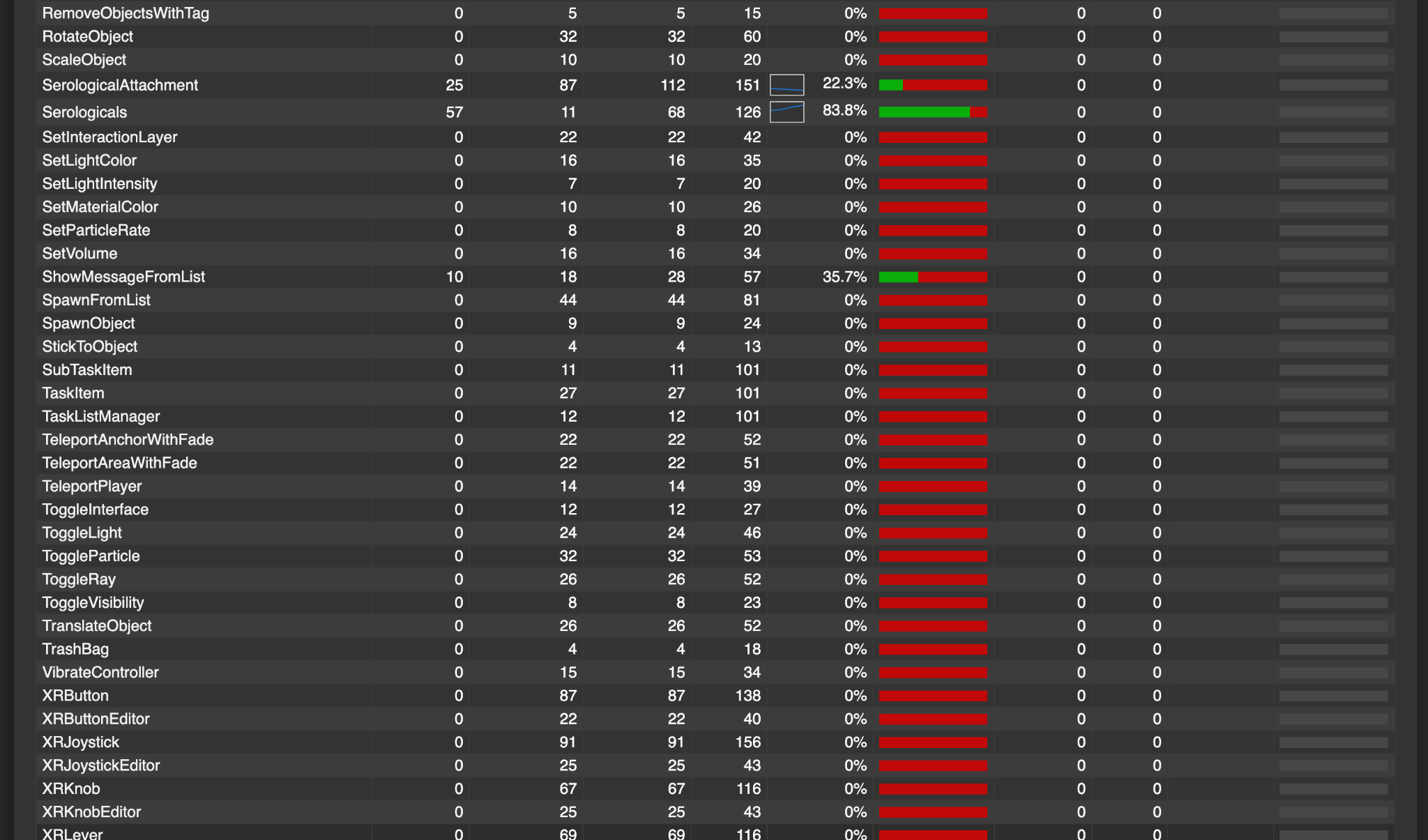
Since our project is being developed in Unity, we have decided to use the Unity Test Framework for unit testing. We used Play Mode Tests in Unity to test different parts of the scripts that we wrote for different objects.

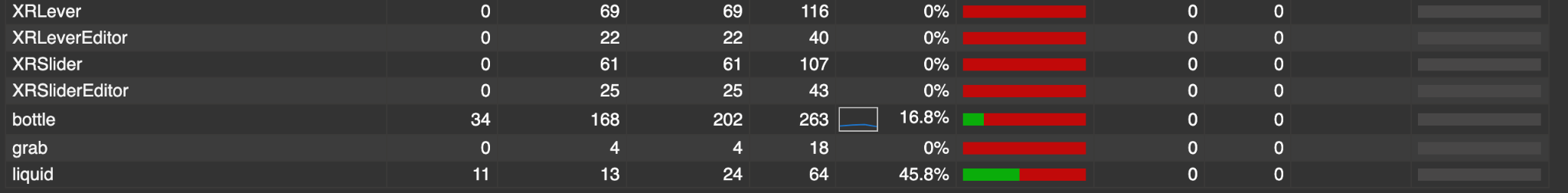
**Coverage Report:** 

*Figure 12: Summary of the Coverage Report*

*Figure 13: Coverage Report Part 1*

*Figure 14: Coverage Report Part 2*

*Figure 15: Coverage Report Part 3*



*Figure 16: Coverage Report Part 4*

Figures 13, 14, 15, and 16 represent the coverage report generated using the Code Coverage tool on Unity. As can be seen in Figure 12, our line coverage is about 10.3% and our method coverage is about 11.1%. These metrics are very low compared to the percentage that is expected for a software project, because our game consists of a lot of scripts that have been generated by Unity. From the figures above, the only scripts that we modified or added are LabJournalUI, SerologicalAttachment, Serologicals, bottle, and liquid. When finding the average line coverage for these scripts, we have a line coverage of about 35%. This still continues to be lower than expected, however, this is largely because of how dependent each object, its scripts and methods are on each other. For example, each object depends on colliders, rendering, and physics to function the right way, which are a lot of external factors that we were not able to test using Play Mode Tests or unit tests. Despite this, all of 13 of our unit tests pass, showing the accuracy of our code for each individual object and script. The integration of all of these features is tested in the next section: Acceptance Testing.

**Acceptance Testing**

Acceptance Test Plan: Team 33 (General Test Cases)

Sam Gerstner, Satwika Kancharla, Jonathan Kolesar, Raven Midgett, Kayla Sanderson

Test Plan Executed: December 6, 2024

For conducting the system tests (these are preconditions for running all the system tests):

* To run the game on the headset, build the game on Unity, upload the build to the headset using the Meta Quest Developer Hub, and run the game on the headset

***Table 2: System Test Plan for General Actions in Unity***

| **Test ID** | **Description** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| Test 1:  Movement with no obstacle | *Preconditions:* Game runs with headset enabled   1. Select a location to move to on the ground using standard controller 2. After ray turns white click button to teleport | You are teleported to the correct location | You are teleported correctly |
| Test 2:  Movement with small obstacle | *Preconditions:* You are able to move about the lab space   1. Try to teleport into a table by hovering the controller in the direction of the table with no other tools in the way | Ray should remain red to indicate the path is blocked | The ray remains red, preventing the user from teleporting/standing inside the table. |
| Test 3:  Movement with wall obstacle | *Preconditions:* You are able to move about the lab space   1. Try to teleport through the walls of the lab | Ray over walls should remain red and path should remain blocked | The ray remains red when attempting to select a wall as a teleport location, thus preventing the user from phasing through the walls. |
| Test 4:  Pick up pipette aid | *Preconditions:* You are near the table holding the pipette aid on the same side of the desk as it   1. Using the right hand controller, move so that the ray connects with the pipette aid and use the correct button to select it | The pipette aid is in your hand and fits in a manner similar to how it would be held in the lab | The pipette aid is successfully picked up with the right hand, after selecting it with the controller ray. |
| Test 5:  Drop pipette aid | *Preconditions:* You are holding the pipette aid   1. Drop the pipette aid onto the ground | See the pipette fall to the ground and make sure it does not fall through the ground | The pipette aid falls to the floor without falling through |
| Test 6:  Pick up the pipette aid from the floor | *Preconditions: pipette aid is on the floor*   1. Pick up the pipette aid by pointing to it on the floor towards the pipette aid and check that the ray turns white 2. Click the button to pick up the pipette aid | Pipette aid is hand at the correct angle. | The pipette aid is picked up in the hand and it’s positioning matches the orientation of the user’s hand |

The system test plan above describes different scenarios that are common to all of the games pertaining to moving locomotion within the Unity project and general interactions with major assets that represent equipment (pipette aid). Each of these tests acts as a prerequisite for the development of the rest of our games. With all of the actual results being the same as the expected results, we have the basis setup to implement other assets of our project.

Acceptance Test Plan: Team 33 (Game 1)

Sam Gerstner, Satwika Kancharla, Jonathan Kolesar, Raven Midgett, Kayla Sanderson

Test Plan Executed: December 6, 2024

For conducting the system tests (these are preconditions for running all the system tests):

* To run the game in the simulation, enable the XR Simulator and hit the play button in Unity
* To run the game on the headset, build the game on Unity, upload the build to the headset using the Meta Quest Developer Hub, and run the game on the headset

***Table 3: System Test Plan for Game 1***

| **Test ID** | **Description** | **Expected Results** | **Actual Results(simulated)** | **Actual Results** |
| --- | --- | --- | --- | --- |
| Test 1:  Minigame starts without being packed | *Preconditions:* None   1. In Unity, press “Launch as Unity game”. 2. *Check Results* | Game launches as an outside application and lab scene is visible (or start menu depending on development stage) | Game launches successfully | Yes, the game successfully launched and displayed the lab scene |
| Test 2:  Minigame 1 launches correctly | *Preconditions:* Game Launches   1. Select game 1 (the pipette minigame) from the 3 available games 2. *Check Results* | Character is moved into lab with the minigame 1 lab journal available in their hand (probably right hand) | Character does not move into lab but spawns in lab | Yes, the character correctly entered the lab, however, the Journal does not appear in a hand because it’s implemented as a floating UI instead |
| Test 3:  Open Lab Journal | *Preconditions:* In Minigame 1   1. Using the left-hand controller, open the lab journal. 2. Navigate to the procedures tab 3. *Check Results* | Journal opens and lab procedures for pipette aid can be seen | Journal opens successfully and player can navigate to relevant tabs including the procedures tab | Yes, the journal successfully opened and displayed a task |
| Test 4:  Task view | *Preconditions:* Lab journal is open   1. Ensure the Task panel is currently selected 2. *Check Results* | Journal is open to task page the current task amount to be dispensed can be seen | Journal is open and currently 13 ml is requested to be dispensed with 0 currently dispensed | The user is presented with a single assigned task. |
| Test 5:  Open  Journal Down | *Preconditions:* Journal is open   1. Look away from the journal and pick up the pipette aid 2. *Check Results* | The journal is left open.  Each step can be clearly read.Text is readable | Journal can be left open while other items are interacted with. This is somewhat finicky however as there is a fine balance to strike between the menu is close enough that it doesn’t interfere with other objects and we could leave the menu open while completing some tasks. | Journal is implemented as a togglable floating UI |
| Test 6:  Close Journal | *Preconditions:* Journal is open and on a table   1. Close the journal using your left hand menu button. 2. *Check Results* | The journal closes and task steps are no longer visible | Journal can be closed | Journal can be opened and closed by clicking the Menu button on the left controller |
| Test 7:  Correct Pipette Selected | *Preconditions:* Task 1 is selected and hands are empty   1. Given the scenario, highlight and pick up the correct pipette for the volume of liquid to be transferred. (selecting the closest while still being larger is preferred, ex if transferring 22ml choose a 25 ml pipette) 2. *Check Results* | The student selects the appropriate pipette for the volume of liquid mentioned in the scenario, and it is held in the left hand | Journal includes details about choosing the right pipette that the users can access | Journal includes details about choosing the right pipette that the users can access |
| Test 8:  Incorrect choice of pipette gives feedback | *Preconditions:* Task 1 is selected and hands are empty   1. Given the scenario, highlight and pick up a very small pipette (for example, if the volume to be transferred is 22ml pick 5 or smaller). 2. *Check Results* | * Student is holding the incorrect pipette in hand in the wrapper, HUD * Student gets a warning to look at the reminders’ page * The reminders’ page shows the warning: “Choosing a pipette that is too small may damage equipment” * Player is allowed to continue regardless, or may place the pipette down. | Journal includes details about choosing the right pipette that the users can access | Journal includes details about choosing the right pipette that the users can access |
| Test 9:  Put down pipette after selecting the wrong one | *Preconditions:* Task selected and wrong pipette is in hand   1. With the pipette in hand and still in paper, put the pipette back into the container of pipettes. 2. *Check Results* | Pipette is placed back in holder and player character voices confirmation “A larger one might work better”  (assuming trigger for contaminating filter has not been hit if so “I can’t contaminate another filter…”) | Journal includes details about choosing the right pipette that the users can access | Journal includes details about choosing the right pipette that the users can access |
| Test 10:  Open a closed container of solution | *Preconditions:* Game runs   1. Using an open hand (either should work), hover over a closed solution container and press the grab button 2. Remove lid 3. Lid can be placed on table near the container 4. *Check Results* | Container is open visually and the lid is visible on the table next to the container, neither is in either of the hands. | Lid can be successfully removed from the container and placed on the table. Putting it in a particular direction (up or down) in the simulated version is somewhat difficult. | Yes, the container is open and the lid is on the wherever I chose to place it |
| Test 11:  Close an open container of solution | *Preconditions:* Container with liquid is open   1. Hover over the lid and pick it up. 2. Put lid back onto container and secure it.(input undecided) 3. *Check Results* | Container is visually closed with corresponding lid on top | Lid can be placed on a container but not securely. Visually closed is extremely difficult in the simulated version | Lid can be placed on the container |
| Test 12:  Attempt to open 2 containers | *Preconditions:* Container with liquid is open   1. Using an open hand (either should work), hover over a closed solution container 2. Remove lid (unclear if this will be via some hand motion or controller input) 3. *Check Results* | Warning should appear in reminder section of notebook “Please remember to shut containers when not in use”  Audible warning of “Hmmm I should probably clean up” | Not implemented. Could have been implemented with feedback, boolean also not implemented for lid removal | No, neither warnings nor cues were triggered. |
| Test 13:  Student unwraps pipette | *Preconditions:* Pipette in hand   1. With closed pipette in hand and other hand free 2. Pinch each side of pipette cover “tails” and open ⅓ -½ of the way (so that pipette does not fall out) 3. *Check Results* | Pipette is still in cover but is open and can be removed. | Not implemented or deemed unnecessary due to scope adjustments. | Not implemented or deemed unnecessary due to scope adjustments. |
| Test 14:  Pick up pipette aid | *Preconditions:* Game is running   1. Pick up pipette aid with empty hand 2. *Check Results* | Pipette aid is in hand with no pipette attached. | Yes, the pipette can be picked up with either empty hand | Yes, the pipette can be picked up with either empty hand |
| Test 15: attach sterile serological pipette to pipette aid | *Preconditions:* Unwrapped pipette in hand   1. Holding pipette in left hand(in paper) and pipette aid in right firmly attach pipette to pipette aid 2. *Check Results* | Pipette aid and serological pipette should be attached with serological pipette firmly attached | Not implemented or deemed unnecessary due to scope adjustments. | No wrapper is implemented |
| Test 16:  Invalid test for attaching serological to pipette aid | *Preconditions:* Unwrapped pipette in hand   1. Holding pipette in left hand and pipette aid in right loosely attach pipette to pipette aid 2. *Check Results* | Pipette aid and serological pipette are loosely attached with some amount of gap between them  Verbal Reminder “I don’t think that's on all the way” | Not implemented or deemed unnecessary due to scope adjustments. | Not implemented or deemed unnecessary due to scope adjustments. |
| Test 17:  Correct invalid attachment to pipette aid | *Preconditions:* Pipette aid is partially attached   1. Grasping serological pipette in paper in one hand the pipette aid in the other, push the serological pipette firmly into the pipette aid until the gap is resolved 2. *Check Results* | Pipette aid and serological pipette should be attached with serological pipette firmly attached  Verbal Reminder “That seems better” | Not implemented or deemed unnecessary due to scope adjustments. | Not implemented or deemed unnecessary due to scope adjustments. |
| Test 18:  Paper is removed after pipette aid is attached | *Preconditions:* Serological pipette is attached and still has paper cover   1. Remove paper from serological pipette 2. Set paper aside in cabinet 3. *Check Results* | Pipette is exposed and attached to the pipette aid. Pipette should be sterile. | Not implemented or deemed unnecessary due to scope adjustments. | Not implemented or deemed unnecessary due to scope adjustments. |
| Test 19:  Paper removed prematurely | *Preconditions:* Pipette Aid and Serological is in hand   1. Before attaching pipette aid remove paper entirely(in order to do this pipette must be in hand) 2. Set paper aside 3. *Check Results* | Verbal warning “That might not be sterile anymore…”(may need to adjust so that sterility is not a boolean value but more of a if you touched above this line that area is not sterile but below is, may need to consult Dr. S.) | Not implemented or deemed unnecessary due to scope adjustments. | No wrapper is implemented |
| Test 20:  Vertical lower | *Preconditions:* Pipette is properly set up and sterile and container is open   1. Pipette is lowered vertically into open container 2. *Check Results* | Pipette is in container within angle tolerance  No verbal feedback given | Pipette can be lowered in vertically | Yes, successfully lowers vertically. |
| Test 21:  Nonvertical lower | *Preconditions:* Pipette is properly set up and sterile and container is open   1. Pipette is lowered into the container at an angle exceeding tolerance limits(probably more than 25 degrees off?) 2. *Check Results* | Pipette is in container outside angle tolerance  Verbal feedback “I should straighten the pipette before aspirating” | Visual feedback only given when pressing aspirate and dispense buttons. Currently the displays for visual feedback sit slightly out of view of the simulated version | Visual feedback only given when pressing aspirate and dispense buttons. |
| Test 22:  Lower into liquid | *Preconditions:* Pipette is properly set up and sterile and container is open   1. Lower pipette into liquid so that tip does not touch bottom (amount will vary based on how much needs to be aspirated and size of pipette) 2. *Check Results* | Pipette is visually in the liquid and the tip is not against the bottom | Pipette can be lowered in so that the tip is not against the bottom | Yes, positioning aligns with expected behavior. |
| Test 23:  Invalid lower into liquid | *Preconditions:* Pipette is properly set up and sterile and container is open   1. Lower pipette into liquid so that tip touches bottom 2. *Check Results* | Pipette is visually in the liquid and the tip is against the bottom  “If the tip is against the bottom it might struggle to draw up the liquid.” | Journal includes general safety aspects while using the serological | Journal includes general safety aspects while using the serological |
| Test 24:  Aspirate correct amount of liquid | *Preconditions:* Pipette aid is set up and lowered into container   1. With pipette aid in hand and serological pipette in liquid slowly press the top trigger while watching the gradation lines of the serological pipette releasing the top trigger when the correct amount of liquid has been aspirated 2. Remove serological pipette from solution container 3. View serological pipette to confirm amount 4. *Check Results* | Liquid fills the pipette to indicated line no verbal feedback | Hard to view gradations while serological is in container due to angle or player, but player can see that some amount of liquid has been aspirated | Yes, liquid levels adjusted appropriately. |
| Test 25:  Aspirate too much liquid | *Preconditions:* Pipette aid is set up and lowered into container   1. With pipette aid in hand and serological pipette in liquid press the top trigger until liquid goes above allowed amount and into pipette aid 2. Remove serological pipette from solution container 3. View serological pipette to confirm amount 4. *Check Results* | Serological pipette is full to the top with no gap.  Verbal warning :  “Oh no! I hope it didn’t get into the filter!”  Written warning:  “Overfilling the serological pipette can damage the pipette aid and may require the filter be changed by a lab supervisor.” | Warning system in the journal that gives you feedback | Warning system in the journal that gives you feedback |
| Test 26:  Aspirate too little liquid | *Preconditions:* Pipette aid is set up and lowered into container   1. With pipette aid in hand and serological pipette in liquid press the top trigger until liquid is below desired gradation line (just under 2-5 ml preferably) 2. Remove serological pipette from solution container 3. View serological pipette to confirm amount 4. Dispense aspirated liquid into empty container 5. Use check task button 6. View feedback 7. *Check Results* | Depending on initial amount requested and how far under the desired amount the feedback may vary. Something other than perfect should display and each level away from perfect should give a different result. | My lab run requested 21 ml I dispensed just over 18 ml and was told “you’re within 15% but aim for more precision” | Warning system in the journal that gives you feedback |
| Test 27:  Dispense liquid into receiving container | *Preconditions:* Liquid is aspirated into the pipette aid   1. With pipette aid in hand or set down using stand, close dispensing container 2. Open the receiving container and place lid sterile side down 3. Lower pipette aid into receiving container 4. Using bottom trigger slowly dispense liquid into receiving container ensuring all liquid is in the receiving container and there is none left in the pipette 5. Close receiving container 6. *Check Results* | * The receiving container has the liquid that was transferred * The serological is empty * The pipette aid is outside of the container | Containers cannot be closed but liquid can be aspirated and dispensed and serological is empty and bottles volumes are adjusted appropriately | Yes, liquid transfer and volume adjustments occurred correctly. |
| Test 28:  Dispose of serological pipette | *Preconditions:* Liquid has been dispensed into the receiving container   1. Remove Serological from pipette aid into empty hand 2. Dispose of pipette into correct container 3. *Check Results* | * The serological and are in a disposal bin | Serologicals can be disposed of | Yes, disposal works, but sterility tracking is not implemented. |

The system test plan above tests each of the goals and mini-goals from Game 1 with scenarios that follow the safety guidelines in the lab. It also looks at common mistakes that are made during each step of the process leading to warnings in the student’s journal. With the progress that we have made during the semester, we are passing all of the tests for functionality that we have implemented. However, as seen in Table 3, there were some functionalities like the wrapper for the serological that were out of our scope for this semester. Despite this, they are included in the system test plan to compare our initial tests with the implementation that we were able to finish.

## **Other Testing**

Playtesting is another type of testing that we used. Our sponsor recruited a few of her students who are the direct users for the games we are developing. They played the game while exploring different actions with the instructions that are provided on the game. Following this, we filled out a Google Form that contained a list of questions that refers to their feedback on the overall game. At the same time, they were also asked questions about the different actions they performed and any bugs that they encountered along the way in a think-aloud format. With more biotechnological experience, their thoughts also gave us improvements that we could implement to make the game more effective.

The results of the playtests and the changes that we made in response to the feedback are as follows.

***Table 3: Results for Playtesting Session 1***

|  |  |  |
| --- | --- | --- |
| **Comments** | **Number of Play Testers** | **How We Addressed Them** |
| Lack of instructions (missing instructions on moving around, on what buttons to press for actions, on identifying equipment) | 9/16 | Added more instructions to the dialogue box on how to perform different actions and added labels to the equipment to state what it is |
| Movement is not smooth (the rays from the controller curve around, random movements to the middle of the toom, playtester gets stuck against the wall, or under the shelf) | 5/16 | Created a new movement system that requires using the joystick and the trigger |
| Incorrect orientation of the pipette aid when in hand | 3/16 | Changed the orientation to be visible straight up when in hand |
| Journal clips into table causing awkward readability | 2/16 | Journal was updated to avoid clipping |

***Table 4: Results for Playtesting Session 2***

|  |  |  |
| --- | --- | --- |
| **Comments** | **Number of Play Testers** | **How We Addressed Them** |
| The labels improve interaction | 11/11 | N/A |
| The dialogue is lot more descriptive and instructions are clearer | 10/11 | N/A |
| Journal is easy to read | 9/11 | N/A |
| The movement has been significantly improved | 8/11 | N/A |
| Aspirating and dispensing was easy after getting used to it | 7/11 | N/A |
| Journal is not easily readable against dark backgrounds | 5/11 | The transparency of the journal has been updated |
| Instructions cannot be accessed once the dialogue box disappears | 4/11 | The instructions on controls have been added to the journal for easy access |
| The controller buttons are hard to understand when in the game | 3/11 | Allowing the playtesters and users to get familiar with the game can help make this easier |
| Some glitches with label flickering or objects passing through the tables | 3/11 | Made changes to the delay of labels |

# Suggestions for Future Teams

#### *Author(s):* Satwika Kancharla, Kayla Sanderson, Sam Gerstner

*Reviewer(s)/Editor(s):* Raven Midgett, Jonathan Kolesar

As we went through implementing this project, we made a lot of our decisions based on the time we had to implement a particular feature, or with the knowledge that we had at the time we were working on that feature. As a result, there are some features that can be redesigned to improve the maintainability of the game, which are listed below.

**Implement Unity Events:** Our current implementation uses variables in scripts that are updated by other scripts and objects to keep track of a dynamic value. For example, the ‘toadjust’ variable is used to update the amount of liquid in the bottle and serological upon aspirating or dispensing the liquid rather than an update just based on the triggering of other events. Additionally this variable is copied to another variable ‘prevToAdjust’ which is used to update the current task. The ‘prevToAdjust’ variable was made as a last resort as due to not using Events (and thus not being able to control when the ‘toadjust’ variable is cleared), we were forced to update tasks on a one frame delay. All of this to say, a lot of our scripts are tied and interlinked together. To avoid this and reduce coupling, Unity allows for using Unity Events for dynamic tasks that can be implemented or added to our current implementation of the game. Upon this implementation, using the previous example, when the update to the bottle is made, the serological and the task would see who called the event, and update themselves.

**Use Meta Quest Hand Tracking:** For the interaction between the Meta Quest headset and Unity game, we use the XR Interaction Toolkit. However, a limitation of this is that it does not allow using a custom controller along with a Meta Quest controller, one in each hand or an empty hand for gesture tracking. As a result, our custom controller currently has an attachment for inserting the Meta Quest controller that presses down a trigger on it, to state that the custom controller is being used. Based on the initial research we did, using the Meta Quest Hand Tracking system, we would be able to avoid this and accurately replicate the pipette aid, while maintaining the accessibility desired by our sponsor.

**Implement 2D Overlays for Event notifications:** As mentioned before, accessibility is a very important consideration for our sponsor. There are a lot of individuals who experience motion sickness in VR headsets, so our game can be played using the headset or using the XR Simulator available on Unity. Currently, we use the same system for event notifications on both versions. However, the aspect ratio and view experience on the simulated version is much smaller compared to the headset. So, some of the notifications are not readable in the simulated version. An example of this is the notification that pops up when aspirating or dispensing the liquid at an incorrect angle. Using 2D overlays for these notifications can improve the overall readability for these notifications in the simulated version.

**Adding Dynamic Feedback to the Journal:** One of the main goals of our game was to allow for feedback as the student made mistakes in the lab. With the time that we had for implementing the game, we were only able to include static feedback in our journal as a list of general things to consider rather than feedback as students make mistakes in the lab. Implementing dynamic feedback is one of the features that we suggest for a future implementation. A lot of the scripts currently include boolean variables that can be checked in order to activate this already.

# Bibliography

AI Tool Used: ChatGPT by OpenAI

Date of Access: October 7, 2024

URL: https://chat.openai.com

Prompt Used: "Review the Interim Project Report (IPR) and suggest improvements in sections "Development Methodology," "Design," and "Project Description."."

Contribution: The AI provided suggestions to enhance clarity, detail, and organization. These suggestions were reviewed, verified their accuracy, and integrated relevant ideas.

Limitations: The AI offered general guidance but couldn't fully understand the project's context or technical specifics, requiring adaptions to fit the assignment's requirements.

# Appendix

#### *Author(s):* Kayla Sanderson, Satwika Kancharla

*Reviewer(s)/Editor(s):* Raven Midgett

### Game 2: Micropipette Functional Requirements

**[Genre]**

The Micropipette game is a tutorial-style game designed as a puzzle game.

**[Target Audience]**

The Micropipette game is targeted towards biotechnology students who have already taken a prerequisite class about lab procedures.

**[Actors]**

A1: The playable character is a student in the lab.

**[Goals]**

**G1**: The goal of this game involves the student learning the correct procedure for using the micropipette.

G1.1: The student chooses the micropipette with the correct range of volume for the experiment.

G1.2: The student adjusts the dial by turning it to match the exact volume.

G1.3: The student selects the correct tip to match the type of micro pipette chosen.

G1.4: The student attaches the tip to the micropipette.

G1.5: The students holds the micropipette in the upright position.

G1.6: The student keeps the plunger depressed until the first stop and puts the micropipette 1-3 mm into the sample and slowly releases the plunger. (must wait for a second after completely depressing)

G1.7: The student inspects the tip for air bubbles after removing it from the container.

G1.8: The student keeps the micropipette at a 10-45-degree angle while transferring it from the solution to the destination vessel.

G1.9: The student expels liquid to the designated container by pressing slowly to first stop then a stronger pressure to second stop.

G1.10: The student removes the micropipette from the destination vessel before releasing the plunger.

G1.11: The student uses the eject button to dispose of the tip to the bin.

G1.12: The student puts down the micropipette onto the stand.

**G2**: The student is able to interact with a computer to access dialogue and feedback.

G2.1: The student interacts with the lab computer and a lab journal menu opens.

G2.2 The student can see any previous dialogue from the grad student in the lab journal.

G2.3 The student can see feedback in the lab journal if they did something that was not safe or sterile.

**[Rules]**

**R1:** The student shall wear their PPE kit the entire time.

**R2:** The student can only hold one thing in each hand at a time.

R2.1: Never overturn dials above or below the range of the pipette. It may break the instrument.

R2.2: Change tip after each volume transfer.

R2.3: Always store instruments upright.

R2.4: Smooth pressure should be applied to the plunger.

**R3:** The student is able to navigate around a lab environment and interact with lab materials.

R3.1: The student is able to turn their head using the controller.

R3.2: The student is able to move around the room using the controller.

R3.3: The student is able to pick up the following objects that they will need to use during the lab: micropipette, pipette tips, solution bottles and lids.

R3.4: The student is able to set down the objects from G0.3.

### Acceptance Test Plan: Team 33 (Game 2)

Sam Gerstner, Satwika Kancharla, Jonathan Kolesar, Raven Midgett, Kayla Sanderson

***Table 3: System Test Plan for Game 2***

| **Test ID** | **Description** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| Test 35:  Minigame starts without being packed | *Preconditions:* None   1. In Unity, press “Launch as Unity game”. 2. *Check Results.* | Game launches as an outside application and lab scene is visible (or start menu depending on development stage) |  |
| Test 36:  Minigame 2 launches correctly | *Preconditions:* Game launches   1. Select game 2 (the micropipette minigame) from the 3 available games 2. *Check Results.* | Character is moved into lab with the minigame 1 lab journal available in their hand (probably right hand) |  |
| Test 37:  Open Lab Journal | *Preconditions:* Game launches   1. Using the left-hand controller, open the lab journal. 2. *Check Results.* | Journal opens and lab procedures for micropipette can be seen |  |
| Test 38:  Task selection | *Preconditions:* Lab journal is open   1. From X available tasks for which a micropipette is appropriate, select task 2. (may require flipping pages? With left hand to navigate tasks) 2. *Check Results.* | Task is highlighted in journal to indicate the current task and step one of the task is in Heads Up Display (HUD). |  |
| Test 39:  Open  Journal Down | *Preconditions:* Lab journal contains information about task 2   1. Put the journal down in a designated spot (maybe in the cabinet but probably not). 2. *Check Results.* | The journal is put down in an open position. Each step can be clearly read.Text is readable |  |
| Test 40:  Close Journal down | *Preconditions:* Lab journal is open   1. Close the journal using your left hand. 2. *Check Results.* | The journal appears closed and task steps are no longer visible |  |
| Test 41:  Correct Micropipette Selected | *Preconditions:* The student’s hands are empty   1. Given the scenario, highlight and pick up the correct micropipette for the volume of liquid to be transferred. (selecting the closest while still being within the range of the micropipette) 2. *Check Results.* | The student selects the appropriate micropipette for the volume of liquid mentioned in the scenario, and it is held in the left hand. There should be a thought box that says “That seems like the right micropipette.” |  |
| Test 42:  Incorrect choice of pipette gives feedback | *Preconditions:* The student’s hands are empty   1. Given the scenario, highlight and pick up a very small micropipette. 2. *Check Results.* | * Student is holding the incorrect pipette in hand, HUD * Student gets a warning to look at the reminders’ page * The reminders page shows the warning: “Choosing a micropipette that is too small may damage equipment” * Player is allowed to continue regardless, or may place the micropipette down. |  |
| Test 43:  Put down the micropipette after selecting the wrong one | *Preconditions:* Micropipette is in hand   1. With the micropipette in hand, put the pipette back into the stand of micropipettes. 2. *Check Results.* | Micropipette is placed back in holder and player character voices confirmation “A larger one might work better”  (assuming trigger for contaminating filter has not been hit if so “I can’t contaminate another filter…”) |  |
| Test 44: Choose the right micropipette tip | *Preconditions:* Micropipette is in hand   1. With the micropipette in hand, open the lid of the correct micropipette tips box. 2. Attach the micropipette to one tip. 3. Pick the micropipette up from the box. 4. Close the micropipette tips box. 5. *Check Results.* | * The micropipette tip is present on the bottom of the micropipette and is secured. * The micropipette tips box is closed. |  |
| Test 45: Choose a large micropipette tip | *Precondtions:* Micropipette is in hand   1. With the micropipette in hand, open the lid of the micropipette tips box belonging to a larger micropipette. 2. *Check Results.* | * The reminders page shows the warning: “This tip seems to be too big for the micropipette you chose”. |  |
| Test 46: Choose a very small micropipette tip | *Precondtions:* Micropipette is in hand   1. With the micropipette in hand, open the lid of the micropipette tips box belonging to a smaller micropipette. 2. *Check Results.* | * The reminders page shows the warning: “This tip seems to be too small for the micropipette you chose”. |  |
| Test 47: Adjust the dial to the correct volume | *Preconditions:* Micropipette is in hand and tip is attached to it   1. With the micropipette in hand with a tip, adjust the dial to the value mentioned in the scenario. 2. *Check Results.* | * The value on the dial of the micropipette matches the volume mentioned in the scenario. * Player is allowed to continue to next step. |  |
| Test 48: Adjust the dial to a larger voume | *Preconditions:* Micropipette is in hand and tip is attached to it   1. With the micropipette in hand with a tip, adjust the dial to a higher value than the volume mentioned in the scenario. 2. *Check Results.* | * The reminders page shows the warning: “The value seems too large for the volume you need to transfer. * Player is allowed to continue to next step. |  |
| Test 49: Adjust the dial to a smaller voume | *Preconditions:* Micropipette is in hand and tip is attached to it   1. With the micropipette in hand with a tip, adjust the dial to a lower value than the volume mentioned in the scenario. 2. *Check Results.* | * The reminders page shows the warning: “The value seems too small for the volume you need to transfer. * Player is allowed to continue to next step. |  |
| Test 50:  Open a closed container of solution | *Preconditions:* Student chooses Game 2   1. Using an open hand (either should work), hover over a closed solution container 2. Remove lid (unclear if this will be via some hand motion or controller input) 3. Lid can be placed on table near the container. 4. *Check Results.* | Container is open visually and the lid is visible on the table next to the container, neither is in either of the hands. |  |
| Test 51:  Close an open container of solution | *Preconditions:* Container is open   1. Hover over the lid and pick it up. 2. Put lid back onto container and secure it. 3. *Check Results.* | Container is visually closed with corresponding lid on top |  |
| Test 52:  Attempt to open 2 containers | *Preconditions:* Container is open   1. Using an open hand (either should work), hover over a closed solution container 2. Remove lid (unclear if this will be via some hand motion or controller input) 3. *Check Results.* | Warning should appear in reminder section of notebook “Please remember to shut containers when not in use”  Audible warning of “Hmmm I should probably clean up” |  |
| Test 53:  Hold up the micropipette vertically | *Preconditions:* Micropipette is in hand with a tip attached   1. The micropipette is held vertically above an open container. 2. *Check Results.* | The micropipette is above container within angle tolerance |  |
| Test 54:  Hold up the micropipette non-vertically | *Preconditions:* Micropipette is in hand with a tip attached   1. The micropipette is held above the container at an angle exceeding tolerance limits(probably more than 25 degrees off?) 2. *Check Results.* | * The micropipette is held over the container outside angle tolerance * Verbal feedback “I should straighten the micropipette before putting it into the container” |  |
| Test 55: Depress the plunger to the first stop | *Preconditions:*Micropipette is in hand with a tip attached   1. While the micropipette is in one hand and outside the container, depress the plunger to the first stop. 2. *Check Results.* | * The plunger is depressed to the first stop outside the container * The container is open * The container’s lid is next to the container on the table |  |
| Test 56: Depress the plunger to the second stop | *Preconditions:* Micropipette is in hand with a tip attached   1. While the micropipette is in one hand and outside the container, depress the plunger to the second stop. 2. *Check Results.* | * The plunger is depressed to the first stop outside the container * The container is open * The container’s lid is next to the container on the table * The reminders’ page has a warning, “You may not be able to aspirate the solution if the plunger is depressed to the second stop”. |  |
| Test 57: Aspirate the solution from the container | *Preconditions:* Micropipette is in hand with a tip attached and container is open   1. While the micropipette is still depressed to the first stop, insert the micropipette into the container until it touches the surface of the liquid. 2. Release the plunger slowly. 3. *Check Results.* | * The micropipette is inside the container * There is liquid within the tip of the micropipette * The micropipette plunger is not depressed anymore |  |
| Test 58: Incorrectly insert the micropipette into the container | *Preconditions:* Micropipette is in hand with a tip attached and container is open   1. While the micropipette is still depressed to the first stop, insert the micropipette into the container so that it touches the bottom of the container 2. *Check Results.* | * The micropipette is inside the container * The reminder page has a warning, “You might want to lift the micropipette higher, so it aspirates the liquid correctly.” |  |
| Test 59: Inspect for air bubbles with none present | *Preconditions:* Micropipette is in hand with a tip attached and liquid is aspirated   1. Look at the micropipette tip to ensure there are no air bubbles. 2. *Check Results*. | * There are no air bubbles in the micropipette tip. * The player proceeds to the next step. |  |
| Test 60: Inspect for air bubbles when present | *Preconditions:* Micropipette is in hand with a tip attached and liquid is aspirated   1. Look at the micropipette tip to ensure there are no air bubbles (the tip contains air bubbles). 2. *Check Results*. | * The micropipette tip contains air bubbles. * The reminder page has a warning, “Make sure the tip does not contain any air bubbles.” |  |
| Test 61: Correctly dispense solution to the destination vessel | *Preconditions:* Micropipette is in hand with a tip attached and liquid is aspirated   1. With micropipette aid in hand, cover the dispensing container 2. Open the receiving container and place lid sterile side down 3. Lower micropipette into receiving container 4. Depress the plunger to the first stop 5. Once all the liquid has left the tip, depress plunger to the second stop. 6. Close receiving container. 7. *Check Results* | * There is no liquid inside the tip * All of the liquid is inside the receiving container * Both containers are closed/covered |  |
| Test 62: Incorrectly dispense solution to the destination vessel | *Preconditions:* Micropipette is in hand with a tip attached and liquid is aspirated   1. With micropipette aid in hand, cover the dispensing container 2. Open the receiving container and place the lid sterile side down 3. Lower the micropipette into the receiving container 4. Depress the plunger to the first stop 5. Close receiving container. 6. *Check Results* | * There is some liquid inside the tip * Most of the liquid is inside the receiving container * Both containers are closed/covered * The reminder page has a warning, “Depressing the plunger to the second stop while dispensing the liquid ensures that the liquid is completely dispensed” |  |
| Test 63: Dispose of the micropipette tip and place it back on the stand | *Preconditions:* Micropipette is in hand with a tip attached and liquid is dispensed   1. With the micropipette in hand, move it over to the disposal bin 2. Click on the button on the top to dispose off the tip in the bin. 3. Keep the micropipette back on the stand. 4. *Check Results.* | * The tip is in the disposal bin * The micropipette is back on the stand |  |
| Test 64: Dispose of the micropipette tip and place it back on the stand | *Preconditions:* Micropipette is in hand with a tip attached and liquid is dispensed   1. Click on the button on the top to dispose off the tip in the bin. 2. Keep the micropipette back on the stand. 3. *Check Results.* | * The micropipette is back on the stand * There is a warning on the reminders page, “The micropipette tip must be disposed off in the disposal bin for safety” |  |

The system test plan above looks at the different scenarios relating to mini goals and goals for Game 2 with the micropipette. Since we have not started working on Game 2 yet (is a part of Iteration 2), none of our system tests are passing.