המכללה האקדמית להנדסה
אורט בראודה
ORT Braude College

Capstone Project Phase B

Personalized Recommendation System for a Learning Environment

Project Number: 21-2-D-27

Supervisor

Dr. Anat Dahan

Team Members

Roy Goichman
E-mail: roygnr@gmail.com

Bar Israel

E-mail: baranvz@gmail.com

Table of Contents

Contents

1. Project Description

1.1 Project Review

The Virtual Classroom is a Unity application designed to run on the Oculus Quest 2 Virtual Reality (VR) headset and provide the user with information and recommendations on their personal learning environment and how to customize it for best results.
The application presents the user with a virtual classroom and a set of lessons followed by an examination to review what they've learned. During the lessons, various distractions of different types (visual and auditory) are generated, and the user's response to these distractions is monitored through head movement tracking and the final exam results. The application is then able to produce a report and deduce which type of distraction is more harmful to the user, and what they should avoid and include in their personal learning environment.

1.1.2 Project Flow

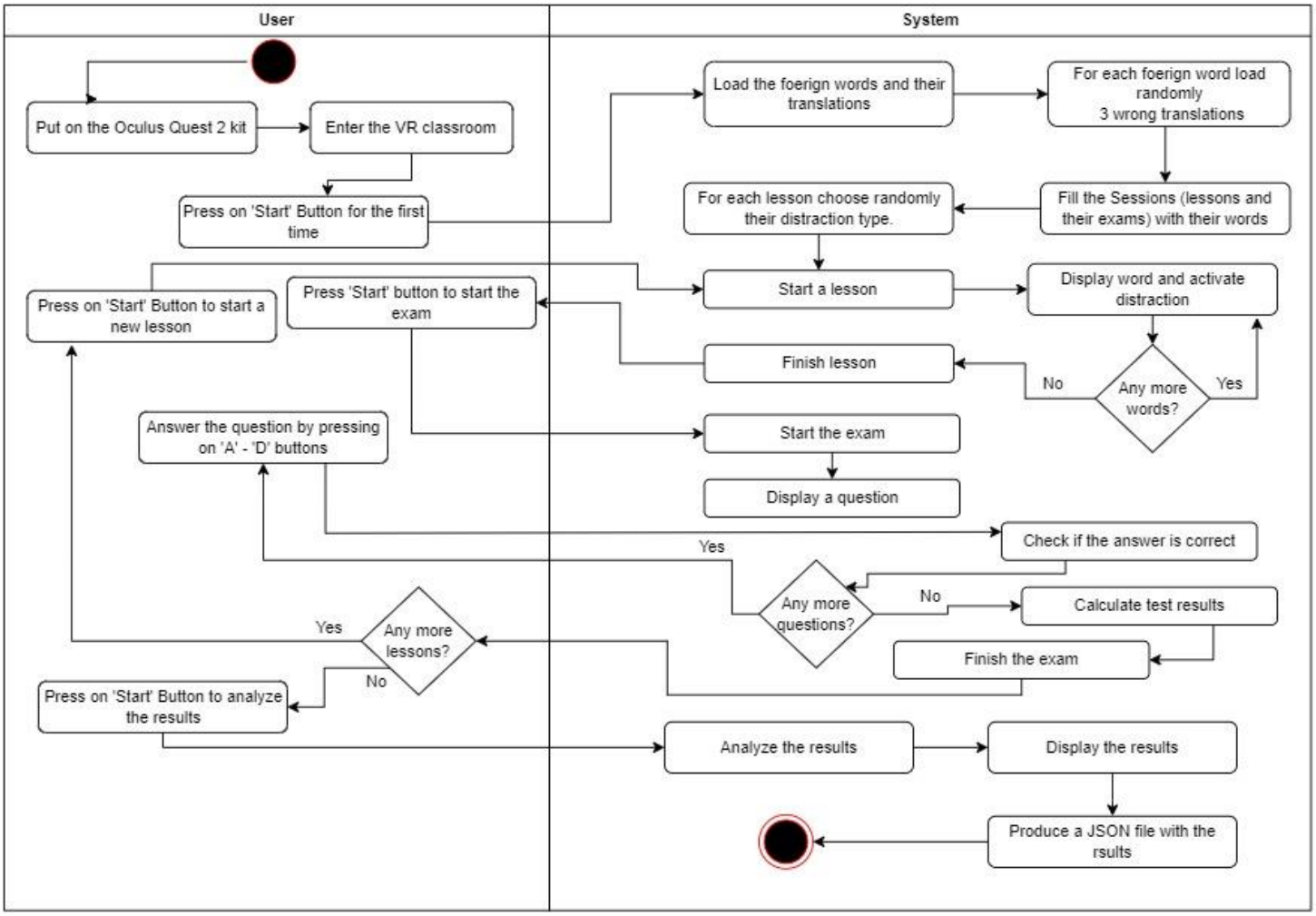

Figure 1 – Activity Diagram

The applications begins by setting the scene and rendering the Virtual Classroom designed in Unity.
Upon clicking the ‘Start’ button, the system begins with drawing random foreign words and their translations, as well as incorrect translations, from an external .csv file provided in the Resources folder of the project. Each learning session is randomly assigned one of the distraction types (Visual, Auditory, None) which will occur during the session, and each word object is assigned a distraction corresponding to the type of the lesson it appears in.

The first lesson then begins, displaying a set (configurable) number of words with their English translation. Each word stays on the board for a set (configurable) number of seconds. While each word appears, a distraction occurs, and the system monitors the user’s response – following Skip Rizzo’s example [1], if the user’s head is diverted away from the center of attention – the board – that word (and its accompanying distraction) is flagged.

At the end of each learning session, the user clicks ‘Start’ to begin an exam. The exam contains the same number of questions that were taught in the session, and each word appears in it. The correctness of each answer is recorded, and the final score calculated.

This process repeats for a total of a set (configurable) number of such learning sessions (followed by exams for each one).

Once the user had completed all lessons and exams, they may press ‘Start’ again for the results to be displayed.
A shortened version of the final recommendations is displayed on the board, while the full report is written in .JSON format to the Oculus Quest 2’s local storage, where it can be transferred to a local PC over USB if desired.

The analysis of the results, and production of the final recommendations, is inspired by Skipp Rizzo’s example video and is performed based on a combination of the final exam scores in each session as well as the proportion of words in each lesson in which the user was distracted away from the board. We consider three scenarios -

- The user passes a (configurable) threshold of words in which they were distracted and *fails* the exam. This is the most severe scenario in which the system can suggest with confidence that this type of distraction is particularly damaging to the user.
- The user passes a (configurable) threshold of words in which they were distracted and *passes* the exam. This is a more moderate scenario in which the system can suggest that this type of distraction is at least somewhat bothersome to the user.
- The user does not pass a threshold of words in which they were distracted. In this scenario, the system can suggest that this type of distraction has a negligible effect on the user. We do not consider the exam scores as we do not possess any further metrics to judge whether the score was affected by the distractions or not, beyond suggesting that the user is not bothered by complete silence.

1.1.3 Project Structure

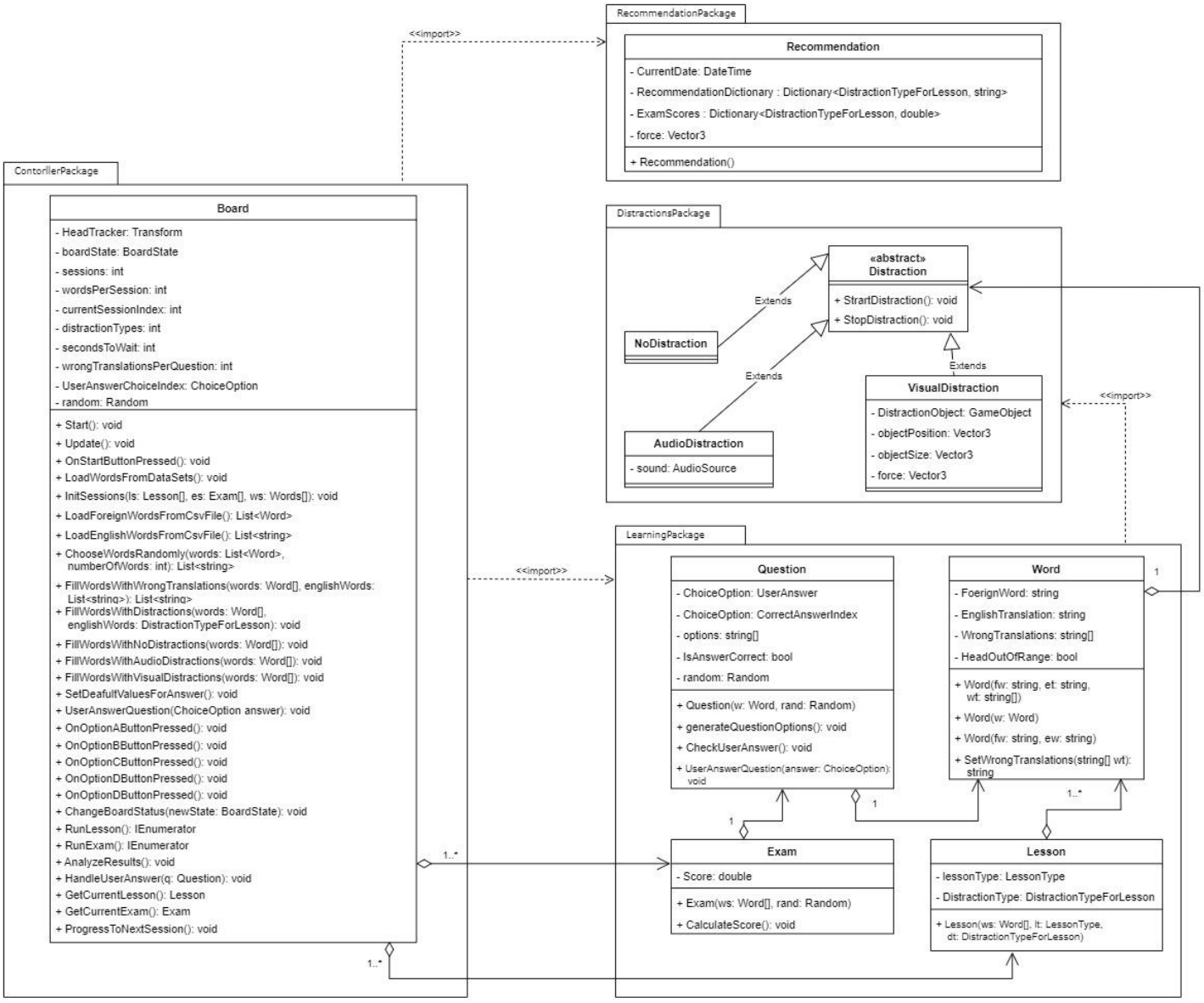

Figure 2 – Package Diagram

1.2 Process

Last semester, we begun the process of researching the subject of learning environments and how various distractions affect the learning process in different types of people. The results of this were outlined in our Project Capstone Phase A document.
This semester, we begun to implement our ideas. We would not have the sensors and monitoring equipment described in Phase A as planned, and so the scope of the project was reduced to using the exam scores and head-tracking data to provide the recommendations.
This process began with learning the basics of Unity programming, as neither of us had any experience with it – Unity is a game engine useful for creating VR environments, and also served as our IDE (integrated development environment) for this project. We acquired an Oculus Quest 2 headset through the College, and begun building our virtual classroom.

Once the classroom environment was complete, we begun to design the lessons and exams, thinking about how they should be structured and presented. We decided on a "chalkboard" in front of the user which will present words in a foreign language and their translations to English, and a table with 5 buttons – one to progress through the lessons and exam, and 4 for selecting an answer to the multiple-choice questions in the exam.

We continued by thinking on the final report, and the application's logic in how it analyzes the data and produces recommendations. The result is a combination of two elements – monitoring the user's response to distractions using the Quest 2's head tracking technology, and the results of the exam.

1.2.1 Challenges

The primary challenge throughout this process was learning the complex Unity system and VR programming, which involves a combination of building the graphical scene itself using the Unity IDE (camera, graphical and audio assets, interactive objects, etc) and scripting its behavior and logic, which was done by writing scripts in C#.
The scene building itself was the bigger challenge, since designing a fully interactive VR environment where the user is not as constrained in what they can do requires a different thought process to more 'traditional' programming we've done so far. The Unity IDE itself is also complex and required some time to familiarize ourselves with, and get all the required components that do are not included with it by default.

Another major technical challenge was dividing the work between us and working on the project concurrently.
We used a GitHub repository and integrated the Unity project with the GitHub Desktop application, but we found that GitHub had multiple issues with the project as it works better with a codebase, while our project makes heavy use of the Unity IDE to design the VR environment. The Unity IDE creates many binary files and it is more difficult to track changes made to the project, for both us and GitHub Desktop, than it would have been if the project was only code.
We only began to understand this challenge at a somewhat late stage in the development process of the project, since we were working independently for much of it.

A more theoretical challenge was understanding how to analyze the final results and produce recommendations to the user. Due to the lack of sensors we planned on having for this project, we lacked some additional metrics to use for monitoring the user response to distractions, and the correlation between those responses and the user answering a question

incorrectly in the exam. We were left with two metrics – the final exam scores and head movement tracking. Since there is little existing literature or research into the subject of head-tracking in this type of environment, we had to rely on relatively little research performed by Skipp Rizzo [1] and apply intuition to produce recommendations.

1.3. Verification and Testing

In this project we primarily tested functionality of the system, such as testing the functionality of UI buttons, that the board behaves correctly and that all components behave as expected. To do this, we employed standard acceptance testing.

To test the overall correctness and logic of our results, we performed preliminary testing on a small group of 4 users to test for sufficient variance in results and any significant biases. None of the users were diagnosed with any attention disorders. In the resulting 4 reports, we found the following exam scores –

User #1

Auditory Session – 30

Visual Session – 50

User #2

Auditory Session – 60

Visual Session – 40

User #3

Auditory Session – 30

Visual Session – 30

User #4

Auditory Session – 40

Visual Session – 70

These results provided two insights –

- We achieved a good enough degree of variance in the output recommendations and exam scores of the Auditory and Visual sessions to rule out, at least based on our extremely small sample size, a bias towards one type of distraction in the system.
- The consistent relatively low scores suggest that the language we used for these tests – German – is likely quite difficult and unfamiliar compared to other languages we could have used, or that the time for each word to appear on screen (set to 6 seconds in these tests) was too short. In fact, average scores for both distraction types are under 50 (40 and 47.5 for Auditory and Visual, respectively), suggesting that the language and properties are currently tuned to be very difficult.

In conclusion, significantly more data is needed for a statistically significant sample size to provide more conclusive results and adjust the system properties as necessary, such as finding more balanced properties to achieve healthier scores.

The acceptance testing table is provided below.

#	Test ID	Description	Expected Result	Precondition	Comments	Result
1	UserStartSystem	User presses start in the beginning.	First lessons begins.	System is running.		Pass
2	UserPressesANotInExam	User presses A while not in exam.	Nothing happens.		In every state except “exam”.	Pass
3	UserPressesBNotInExam	User presses B while not in exam.	Nothing happens.		In every state except “exam”.	Pass
4	UserPressesCNotInExam	User presses C while not in exam.	Nothing happens.		In every state except “exam”.	Pass
5	UserPressesDNotInExam	User presses D while not in exam.	Nothing happens.		In every state except “exam”.	Pass
6	UserPressesStartInLesson	User presses ‘Start’ while in lesson.	Nothing happens.			Pass
7	UserPressesStartToEndLesson	User presses ‘Start’ at the end of the lesson to start exam.	Exam begins.			Pass
8	UserPressesAWhileInExam	User presses A while in exam to answer a question.	Answer A selected. Next question is displayed.	In ‘exam’ state. Not final question.	System checks correctness.	Pass
9	UserPressesBWhileInExam	User presses B while in exam to answer a question.	Answer B selected. Next question is displayed.	In ‘exam’ state. Not final question.	System checks correctness.	Pass
10	UserPressesCWhileInExam	User presses C while in exam to answer a question.	Answer C selected. Next question is displayed.	In ‘exam’ state. Not final question.	System checks correctness.	Pass
11	UserPressesDWhileInExam	User presses D while in exam to answer a question.	Answer D selected. Next question is displayed.	In ‘exam’ state. Not final question.	System checks correctness.	Pass

12	UserPressesStartInExam	User pressed 'Start' while in exam.	Nothing happens.	In 'exam' state. Not final question.		Pass
13	UserPressesStartAtEndOfExam	User presses 'Start' at the end of an exam.	Next lesson begins.	There are more lessons to come.		Pass
14	UserPressesStartAtEndOfExam	User presses 'Start' at the end of an exam.	Routine ends and recommendations are produced.	There are no more lessons to come.		Pass
15	UserPressesAWhileInExam	User presses A while in exam to answer a question.	Answer A selected. Exam ends.	In 'exam' state. Final question.		Pass
16	UserPressesBWhileInExam	User presses B while in exam to answer a question.	Answer B selected. Exam ends.	In 'exam' state. Final question.		Pass
17	UserPressesCWhileInExam	User presses C while in exam to answer a question.	Answer C selected. Exam ends.	In 'exam' state. Final question.		Pass
18	UserPressesDWhileInExam	User presses D while in exam to answer a question.	Answer D selected. Exam ends.	In 'exam' state. Final question.		Pass

1.4. Results and Conclusions

The goals of the project were to create a system that is able to analyze the user's personal sensibilities and make suitable recommendations for their learning environment.
We feel that these goals were achieved – we created a full VR virtual classroom environment that is able to deliver lessons in a foreign language to the user while creating various distractions of different types, test their knowledge of what was taught, and produce recommendations based on their response to said distractions.

We dealt with the challenges described above as follows –

- We were able to familiarize ourselves with Unity and learn how to design the environment and program its scripting by consulting the official Unity documentation as well as a process of independent study using widely available online resources such as YouTube guides, and building the system block by block.

- For most of the semester, we did not work on the project concurrently and independently, so scheduled work around other commitments in the semester and the issues with GitHub. As work on the project progressed, we began to work together

over Skype more, rather than attempting to divide the work between us which caused issues when we tried to work on different parts of the project at the same time.

- We decided that for the results and final recommendations, we would lean on Skip Rizzo's video showing the differences between head movement responses to distractions in non-diagnosed and ADHD-diagnosed children. The material shows that the deciding factor in whether a distraction is damaging or not is whether or not the head is turned away from the center of attention – in our case, the board at the front of the classroom, which the user is facing during the lesson. We also applied our own intuition and logic to combining this metric with the final exam scores – we knew that the head tracking data tells us with a high degree of confidence whether or not a distraction was bothersome, and used the exam scores to add more granularity and an additional layer for estimating just how bothersome the distractions are.

- We also ensured as much as we can that the system is modular and flexible, so that in the future it can be easily expanded in various ways – adding distractions, changing the number of sessions or words in a session, adding new metrics for analysis if they become available, etc.

In conclusion, we feel that we worked correctly but if we had the opportunity to do something differently, we would find a different method of sharing and dividing our work than GitHub since in our experience it was unsuitable for a Unity project and forced us to schedule our work in a suboptimal way.

On a technical level, we would also spend less time trying to program functionality for the user moving through the virtual environment – for much of the semester, we had the user interacting with our application by pressing "physical" buttons laid out in the environment. This caused a lot of issues and was not necessary for what the application aims to do, and was changed later in development to a more traditional UI menu laid out in front of the user that they can interact with using the controller ray.

2. User Documentation

2.1 User's Guide

2.1.1 General Description

The Virtual Classroom is a VR application designed for the Oculus Quest 2 VR headset. The application will present you with a virtual environment modeling a classroom in which lessons are delivered in a number of sessions and exams are performed to test what you've learned. The goal of the system is to help you tailor your personal learning environment to your needs and sensibilities – during the lessons, various distractions (both visual and auditory) may appear, and the system will monitor your response to these distractions, and how they may have affected the final scores in your exams. You will then be presented with a set of personalized recommendations on what you can adjust in your personal learning environment when studying.

2.1.2 Installation Instructions

By default, applications can only be installed to the Oculus Quest 2 if they are approved and uploaded to the official Oculus Store. In order to install an external application, a process called side-loading must be performed.
The most common way to do this, and the one we recommend, is using a PC application called SideQuest. Before this can be done, however, 'Developer Mode' must be turned on in the Quest 2 – this requires an Oculus Developer Account. This process is described in the Oculus documentation [2].
Instructions on how to sideload applications using SideQuest are described in the SideQuest guide [3]. The '.apk' build file is provided with the project files.

2.1.3 Operation Instructions

Important note – the Virtual Classroom application is designed for stationary interaction, sitting down as you would in a classroom. Make sure the Oculus Guardian is set to Stationary rather than Room-Scale.

To open the application, in the Quest 2 home screen, click on the Oculus Menu button on the right controller -> Apps, and in the dropdown menu on the top right choose 'Unknown Sources'. Click on the 'VirtualClassroom' application to start it.

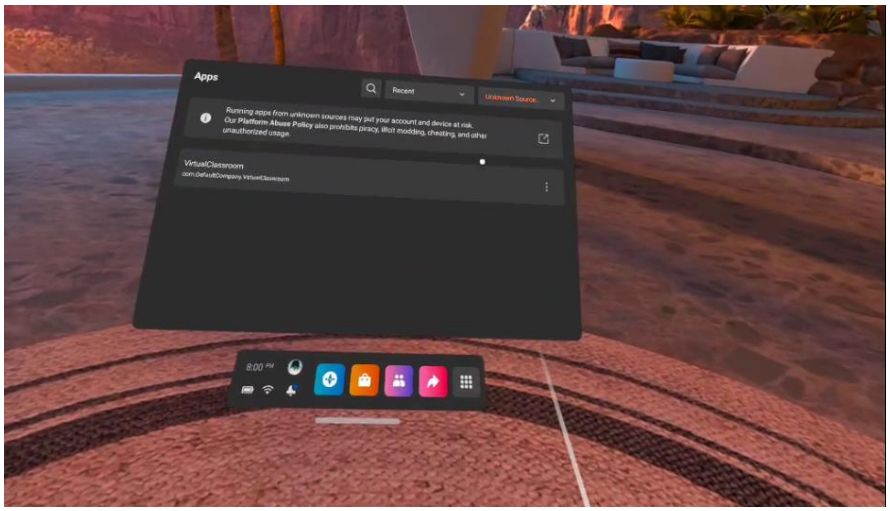

Figure 3 – Launching the Virtual Classroom

Once inside the Virtual Classroom, press 'Start' by aiming the red ray at it and pressing the trigger button (with either controller) to begin the first lesson.

Figure 4 – Starting a lesson

Pay attention and try to memorize the correct translations. During the lesson, various distractions will appear and the system will monitor your response to them.

When a lesson ends, press ‘Start’ to begin the exam. There is no time limit to the exam, but once you answer a question you cannot go back and change the answer.

Figure 5 – End of lesson, starting exam

When a question appears, select the answer you want by pointing at the corresponding button in the menu and clicking trigger on the controller.

When you’ve completed all lessons and all exams, press ‘Start’ to display your results and reocmmendations.

Figure 6 – End of all sessions, displaying results

To exit the Virtual Classroom, press the Oculus Menu button on the right controller and click ‘Quit’.

The recommendations displayed at the end are a shortened version of the full report. The full report is written to a .JSON file in the local Oculus Quest 2 storage. To access it and transfer it to a PC, connect the Quest 2 to a local PC via a USB cable, allow access to files via USB in the Quest 2 when prompted, and access it in the path –

\Quest 2\Internal shared storage\Android\data\com.DefaultCompany.VirtualClassroom\files

The .JSON report contains expanded recommendations as well as your final exam scores in each category.

2.2 Maintenance Guide

2.1.1 Code Maintenance

The project was written to be modular, highly adjustable, and easily expandable for future use and expanding the system’s features in various ways.

The ‘Board’ class in the Board script (‘Board.cs’) contains most of the project’s core functionality and logic. It is linked to the ‘Board’ GameObject in the Virtual Classroom. The class has various properties that allow us to easily adjust the properties of the system, such as the number of words taught per session, the number of distraction types, the number of choices per multiple-choice question, the time allocated to each word (and its translation) appearing on the board during the lesson, etc.

The system pulls foreign words, as well as their correct and random incorrect English translations from ‘.csv’ (comma separated values) files contained in the project build ‘Resources’ folder. Any language can be used – simply replace the existing .csv files in the folder with new ones and update the ‘LoadForeignWordsFromCsvFile’ and ‘LoadForeignWordsFromCsvFile’ functions with the names of the new files.

The existing code contains three types of distractions – Visual, Auditory, and None. This can be expanded by adding new types to the ‘DistractionTypeForLesson’ enumerator in the Board class and creating new distraction classes that inherit from the abstract parent class ‘Distraction’.

Result analysis is performed in the ‘AnalyzeResults’ function. The scope of this function can be expanded if new monitoring methods (such as various sensors) are added to the existing head-tracking data and exam scores. The threshold for determining whether the user was distracted by the distraction type during the lesson or not is set to half the number of words per session in our code and can be changed by changing the ‘distractionThreshold’ variable in the function.

2.1.2 Installation and IDE instructions

The project uses and requires Unity version 2020.3.21f1 (LTS) and Visual Studio 2019. This version of Unity must be installed, and the project imported. All graphical, audio, and data assets are included in the project files, as well as any additional packages used by the project.

The application itself runs on Oculus Quest 2 VR headset.

The project uses XR Interaction Toolkit package. It is included with the project and does not need to be installed separately. As part of it, there is an inactive XR Device Simulator GameObject in the scene. The XR Device Simulator provides keyboard bindings for movement within the scene and allows the developer to simulate a run within the editor rather than on the Quest 2, making testing significantly faster and more efficient. It can reactivated whenever that is necessary, but must be deactivated when the project is ran on the Quest 2 as it interferes with it.

3. References

[1] Head-tracking in a Virtual Classroom to assess ADHD
https://www.youtube.com/watch?v=BQyO3oDMKbI&ab_channel=SkipRizzo

[2] Device Setup – Oculus Documentation
https://developer.oculus.com/documentation/native/android/mobile-device-setup/

[3] GUIDE – How to Sideload Content on Oculus Quest using SideQuest
https://uploadvr.com/sideloading-quest-how-to/