Metaverse Classroom for Virtual Learning

Group 2

Furkan Tas
393340
Technische Universität Berlin
M.Sc. Computer Science
Berlin, Germany
furkan.tas@campus.tu-berlin.de

João Lucas Mendes de Lemos Lins
0461641
Technische Universität Berlin
M.Sc. Computer Science
Berlin, Germany
althis@campus.tu-berlin.de

Jacopo Ceravolo
0488030
Technische Universität Berlin
M.Sc. Computer Science
Berlin, Germany
j.ceravolo@campus.tu-berlin.de

Tarik Abdel-Moati Moussa Salama
0380833
Technische Universität Berlin
M.Sc. Computer Science
Berlin, Germany
tarik.moussa@tu-berlin.de

***Abstract*—The COVID-19 pandemic has forced a shift towards online learning, which has resulted in challenges for student engagement and participation. This project aims to alleviate these issues by leveraging metaverse technology. Specifically, we present a digital classroom built using Unity3D that incorporates features to promote physicality and increase engagement, including virtual avatars, lip-syncing, and virtual reality integration. Our approach represents a novel solution to the problems facing online classrooms and has the potential to significantly enhance the overall learning experience.**

I. Introduction

A. Motivations

The COVID-19 pandemic forced a transition from in-person to online learning. While online conferencing has benefits, such as saving time and enabling participation from anywhere, it also has limitations like lack of feedback and immersion. We propose a metaverse-based solution to address these limitations and enhance the virtual classroom experience by providing more opportunities for interaction and engagement. We review the limitations of traditional online conferencing platforms and describe the requirements and system specifications of our proposed virtual classroom system.

B. Limitations of Traditional Online Conferencing Platforms

Online conferencing platforms such as Zoom, Microsoft Teams, and Google Meet have revolutionized the way education is delivered by enabling remote learning and teaching. However, these platforms also have some limitations that can negatively impact the quality of the educational experience. In this section, we will discuss two major limitations of online conferencing platforms: the lack of feedback and the lack of immersion.

Feedback is a critical component of effective teaching and learning, as it enables students to understand their progress, strengths, and weaknesses. In online conferencing, feedback can be limited or distorted, leading to a demotivating and disorienting experience for both students and educators. Studies [1] [2] found that online teaching can hinder effective feedback due to the lack of immediate and direct responses, resulting in lower engagement and performance. Therefore, the lack of feedback can lead to a suboptimal learning experience in virtual classrooms.

Another limitation of online conferencing is the lack of immersion and sense of presence that can be found in traditional face-to-face classes. Immersion can enhance learning outcomes by increasing attention, motivation, and memory retention, and can be achieved through various means such as sensory stimuli, interactivity, and social presence. However, the lack of physical co-presence and sensory stimuli in online conferencing may reduce the level of immersion and presence, leading to less engagement and participation. Research [3] noted that online learning can be less effective than traditional face-to-face learning due to the lack of social presence and the inability to read and interpret nonverbal cues. Therefore, the lack of immersion can negatively impact student motivation, learning outcomes, and overall educational success.

In conclusion, while online conferencing platforms have enabled remote learning and teaching, they also have limitations that can negatively impact the quality of the educational experience. A metaverse-based solution can provide a more immersive and engaging virtual learning experience that supports feedback and collaboration, which can enhance the learning experience.

C. Use Case and System Requirements

Our virtual classroom platform aims to create an immersive online learning experience that closely resembles a physical

classroom. To achieve this, the platform must have specific system requirements, such as a robust video conferencing system, collaboration tools, and a virtual whiteboard for real-time presentations. The primary use case for the platform is to provide students with an engaging and interactive platform for learning together, interacting with teachers, and collaborating on assignments. The platform will include features like real-time chat, hand-raising, note-taking, document sharing, and customizable avatars. These features will foster a sense of community and enable students and teachers to interact in real-time.

II. General Specifications

A. Design

The design of the metaverse classroom system has been thoughtfully developed to provide an immersive and engaging learning environment for both teachers and students. The goal is to utilize the advantages of 3D virtual spaces to create a unique and effective way of delivering educational content.

Upon starting up the system, a log-in prompt appears, where the user must input their credentials to access the classroom. The roles are then differentiated based on the provided credentials, distinguishing between students and teachers.

Once a student logs in, they are randomly assigned a seat in the classroom based on the number of participants. The student's position is fixed at their seat, with only the ability to move their upper body. The first-person camera allows them to view up to a 270-degree angle, providing an opportunity to engage and interact with their peers sitting next to them.

The student's frontal view displays the teacher, along with the screen displaying the lecture, which is presented as an embedded 2D web application in the 3D environment. A personal UI, also a 2D screen placed on the 3D desk, is accessible to the student, enabling them to view PDFs, take notes, and perform other activities that facilitate learning. Additionally, a button with a hand painted on it is placed next to the student's UI. When pressed, it raises the student's hand to ask a question to the teacher. The animation of the student raising their hand is visible to everyone, including the teacher and every other student in the classroom. Once the question is asked, the button can be pressed again to lower the hand.

Upon the teacher logging in, their avatar's first-person camera is activated, facing the students. The teacher can move around the room freely, providing a unique and interactive experience. By default, the teacher's position is with their back against the blackboard where the teacher UI is located, which enables them to present their classroom materials to the students.

Both students and teachers possess motion tracking and lip-synching functionalities. This ensures seamless and natural communication, further enhancing the immersive experience for all participants.

B. Implementation

The metaverse classroom has been developed using a combination of Unity and C# programming language. The system consists of two scenes, one for the log-in process and one for the classroom environment. The main menu of the system is a canvas overlayed on the 3D world, providing an intuitive interface for users.

The log-in panel is composed of two text boxes and a button, with a password input text field. Once the user inputs their credentials and presses the button, a preventive check is performed to ensure the login conditions are satisfied. If the credentials are correct, a new scene is loaded.

To pass the user's login information to the new scene, a static script component attached to an empty GameObject present in both scenes is used. The UnityEngine.SceneManagement library is then used to load the new scene, which presents the user with the 3D classroom environment.

If the user logging in is a teacher, all other cameras except theirs will be deactivated for the current instance. Similarly, if a student logs in, a camera reserved for students will be randomly activated. The students' interactions with the classroom are limited to using their personal UI and the "raise hand" button, which triggers the AskingQuestion animation state. Each student is controlled by the SittingAnimatorController.

The teacher has more freedom of movement and can move around the room using standard keys (W, A, S, D) and their first-person camera with the mouse pointer. The teacher's movement direction is calculated using a combination of pressed keys and the Quaternion object applied to the camera position. Pressing the keys also triggers a walking animation, while running is possible by pressing the LeftShift key. The teacher's movement is enhanced by gravity, allowing them to walk up and down stairs.

All animations used in the system have been downloaded from Mixamo, an Adobe online animation service that provides pre-made motion-capture animations. The animations have been retargeted to work with VRM and Humanoid avatars.

III. Motion Tracking & Lip Syncing

A. Introduction

Human beings are wired to be highly attuned to facial expressions and body language as a means of interpreting and understanding the emotions and intentions of others [4]. Therefore we consider facial tracking and lip syncing in real-time to be essential elements in creating a realistic virtual experience. They allow for avatars to mimic the natural movements and expressions of a human face, enhancing the overall sense of immersion for the user. In the following subsections the structure will follow the different solutions for Unity we came in contact with during our implementation. We will also briefly evaluate our final implementation and give a outlook on the technology as a whole.

B. Facial Tracking - Unity Live Capture

In our search the software solutions that was represented on the internet very prominently was Live Capture [5] from the Unity developers themselves. The Live Capture package works

by creating a server on which the companion iOS app Unity Face Capture connects to. The companion app utilizes Apple ARKit's Face Tracking and uses the camera of the mobile device to track the users face and to send the results to the host which runs the Unity application. The Unity application then can use the data to apply the movement to the face and head of the virtual avatar. s To learn how to use Live Capture, we followed a YouTube tutorial [6] which also showed us how to use Live Capture in combination with a Ready Player Me avatar. Although we were really pleased with the accuracy of Live Capture/Apple's ARKit, this feature was not suitable for our project's requirements, as it was only compatible with Apple iPhone/iPad devices. We needed a more cross-platform solution that could work with a regular webcam or at least also supported Android devices.

C. Facial Tracking - OpenSeeFace

1) VTubers: Since we could not use Unity's Live Capture package we searched for other solutions. In our research we stumbled across VTubers, a phenomenon of online entertainers who use digital avatars while streaming to their audience [7]. VTubers are an interesting case study for a metaverse setting because they demonstrate how people can create and express themselves as well as connect with virtual characters in immersive and interactive ways in real-time.

More importantly they typically do not use specialized hardware but instead use regular webcams and microphones with a puppeteering program called VSeeFace. VSeeFace uses webcam or smartphone sensors to track the user's facial expressions and gestures, and then maps them onto a 3D model that can be customized with various accessories and effects. VSeeFace also supports lip sync, eye blinking, eye tracking, head rotation, body movement, physics simulation and more. [8], [9]

Despite the fact that VSeeFace is closed source, it is based on an open-source software called OpenSeeFace [8], [10].

2) OpenSeeFace: OpenSeeFace [10] is a Python library that provides robust real-time face and facial tracking library based on MobileNetV3, an efficient convolutional neural network model. The model can be used standalone using the facetracker.py script file which accesses the webcam and returns tracking data. It also comes comes with accompanying Unity components [10]:

- OpenSee.cs: This is the main script that communicates with the OpenSeeFace tracker and receives face data from the facetracker.py using UDP. It also provides some helper functions for other scripts
- OpenSeeIKTarget.cs: This script can be used to create head and eye targets for inverse kinematics (IK) systems
- OpenSeeVRMDriver.cs: This script applies the face data to a VRM avatar using blend shapes and bones. It also supports jaw bone animation

OpenSeeFaces uses the VRM format, which is a file format for 3D humanoid avatars (3D models) intended for VR applications. The VRM format can store information such as textures, bones, blend shapes, materials, spring bones and first-person settings for 3D avatars. The VRM format can be used in Unity with an open source standard implementation called UniVRM. UniVRM can import and export VRM files in Unity and provide various functions such as runtime loading, animation control and customization. [10]–[12]

3) Implementation: In order to integrate OpenSeeFace into our classroom environment, we wanted to replace the VRM avatar used in the sample provided by the OpenSeeFace repository. To achieve this, we examined how the sample worked and identified the key components required for the avatar tracking. The tracking process involves running the facetracker.py script to gather facial data, which is then transmitted to an OpenSee component. The OpenSeeVRMDriver is then used to control the VRM avatar. Using these components it is already possible to have facial tracking.

However, to enable the avatar to track the head movements of the person behind the camera, a game object exists that moves and rotates based on the person's head movements. This was achieved using the OpenSeeIKTarget script. The avatar's head, neck, and shoulders were animated using an inverse kinematic bone chain that was attached to the avatar's head using the FastIK plugin. As a result, when the person behind the camera moves their head, the IK target moves, and the avatar's head, neck, and shoulders move accordingly.

Initially, we were unsure how the IK bone chain was attached to the avatar. We discovered that the animation rigging extension needed to be enabled to visualize all the available bones in the model. Once enabled, we quickly identified how the IK bone chain was attached to the avatar.

4) Creating a VRM Avatar: Initially, we found that the default avatar in our sample did not fit into a realistic classroom setting. We then searched the internet for available VRM models and discovered that nearly all of them were designed for fictional settings, which would not be suitable for our purposes. As a result, we decided to create our own VRM model.

One option we considered was using VRoidStudio [13] to create a VRM model from scratch. However, we decided to explore the process of converting a Ready Player Me avatar to VRM format instead.

To convert the Ready Player Me avatar into VRM format, we followed two different video tutorials that complemented each other [14], [15]. The conversion process involved several steps, which we summarize below:

1) Using Blender, we converted the GLB file format to FBX format with specific settings.
2) We imported the FBX file into Unity using UniVRM and changed the rig to humanoid.
3) We re-applied the correct materials and textures to the model in Unity.
4) We set the blend shape in order to allow movement of the mouth and eyes.
5) Finally, we exported the model to VRM format.

D. Lip Synching

Lip Synching in our context is the process of creating realistic mouth movements for 2D or 3D characters based on audio input. The software that is commonly used for lip synching in Unity is Oculus Lipsync. Oculus Lipsync is a plugin that provides high-quality lip sync using deep learning technology. Oculus Lipsync can analyze audio input from a microphone, an audio file or a text-to-speech system and generate corresponding visemes (visual representations of phonemes) for the character's mouth. [16]

To implement Oculus Lipsync in our project, we installed the plugin, but encountered issues when attempting to use it with a VRM model. In order to resolve this, we searched for examples of lip synching with VRM models and found a MIT-licensed GitHub repository [17]. Similar to our integration of OpenSeeFace, we first analyzed how the sample worked and then attempted to replicate what we learned in our own project. However, we discovered that we could not use the lip synching and face animation simultaneously using this approach, as the OpenSeeFace scripts overwrote the mouth movements. Fortunately, we were able to resolve the issue by setting the audio source object to our new MicrophoneInputSource object within the OpenSeeVRMDriver object, which had the OVRLipSyncContext attached to it.

E. Evaluation

Lip synching and facial tracking are challenging tasks that require accurate and realistic animation of the mouth and face movements according to the speech and expression of the user. Therefore, it is difficult to estimate the performance and accuracy of animations.

However, we noticed some discrepancies between the mouth movements and sound the sounds we spoke into the microphone. The facial tracking still has room for improvement. Particularly, the movements of the eyebrows and facial muscles were not as noticeable as we expected, making it difficult to convey emotions effectively. It's plausible that these limitations may be attributed to the insufficient blend shapes available for facial animation. It's worth noting that despite these challenges, we were satisfied with the performance of the machine learning models, namely OpenSeeFace and OVR LipSynch.

F. Outlook

As virtual reality and the metaverse continue to gain popularity, it is becoming increasingly important for avatars to accurately mimic natural human movements and expressions in real-time, in order to provide users with a more immersive experience.

To achieve this, we propose focusing on improving the mapping of machine learning model results onto avatars in a simpler and more effective manner. In our experience, it was easier to find a model that accurately tracks facial landmarks than to accurately map it onto an avatar. We believe that an essential step towards this goal would be to establish a standardized format for avatar models/animations like VRM that also supports advanced techniques for facial animation.

IV. Virtual Reality

In this section, we will discuss the reasoning for pushing for Virtual Reality in this project, detail the design decisions made to achieve those goals, and detail a bit more about the implementation. Finally, we will discuss other features which we wish we could have implemented, but didn't have time for and leave them as suggestions for future work. One of our main motivations for starting this project was the lack of engagement in flat virtual classes. Elements of interactivity are essential for engagement. What is left then, are two options; either change the whole classroom structure to fit the medium (challenge this that comes it its own associated costs and issues), or change the medium to enable some further element of physicality, while mainly remaining a digital setting. Our group then decided the Metaverse was a good fit for this direction.

The two main advantages of VR in this aspect are the following:

- VR moves the student out of whatever reality they inhabit physically, to a new environment under the control of the professor, thus re-establishing control of the environment to the teacher and removing extraneous distractions.
- VR enables the return of physicality to the classroom environment, by enabling professors and students to interact in ways much closer to the regular classroom experience they've grown accustomed to.

Not only that, but VR actually enables us to *augment* some of this expression in new directions, by providing a new layer of interaction that is impossible in physical environments. Through the use of 3D modelled avatars, the loading of new classroom environments and 3D models of the subjects being studied, and other features such as 3D drawing, both professors and students have more avenues with which to express themselves, which has been shown to be associated with better learning outcomes [18].

A. Design

While there were many things we'd like to explore with VR, due to time constraints we had to choose the most balance the impact of the features in our list with the time they'd take to implement. We've thus listed all of our features and created a timeline for development.

We've decided that some features were essential for the VR experience, and those were pushed to the top of the list. Those were a first-person camera, locomotion, and object interaction.

We knew that camera control was the most basic aspect that needed to be possible, and that itself was rather unproblematic. After that, we chose to focus on locomotion, by implementing both standard methods adopted by the industry: Joystick movement and teleportation. After that, we focused on interaction with scene objects since our thesis was that the problem with flat classrooms was the lack of interaction and that more interaction would lead to greater engagement. We

implemented object manipulation with some scene objects, but there is much more that can be done which we will touch on in the future work section.

Once the essentials were properly blocked into our schedule, we started looking for other features that could be implemented in the remaining time frame. Of those, we chose to focus on 3D drawing and a few gestural controls, such as raising one's hand creating triggering the raise hand feature and the handing of papers to students being possible with a throwing gesture. Eventually, due to complications in the integration step, the gestures were scrapped, and we concluded the project with only 3D drawing as an extra feature included other than the basic locomotion and interaction.

B. Implementation

Our efforts began with finding training materials for modern VR development. While one of our team members had experience working in VR, his experience was mostly with the beginning of VR back in 2016, and that experience wasn't of much use since the field changes so quickly. Initially, we looked to Youtube and Medium tutorials for training materials, but that turned out not to be a good idea because VR changes so quickly in the time span of fewer than 6 months most of these tutorials were basically useless. After a while, we found a 36-hour digital course provided by Unity itself[19], which was kept mostly up to date with the current development practices. (As of the writing of this report, however, about 1/3rd of the course has been removed from the page, possibly for revisions and updates)

In the 4 years since the inception of VR, most of the code that originally had to be written by hand has now been standardized and pre-packaged in easy-to-use plugins. In special, the Unity tutorial we followed suggested we use the OpenXR plugin for Unity, and it comes with most of the functions for camera control and locomotion already pre-packaged under the XR-Origin prefab. Not only that, but it also provides a single interface layer that removes most of the problems in working with different models from different manufacturers and unifies these under a minimal model. With this feature, the only reason to interact with the drivers of your target headset is if you need to use one particular feature not covered by OpenXRs generic model.

Other than that we also used the XR Interaction Toolkit plugin, which simplifies the task of capturing input from OpenXR's interaction model and turning them into "Interactor" and "Interactable" components that can easily be attached to scene objects. On this front, the Unity tutorial also provided a handy "defaultActions" profile that can be loaded into Unity preferences that removes the tedious task of connecting every input event to every component action in the Interaction Toolkit, and that was quite handy.

All that was left to do then was set up the connections between the XR-Origin prefab and the camera and the hand models, and then add the interaction toolkit's components to the hand model and connect the component actions to the input events. And just like that, we had a full camera and locomotion.

As mentioned previously, the OpenXR plugin already comes with both teleportation and analogue locomotion ready out of the box, and one can switch between these in the settings of the XR-Origin component and the controller components. For teleportation, we can choose between free teleport and target teleportation by adding the "teleportation provider" component to any object in the scene. Interestingly, the plugin even comes with the option to use fade in/fade out during teleport pre-enabled in its settings, as it has been shown to help with simulator sickness[20].

The plugins also provide some basic elements necessary for object manipulation, such as line raycasters and sockets for manipulable objects. The former is just a line-renderer, but it is handy that it already comes pre-connected with hand input events. The latter is more important. When developing VR hands, one has two options for handling collisions; one may add physics colliders to the hands, which means when the hands collide with something they will lose synchronicity with the physical hands; or one might just let hands phase through objects while maintaining sync with the real world controllers. Both options make it very challenging to grab and hold scene objects, and the sockets solve this by making objects snap to the hands of the player upon a button event happening inside a certain radius of the object.

Finally, the one thing the XR Plugin did not have that was part of our design was 3D drawing, but luckily for us, this is actually part of the Create with VR course. While it did not teach you how to create the feature itself, it gave files for such as part of the challenges for Module 2. In this case, they gave us broken files, and we then had to use what we learned in the module to fix them and made the feature work properly. Through these files, we could have a look at the logic for 3D drawing, and understand how it is done. Basically, we use a raycaster in the right hand to capture 3D positions in every frame when the trigger is pushed, and these are added to an object with a line-renderer and also a "brush" tag. We then use the tag to locate the objects in the hierarchy whenever we need to change or delete them.

As previously stated, the greatest challenges with VR were the many problems with version incompatibility. While developing our project unity updated a few times, such that we started development in version 2021.3.13f1, and finished in 2021.3.16f1, with most of our project in 2021.3.16f1. Even such tiny version differences completely broke parts of our application. The reason for moving forward in the version history was because .16f1 was the version most of the tutorials were made with, and when trying to import the project into 13f1 none of the input tracking worked properly. Not only that, but downgrading the plugin itself caused its own problems, since when we tried downgrading just 2 versions prior the interaction model was completely different, using an XR-Rig object, instead of a unified XR-Origin.

C. Outlook

Due to time constraints, we had to severely reduce the amount of interactable objects in the classroom scene. The main problem is that for every prefab, the interaction socked needs to be added one by one, to make sure the object's position matches what is expected of the user when they hold them. This is a time-consuming task, especially when a scene has so many objects you'd like to interact with as ours has.

In our report, we mentioned previously a need to really think about the interaction opportunities an object provides. Because of that, instead of spending a lot of time on providing the minimal expected interaction to a ton of objects in our scene, we instead focused on providing one interaction model that would best exemplify what we meant by "augmenting the classroom". For us, this object was the pen, which enabled the user to draw in 3D in a magical way that is only possible in VR. Given more time, we'd like to apply this design philosophy to more objects in the scene, such as water bottles, paper and chairs. We believe there is more to be done with these if we are willing to go past the "basic interaction model" that can be expected of them.

We would also advocate for the benefits of gesture interactions. Through our tests, we found that where VR shines is not in being real, but in providing something "more" than reality, and that is perfectly exemplified through the use of gestures. Gestures like throwing study material to the audience and having it show up on their HUD, or raising our hand and having an icon appear above you transform the classroom into something more than real and highlight the intrinsic desires of the actions underneath. This power of expression, augments the inner desires of the users and makes them visible, physical, and interactable. These, we believe, are where the strengths of VR lie.

V. Note-taking and information sharing functionality

A. Introduction

In this section, we describe the development and implementation of note-taking and information-sharing functionality in a Unity VR lecture. The goal was to provide an immersive and interactive learning experience for users, allowing them to take notes and share information within the VR environment. To achieve this, we conducted a review of related work in this area, which provided insights into existing approaches for integrating note-taking and information sharing in virtual environments. Based on this review, we designed and developed a solution that integrated a web application for note-taking and information sharing into the virtual environment using UnityWebBrowser from Voltstro-Studios [21]. In this section, we provide details on the design and implementation of this solution, as well as an evaluation of its effectiveness.

B. Related Work

Various approaches to integrating note-taking and information sharing in virtual environments have been explored in the past. Some of the relevant works include:

- "Edu VR: Design and Implementation of Virtual Classroom Environment in VR for Remote Learning"[22]
  This study describes the development and implementation of a virtual learning environment called "Edu VR" that provides an immersive experience for students and teachers, enabling real-time interactive collaboration and communication. The virtual environment features an integrated note-taking and drawing function that allows users to create and share notes and drawings in a virtual environment.
- "iVRNote: Design, Creation and Evaluation of an Interactive Note-Taking Interface for Study and Reflection in VR Learning Environments"[23]
  This study describes the development and evaluation of an interactive note-taking interface called "iVRNote". The interface is designed for use in virtual learning environments, allowing users to take notes and organize content in an immersive environment. Users can also incorporate interactive elements into their notes and share their notes with other users.
- "A Collaborative VR Visualization Environment for Offshore Engineering Projects"[24]
  This study describes the development and implementation of a collaborative virtual environment for offshore engineering projects. The virtual environment enables interactive collaboration between engineers and provides an immersive visualization of offshore structures. Users can add notes and comments to the visualized structures and share them in real-time.

In the context of developing a virtual lecture in Unity, note-taking and information sharing functionality was implemented through a web application integrated into a World Space UI. Using a web application allowed for a flexible and customizable user interface that enabled users to create and save interactive notes and images. Integration of a web application in Unity was achieved using the external plugin UnityWebBrowser from Voltstro-Studios. This allowed for embedding a custom web interface within a Unity scene, which was ideal for the note-taking and information sharing functionality in the virtual lecture.

C. Design & Concept

The goal of the note-taking and information-sharing functionality was to provide an immersive and interactive learning experience for users, allowing them to take notes and share information within the virtual environment. To achieve this, we designed a solution that would integrate a web application for note-taking and information sharing into the virtual environment.

The solution consisted of two main components: the World Space UI and the web application. The World Space UI provided an interface for users to interact with the virtual environment, while the web application provided a platform for note-taking and information sharing.

To design the World Space UI, we used the Unity UI system and created a custom UI that was integrated into the virtual

environment. The UI consisted of two main components: a table for note-taking and a blackboard for information sharing. The table allowed users to create and save notes in a virtual notepad, while the blackboard allowed users to share information and graphics with other users in the virtual environment.

For the note-taking feature, we designed a simple interface that allowed users to easily create, edit, and delete notes. The virtual notepad on the table provided a familiar and intuitive interface for users to take notes. Users could easily add text, draw diagrams, or upload images into the virtual notepad.

For the information-sharing feature, we designed a blackboard that allowed users to share information and graphics with other users in the virtual environment. Users could upload images or other visual content onto the blackboard and share them with other users. To make the interface more interactive, we added the ability to draw or annotate on the blackboard.

To implement the web application, we used a modified version of the webviewer from pdftron [25]library and Next.js framework[26] to create a custom web interface. The web application allowed users to take notes, create diagrams, and upload images within the virtual environment. The web application was embedded into the World Space UI using UnityWebBrowser from Voltstro-Studios, allowing users to seamlessly switch between the virtual environment and the web application.

Overall, the design and concept of the note-taking and information-sharing functionality was focused on providing a user-friendly and intuitive experience for the professor who would evaluate the group project. By integrating a custom UI and web application, we were able to provide users with a flexible and customizable platform for taking notes and sharing information within the virtual environment. We also ensured that the interface was easy to use and intuitive for users. The storage and sharing of the notes and information was not yet defined.

D. Implementation

We implemented the note-taking and information-sharing functionality using a combination of Unity, C#, and web development technologies. The World Space UI was created using Unity's UI system, and the web application was built using the pdftron library and Next.js framework. The World Space UI was integrated into the virtual environment and consisted of a table for note-taking and a blackboard for information sharing, both of which could be interacted with using a combination of mouse and keyboard input.

We embedded the web application into the World Space UI using the UnityWebBrowser plug-in from Voltstro-Studios. The web application allowed users to take notes, create diagrams, and upload images within the virtual environment. For note-taking, we created a virtual notepad on the table that allowed users to easily create, edit, and delete notes. For information-sharing, we designed a blackboard that enabled users to share information and graphics with other users in the virtual environment.

Our implementation of the note-taking and information-sharing functionality aimed to provide a seamless and intuitive experience for users. By combining the World Space UI and web application, we were able to offer users a flexible and customizable platform for taking notes and sharing information within the virtual environment.

The UnityWebBrowser plug-in is a useful tool that enables the integration of web content into Unity. It uses the Chromium Embedded Framework (CEF) as the web engine to display web content in Unity. However, to use the UWB plug-in on macOS (M1), two steps are required. First,the engine must be recompiled for macOS Intel. Second, the CEF version in the package must be changed to the macOS architecture. Although it's worth noting that Intel applications can be emulated on macOS (M1) via Rosetta 2.

In step 1, the engine must be recompiled for macOS Intel, since macOS (M1) does not support running code compiled for Intel-based CPUs. This allows the engine to run at the Intel compatibility level on macOS (M1). In step 2, the CEF version in the package must be changed to the one that is compatible with macOS. Since CEF supports specific architectures and operating systems, the CEF version in the package must be changed from UWB to one that is compatible with macOS. Changing the CEF version in the package to the macOS version allows the engine to run properly on macOS (M1).

After completing these steps, the modified UWB plug-in can be used on macOS (M1) to integrate a custom web UI into a Unity scene.

VI. Conclusion

The global pandemic has caused a widespread shift towards digital settings, including classrooms. While this change has been disruptive, it has also presented new opportunities and possibilities for educators and learners worldwide. Digital classrooms have become a permanent fixture, necessitating a re-evaluation of traditional teaching methods and pedagogical practices. In this paper, we propose that settling for easy solutions in school settings is insufficient and that addressing important aspects of a physical classroom can lead to significant improvements in the digital learning experience.

The present study aimed to create a digital classroom that leverages Metaverse features to attenuate the challenges associated with the current flat digital classroom setting. Using state-of-the-art plugins for the Unity3D Engine, we created a 3D environment that included avatar creation and browser integration. We also incorporated virtual reality technology to enhance immersion and enable desirable interactions that are impossible in real-life classrooms. Our objective was to create an engaging and immersive digital classroom that outperforms current flat digital classroom options such as Zoom, Google Meet, and Microsoft Teams and we believe that we succeeded at that.

It is worth noting that standardized testing was outside the scope of this project, and our opinions are based solely on our experience. We invite readers to download our project from our repository to evaluate its effectiveness for themselves.

Our repository includes a compiled classroom with a licensed 3D Classroom asset, which we used in all video material for this project. Additionally, the codebase for the same project is available, with a free classroom model replacing the licensed one.

Overall, this project serves as a starting point for future research on the potential benefits of metaverse technology in digital learning environments. We believe that further investigation into the use of VR and other emerging technologies in education can provide a richer, more engaging, and more effective learning experience for students.

REFERENCES

[1] S. Turker, S. Poyrazli, and L. Rodriguez, "Examining the effectiveness of online learning and its impact on cultural competence," *Journal of International Students*, vol. 10, no. 1, pp. 26–46, 2020.

[2] C.-L. Lai, G.-J. Hwang, and J.-C. Liang, "A review of using eye-tracking technology in online learning," *Journal of Educational Technology Society*, vol. 23, no. 3, pp. 97–109, 2020.

[3] F. Biocca, C. Harms, and J. Gregg, "Communication in the age of virtual reality," *Psychology Press*, vol. 15, no. 2, pp. 213–225, 2003. DOI: 10.1207/s15327876mp1502_7.

[4] L. E. Ishii, J. C. Nellis, K. D. Boahene, P. Byrne, and M. Ishii, "The importance and psychology of facial expression," *Otolaryngologic Clinics of North America*, vol. 51, no. 6, pp. 1011–1017, Dec. 1, 2018, ISSN: 0030-6665, 1557-8259. DOI: 10.1016/j.otc.2018.07.001. [Online]. Available: https://www.oto.theclinics.com/article/S0030-6665(18)30121-X/fulltext (visited on Feb. 26, 2023).

[5] "About live capture." (), [Online]. Available: https://docs.unity3d.com/Packages/com.unity.live-capture@1.1/manual/index.html (visited on Feb. 26, 2023).

[6] "Ready player me x unity live capture / face capture app." (), [Online]. Available: https://www.youtube.com/watch?v=d1JhZooDHeM (visited on Feb. 26, 2023).

[7] "What is a vtuber? - how-to geek." (Apr. 18, 2021), [Online]. Available: https://www.howtogeek.com/720841/what-is-a-vtuber/ (visited on Dec. 30, 2022).

[8] "Vseeface." (), [Online]. Available: https://www.vseeface.icu/ (visited on Dec. 30, 2022).

[9] "Vseeface - vroid hub." (), [Online]. Available: https://hub.vroid.com/en/apps/zf1qIB2ExGNNOEb3OEBT6uDghdwS6k7NSDwB5DbcyCE (visited on Dec. 30, 2022).

[10] "Github - emilianavt/openseeface." (), [Online]. Available: https://github.com/emilianavt/OpenSeeFace (visited on Dec. 30, 2022).

[11] VRM Consortium. "Vrm consortium." (2023), [Online]. Available: https://vrm-consortium.org/en/ (visited on Feb. 26, 2023).

[12] VRM Consortium. "Vrm documentation." (2023), [Online]. Available: https://vrm.dev/en/ (visited on Feb. 26, 2023).

[13] "VRoid studio." (Mar. 2, 2023), [Online]. Available: https://vroid.com/en/studio (visited on Mar. 5, 2023).

[14] UNION AVATARS, *How to convert GLB to VRM (updated 2022)*, Oct. 20, 2022. [Online]. Available: https://www.youtube.com/watch?v=BXONTLXFCHk (visited on Mar. 5, 2023).

[15] Blazing Crafts Studio, *Convert ready player me avatar for use in VSeeFace & vup*, Aug. 29, 2021. [Online]. Available: https://www.youtube.com/watch?v=aunm8xn_tPk (visited on Mar. 5, 2023).

[16] "Oculus lipsync for unity development: Unity — oculus developers." (), [Online]. Available: https://developer.oculus.com/documentation/unity/audio-ovrlipsync-unity/ (visited on Jan. 28, 2022).

[17] S. Sugimoto, *VRM LipSync sample*, original-date: 2018-11-23T12:02:35Z, Mar. 2, 2023. [Online]. Available: https://github.com/sotanmochi/VRMLipSyncSample (visited on Mar. 5, 2023).

[18] C. R. Glass, "Self-expression, social roles, and faculty members' attitudes towards online teaching," *Innovative Higher Education*, vol. 42, pp. 239–252, 2017.

[19] U. Technologies. "Unity: Create with vr." (), [Online]. Available: https://learn.unity.com/course/create-with-vr (visited on Feb. 1, 2023).

[20] I. Mahalil, A. M. Yusof, N. Ibrahim, E. M. M. Mahidin, and M. E. Rusli, "Implementation of an effective locomotion technique in virtual reality stress therapy," in *2019 IEEE Conference on Graphics and Media (GAME)*, 2019, pp. 1–6. DOI: 10.1109/GAME47560.2019.8980987.

[21] Voltstro-Studios, *Unity web browser*, Accessed: March 12, 2023, 2023. [Online]. Available: https://projects.voltstro.dev/UnityWebBrowser/articles/.

[22] K. K V, Y. B, R. Raut, and S. T, "Edu vr: Design and implementation of virtual classroom environment in vr for remote learning," Jun. 2021. DOI: 10.20944/preprints202106.0447.v1.

[23] Y.-T. Chen, C.-H. Hsu, C.-H. Chung, Y.-S. Wang, and S. V. Babu, "Ivrnote: Design, creation and evaluation of an interactive note-taking interface for study and reflection in vr learning environments," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 172–180. DOI: 10.1109/VR.2019.8798338.

[24] I. H. F. dos Santos, L. P. Soares, F. Carvalho, and A. Raposo, "A collaborative vr visualization environment for offshore engineering projects," ser. VRCAI '11, Hong Kong, China: Association for Computing Machinery, 2011, pp. 179–186, ISBN: 9781450310604. DOI: 10.1145/2087756.2087781. [Online]. Available: https://doi.org/10.1145/2087756.2087781.

[25] P. S. Inc., *Pdftron sdk*, Accessed: March 12, 2023, 2021. [Online]. Available: https://www.pdftron.com/documentation/web/.

[26] Zeit, *Next.js - the react framework*, Accessed: March 12, 2023, 2021. [Online]. Available: https://nextjs.org/.