Accessible SDKs in Virtual Reality

A Multimodal Means of Orientation for BVI Individuals in VR

Senior Design Group: G22

Kasidy Landry

Emmanuelle Rebosura

John Boyd

Justin Morera

Sponsor:

Joey Down

Polaris Technology Group

Table of Contents

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

Problem to solve

The most exciting aspect of virtual reality is immersion. What makes it more unique than regular video games and simulations is the fact that you are using a headset to see into the environment and the controllers used are almost like an extension of your limbs. These features allow you to be visually and physically more immersed.

The virtual reality headset only allows you to see the virtual environment and nothing else, making it look like you are physically there. This cannot be achieved with a regular computer screen as it only displays the environment in which you are interacting with in a finite area, which in turn means you can see outside the environment which takes you out of the immersion. The controllers for virtual reality work similarly. When you are using a computer to play a game for example, you are using a mouse or a keyboard to traverse through the gameplay, whereas in virtual reality, you are using two controllers in your hands and moving them around and using them for everything like your own hands.

In the real world, people who are blind and visually impaired can get accommodations for that to make their everyday experiences a little better. They are able to more comfortably interact with their environment with tools such as text to speech, mobility canes, eyeglasses, or magnifiers, whether in real life or in 2D technology.

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

There are little to no accommodations for the blind and visually impaired in 3d virtual reality as opposed to the real world or even 2D technology. As previously described, virtual reality uses the whole body to function. Because of this, the commercialized technology of virtual reality that is used by the general public can be very inaccessible. Although some people with disabilities may have access to special personalized virtual reality technology, the common person will only have access to the type of technology previously mentioned. Therefore, having the ability to add visual accessibility features to virtual reality opens it up to a whole new group of people who previously have not been able to enjoy virtual reality, at least not as comfortably as others.

This is what our project does. We further developed an already existing SDK that adds accessible features for the blind and visually impaired to any virtual reality software.

Getting Started

Immediately after this project was assigned, a Discord channel was created for team communication. Then our sponsor, Joey Down, was contacted via email. We introduced ourselves, then scheduled an online video conference to formally introduce ourselves to Joey. We introduced ourselves by sharing what experience we have and what inspired us to choose this project. Kasidy shared that one reason she chose this specific project pertaining to virtual reality is because she does not have any experience with VR and wanted to learn more about it. She also shared that

she wanted to work on something important for society, so accessibility interested her.

After our introductions, Joey spoke about his visual impairment and how Accessible SDKs for VR, initially titled Accessible APIs for VR at the time, was inspired by his own struggle with low vision. Next, we spoke about the first steps for us to start working on the project. Joey explained that the next major step was to complete IRB testing so that we can use the results from testing to enhance the already existing program. While the IRB testing is getting worked out, he advised that we should contact the previous project manager to get the previous team's repository and documentation about their work on the project. He suggested we familiarize ourselves with those things before we can start working on it.

After Joey left the meeting, our group stayed behind to discuss our roles, most importantly the role of Project Manager. The role of Project Manager was down to Emmanuelle and Kasidy. We both stated our cases including our strengths and qualifications, then as a group we decided. The decision was that Kasidy would be Project Manager and Emmanuelle would do communications. They would facilitate communication with Joey, Logan, and anyone else as necessary on behalf of the group. This would leave Kasidy to lead the group and oversee the project's success as a whole. About a week later we assigned preliminary roles to Justin and John, those being lead researcher and product owner, respectively. Since we would not be working directly on the project development for a while, we established these roles with the expectation that more technical roles would be assigned when development starts.

After getting the contact information for Logan, the project manager for the team that previously developed the program, Emmanulle emailed him to introduce our group and ask for the necessary documents and items needed for us to take over the project. After emailing Logan, we received his response with the repository and their design document. Individually we reviewed the design document to get an idea of what the project contains, then in our weekly meeting we discussed it. We discussed what the previous team did and what we thought we could add to it. This began our research and brainstorming for the future of the project.

IRB and Testing

IRB stands for Institutional Review Board. These are committees that evaluate proposed research studies to make sure that the participants are protected by the regulations and ethical standards that are in place. Before the research studies can be conducted with participants, an Institutional Review Board must approve the study, making sure it adheres to the regulations in place, meets certain ethical standards, follows institutional policies, and protects the people participating. In addition to reviewing before testing begins, the International Review Board must also periodically evaluate the study to make sure it continues to meet the required standards for the duration of the research study. For a research study to pass the evaluation, it must be clear to the International Review Board that the potential risks to participants are as minimal as possible and reasonable regarding the results that the study is expected to produce. In addition to

that, there must be clear reasonable plans to get the consent from subjects and that said consent is adequately informed.

Our project, Accessible API in Virtual Reality, was already established when we chose it. It has code that works and is playable. The next thing that the previous team wanted to do was test it with people who would benefit from the accessible features integrated into it. Our sponsor was in contact with the International Review Boards and the University of Central Florida's Student Accessibility Services. He got approval from an International Review Board to begin the study. From this study we as a group learned more about what works and what does not for the participants. We used the feedback for our project as a starting point for our work on the project. We used the reviews to make decisions on what to do next and what to change about the existing project.

Technology Stack

Codebase

- C#

  Our code is primarily written in C# utilizing its Object-Oriented principles and simplistic syntax to create organized and easily readable script files. These scripts are organized automatically by Unity, a game engine and development platform for three-dimensional experiences.

- C++

  The code was translated into C++ code files at compile-time to be read and run by the Unity game engine itself, allowing users to write the initial code in any language (in our case, C#) that can produce .NET IL they are comfortable with without sacrificing optimality in translating to C++.

Version Control System

- Git & Github

  The files were maintained both locally and in our Git repository hosted remotely via GitHub for parallel programming practices between our team members, as mentioned earlier in this document. The scripts are assigned to their respective Unity objects or tools and are easily identifiable via the Unity Hub program that parses the directories and automagically sorts the object files and designates their proper script file for immediate editing.

Code Editors

- Unity

Unity also offers a number of helpful features that automate the difficult process of maintaining the simulation files and tools across multiple platforms as well as allow for straightforward testing. Unity essentially took care of the cumbersome task of maintaining cross-platform portability and allowed the application to be run on Windows, Mac, and Linux systems in addition to the multitude of virtual reality headsets that were available during development.

Additionally, the simulation could be tested on a virtual reality headset immediately through Unity without the need to host on a game platform such as Steam or Epic Games. The models for each object could also be observed in a three-dimensional environment and manipulated visually without requiring code changes as the files were automatically regenerated upon saving.

- Visual Studio & Visual Studio Code

  The primary integrated development environment (IDE) among our team was Visual Studio Code, as it offers a plethora of useful extensions, simplified integration of console-based and user interface-based commands for maintaining the Git repository and local development directories. In addition to Code, our team also worked with standard Visual Studio for it’s built-in and easy integration into C# and Unity projects.

  Additionally, because everyone on the team used Visual Studio Code for development on this project, it reduced confusion and

increased cooperativity between team members when there was an issue because we were all using the same environment and were able to help each other more whether by assisting each other with navigating the user interface, installing extensions or packages, or viewing each other's code. Having everyone on the same setup with developer programs increased efficiency even though our team was not requiring that each member use the same programs.

Hardware

- Meta Quest 2

  As our project was entirely based in virtual reality environments, our technology stack must include a virtual reality headset. Our sponsor graciously supplied us with a Meta Quest 2; this 224-millimeter by 450-millimeter piece of awe-inspiring technology came complete with a fast-switch LCD display offering a resolution of 1832 pixels x 1920 pixels per eye and supports 60-hertz, 72-hertz, and 90-hertz refresh rates, six degrees of freedom positional tracking of controllers and the headset, 256-gigabytes of storage, two newly-designed ergonomic controllers with thumb rests, and built-in three-dimension positional audio, all of which helped our team to test the capabilities of the audial, visual, and haptic features under development. An additional note is that the headset was

also glasses-compatible, which allowed everyone on our team to use the headset with ease, leaving no developer out!

Communications

- Discord

  Our primary means of communication was through the messaging platform Discord, on which we created our own server. There we were able to quickly get in contact with each other and respond to any needs that the team may have had. We also performed our Scrum daily stand-ups through Discord, as a means of holding ourselves and each other accountable for the work that we did.

Organization

- Atlassian Jira

  We worked with Agile Scrum methodologies to develop this project. We decided that the best tool to aid in keeping track of our tasks and responsibilities was Jira. Jira allowed us to organize our tasks into sprints and easily allowed the rest of the team to witness our progress in the project.

- Google

Our files were stored in a shared Google Drive folder for easy access for the entire team. We used Google Sheets to keep track of our talking points for each meeting. We also used Google Docs to write collective documents and to record the minutes of our meetings with our sponsor.

Experience

Since the project already existed, we did not have any say in the technology used for the project. Most of the technology that the previous group used we had no experience with so we learned how to use them on our own and as time went on.

Based on the previous group's final design document, they used Unity, Virtual Reality, Lucidchart, GitHub, C#, Visual Studio Code, Jira, and Discord. Of those listed, most of us only had experience with GitHub and Discord so it was an adjustment for us to get familiar with them so we can contribute to the evolution of the project.

As a group, we hoped to be able to make significant progress in our skills as soon as possible so that we could contribute to the project as soon as possible. We did not want to make our group fall behind in the development schedule and let each other down. We received a virtual reality headset from our sponsor so that we could test and explore the project that was given to us as well as the future developments we had with the project. A few of us had used virtual reality before so we were confident that we could use the headset to help our development process. We learned more about the technologies used for the project every day and, eventually, we

knew them inside and out and we were able to comfortably work on the project without roadblocks.

Concept of Operations

The project previously consisted of a simulation including six main levels, an office level, and a data display room inside a scene in the Unity application, as well as four main tools: Sonar, Auditory, Haptic, and Visual. The concept was that the levels were made to showcase the tools in various scenarios to improve accessibility in virtual reality environments for blind and visually impaired individuals. At that point, users loaded into the virtual environment and went through the levels to practice using the tools, and some basic data was displayed at the end in the Data Display Room.

However, the focus of this Senior Design Project was to expand on the previous application to thoroughly test the tools, and then to select some of the most beneficial tools for a more modular approach ultimately allowing those tools to be used by other developers in separate virtual environments outside of the simple simulation levels created for this one.

The ultimate goal for our iteration of the project was to modify the tools' code to formulate the files into a development kit that virtual reality developers can simply download or import into their development environment, load the scripts for the tools, and apply the tools' functionality to their own applications. Once the tools were optimized both for accessible ease-of-use, and for portability, the realm of virtual experience development as a whole was made significantly more inclusive and accessible by setting a

precedence of enabling highly sought-after accessibility features to be importable into any environment.

By importing our kit into their environment, our application immediately integrates its features into the user's experience by allowing their controllers and headset to use our accessibility tools. The goal is to minimize setup on the part of developers and require minimal changes to a project's scripts to get the full benefits of the imported features.

Project Summary and Goal

For this project, our goal was to create an SDK toolset that would make virtual reality more accessible to those who are blind and visually impaired. The idea was to create a Software Development Kit for Unity that virtual reality developers can access and insert into their own games and projects, making their content more accessible to a wider audience in virtual reality.

This was a continuation of the work that was done in the first and previous version of the project, where the previous senior design group was able to create a set of four possible orientation tools for blind and visually impaired people to use in virtual reality. For the first semester, we initially planned to do IRB testing with actual people of the blind and visually impaired community in order to test the effectiveness of these different orientation tools in virtual reality. The results of testing would determine which of these tools we would focus on improving and fine tuning during the next semester, and which ones we would no longer be working on.

We also looked into implementing a selection of new features that would help in our goal in creating a more accessible virtual reality for those on the visually impaired spectrum:

- A selective screen reader that reads out any text to the user along with any alt text
- A program that allows developers to tag an object in virtual reality and add alt text to said object
- A feature that allows for high contrast in the colors and environment for the user to see more easily

Initially, these were either not in the original project or were more hardcoded into the original project, which did not allow these functions to be very useful outside of what was built for the first version of this project. However, we wanted to create something that would allow for a more broad use outside of just this project to make virtual reality more accessible, and aimed to also make these features helpful for those who are not visually impaired, making virtual reality a better experience for everyone.

One feature we originally had the idea to implement was a zoom feature that would have functioned similarly to the zoom feature on an iPhone for accessibility– it would operate independently from the developer's software and allow the user to zoom in on the screen. However, it was later determined that we would not be able to implement this due to complications regarding user experience, which will be talked about in this document.

Team Contract

Team Motivations

Kasidy Landry

When choosing between project pitches, I was most interested in projects that would have a positive impact on society. I was also very interested in the projects pertaining to virtual reality since I had played some VR games before but never worked with it. This project combined those two interests making it a top choice of mine.

I believe that accessibility should be more available in every aspect of life. Technological advancements such as virtual reality aren’t as impressive if only a certain group of people can comfortably use and enjoy them. It has been shown that accessibility for a few can also help the majority’s

convenience, making it helpful to everyone. I personally like to use closed captioning, if it is available, to watch movies and shows because it helps me process what is being said even though I am not hard of hearing. Something that is available to make visual media more accessible to those who are hard of hearing is also helpful to those who are not. This is why I believe making virtual reality a little more accessible would be beneficial for everyone. Even if someone does not use accessible features, them being available does not hurt them, but it can be life changing for someone who needs it.

I was excited to work on this project because I knew I would gain experience working with virtual reality which is something I am quite interested in but have never had the opportunity to work on. On top of that, making something as entertaining as virtual reality more accessible to everyone was something I was eager to do so that more people can enjoy the fun that virtual reality offers.

Emmanuelle Rebosura

Going into Senior Design, I had no idea what kind of project I wanted to work on for the next two semesters. Many of the projects being presented for me ended up falling under “kind of interested” or “not interested.” However, watching the video pitch for this project, what immediately caught my attention was that it was a project to help make virtual reality more accessible to blind and visually impaired people.

Inclusivity is something that is very important to me being part of different groups of minorities. Accessibility is something I have only become more aware of and more interested in in the past several years, and it should definitely be something to have in mind when we try to be inclusive of as many groups as possible. As I continue to be more informed about accessibility needs, I realize how much people as a whole can benefit from

these accessibility needs, not just those who are disabled, and how much more we are able to reach people when we allow them the opportunities to be a part of our growing culture.

As technology continues to advance, and advance quickly, I believe it is important to make sure that as many people as possible are able to use and benefit from it. I personally do not have *any* experience with virtual reality– and this includes even using a headset to try out an experience or game in VR– but I still believe it is important to make it accessible for everyone as anyone can benefit from doing so, not just those with disabilities. Virtual reality has many valuable applications, but if we only limit its accessibility to those without visual impairments or people without some kind of mobility restriction, then it does not actually help benefit society as a whole. Instead, it isolates and separates people, leaving them out of what should be an immersive and beneficial experience for everyone.

While this project only focused on making virtual reality more accessible to blind and visually impaired people, what I wanted from this project is for people to not only see that it is possible to make virtual reality accessible to a wider range of people, but also encourage developers to make their virtual reality projects more accessible to *all* kinds of people, including those with disabilities. I want developers to be able to keep disabled people in mind in order to have a wider reach, and hopefully more can be done in the future to expand accessibility needs in virtual reality not just for the blind and visually impaired, but for a wide range of disabilities.

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

John Boyd

There were a lot of considerations that went into picking my Senior Design project. This project is the crowning achievement of my time at the University of Central Florida, so there was a lot of pressure to do my best to get assigned to a good one.

I am excited to work on this project because it’s an opportunity to demonstrate all that I have learned during my time at the university. Recruiters and hiring managers are caring less and less about what classes their applicants have taken or what their GPA was. They want to see tangible projects highlighting what those prospective employees learned in those classes.

Final Design Document

Senior Design is a unique opportunity to show off all the skills I've gathered in one big project. This is a means for me to gain practical, direct experience that can enrich my resume. Learning in a classroom has been beneficial to my growth as a developer, but I need to learn how to apply these skills in practice. This will be my second time working with Unity applications, and my first time developing a package for Unity. Additionally, this will be my first time working with VR systems and applications. Working on a real-world project like this will not only bolster my technical skills but also make me a more attractive candidate for future career opportunities.

More than for the benefit to me, however, I'm grateful that this project has the goal of improving quality of life for others. A belief that I hold is that when you make an experience more accessible for the minority, you improve the experience for the majority as well. I believe in the importance of making VR experiences inclusive and accessible to individuals with varying levels of sight. There are so many unique and amazing experiences in virtual reality, and people of all ages should be able to experience them. This project provides me a meaningful opportunity to address this challenge and I'm happy that this gets to be my final work before graduation.

Justin Morera

I am motivated to work on this project because I think accessibility and inclusivity are very important. I am really proud to have the opportunity to work on a project with such a noble goal and to be able to make a difference in the lives of those who need it. To me, this project is more than just testing the limits of accessibility in virtual reality; it’s about making our changing and ever-advancing world include everyone so that no one is left behind as our technology grows.

In a more personal and technically-oriented light, I am also excited to get experience in working with virtual reality as a developer because I have always found virtual reality to be an invigorating and rather mind-blowing experience, and I would love to be able to gain knowledge into developing some of those experiences for others in the future.

Meetings

As a group, we met weekly on Thursday nights to discuss what we have done, what needed to be done, and any issues present. Every other week, our sponsor, Joey Down, attended these meetings, and if there was something specific we needed to discuss sooner, he would join our weekly meeting to address it.

Kasidy created a Google Sheets notebook to assist us with task management and meetings. Every meeting had its own spreadsheet in the notebook (see Figure 1). Each spreadsheet has the date of the meeting, an attendance tracker, an agenda list, a to-do list, a link to the meeting's minutes, and the date of the next meeting. It also has each team member's name so that tasks can be assigned to them. If an agenda item pertained to a member, there would be a checkmark by that person's name assigning the task to them and that would let them and the team know that during the meeting, that issue would be taken by that person. During the meeting as tasks to be done arose, they would be added to the to-do list and then assigned to the team member whose responsibility it would be. The spreadsheets have two separate to-do list sections. One being what needed to be done after that meeting and before the next meeting, and then a separate section to list the things that needed to be done in the long run. They are labeled accordingly and separated for clarity and for organization purposes. At the end of a meeting, the to-do lists would show the team what each member will be doing during the next week as well as over a longer period of time.

Final Design Document

Our team member John was in charge of Jira and the scrum progress. Jira also kept track of what we need to do, but we decided to have this notebook as well so that we could have one spreadsheet showing everything that relates to each individual meeting. This also allowed us to have something to refer to during meetings as well as later since it retains the previous meetings' spreadsheets.

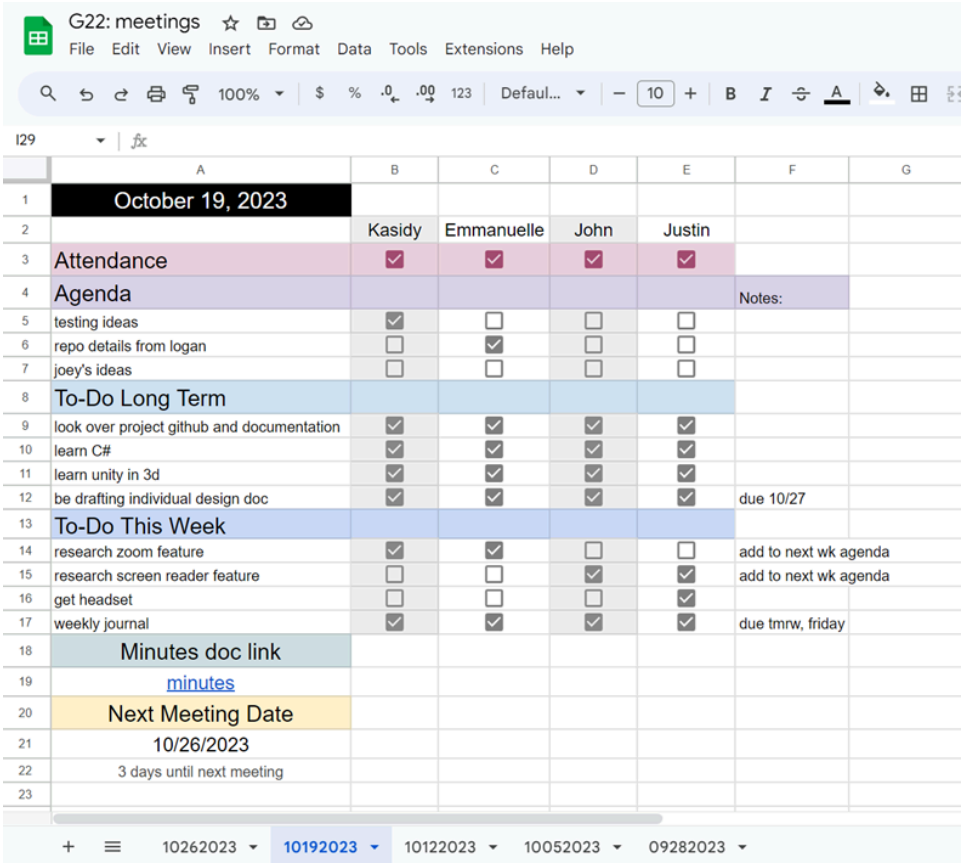


Figure 1: Spreadsheet for the meeting held on October 19, 2023

Expectations

Communication is the most important aspect of working with a team. We expected every team member to communicate any problems, questions, concerns, developments, and achievements with the rest of the team. After any aspect was completed, a simple message in the Discord chat to notify the team was expected.

We wanted to ensure everyone knew what was going on at all times, and we achieved this by communicating constantly and clearly with each other. We also did not want anyone to be struggling with something alone. Doing this would make things more difficult and time consuming not just for that team member, but for the entire team. If someone was struggling with any part of the project, they would alert the team as soon as possible so that others could help.

Each team member would check the Discord server, Jira project, and meeting spreadsheet periodically and consistently. They would check the Discord server at least every day and have notifications enabled in order to keep up with what was currently happening and to be aware of any changes as soon as possible.

Starting Ideas

Kasidy Landry

This project is one that a previous senior design team worked on, meaning that we inherited this project to continue it and to make it better. The previous team’s next step in development was testing. The project was

to be tested with real people who would benefit from the accessibility features and give feedback to enhance the experience. This makes the testing phase our first move. We will have to test it and get the results back to know exactly what to do next. Before testing, however, we will need to come up with specific ideas on what we want to know from the testing. One idea I had for what I would like to know from testing is customization options. Are the features customizable enough to make it helpful for people with different needs? Another thing I would like to know from the testing is whether the visuals are good enough for people with low vision or colorblindness. Are the color contrasts effective enough for anyone to distinguish them clearly, and are the fonts and font size chosen easy to read? These questions along with others will help us to know what needs to be enhanced.

Research will also help us determine the most accessible features to incorporate. I think we could do some research to discover new features to add in addition to the existing ones. I don’t want to have only a small amount of accessibility features, so I think getting as many features integrated as possible is important. Even if a feature is less used than others, I believe even if it helps only a few people, it is beneficial. Of course, we will want to refine the most used features to make them as effective as possible as well. I think we will want to assign a research role to a team member who will focus on gathering information on new features while the others can refine the existing ones.

Emmanuelle Rebosura

At this point in time, I did not have too many definitive ideas for the project, as we needed more information from the research and testing of the previous version of the project before we knew exactly how to go forward with this project. Input from the community this is meant to help benefit would allow us to understand what works well and what does not, which would give us the direction we want to go and figure out what we should improve on or what we should scrap all together. However, that did not mean there were no ideas as to how we could improve what has already been worked on, as well as ideas on how we could test the previous project.

The previous group for this project created different tools for the user to navigate the levels they have designed. For the research and testing in the first semester, we needed to know which of these tools were most effective to use. While a built-in timer was developed for the levels, we could also use a heatmap to determine the path users take to navigate through a level, which could show us if there were specific areas where users would get stuck and give us an idea as to how effective the previously built tools are. Counters could also be used to see how often someone used a specific tool, which would give us insight as to how much they rely on it.

Aside from improving and possibly reworking any of these features already built, we could also work to make the UI more accessible for users, especially for a project focused on accessibility. Since customization was not something that was implemented in the previous version of the project, that is something we could add, such as allowing the user to change text size or

font to be easier to read, or allowing them to change the sound for the auditory tool to make a more pleasant experience trying to navigate.

If needed, I also believed we could also design a couple more levels if we find that we managed to improve the tools we were given, though of course this would have required more testing. Users would have to be able to navigate through the previously built levels in a rather efficient fashion as well as informing us that they found the tools to be effective and practical before we could think of building more levels, but I felt if we ever reached that point, creating new levels to navigate would have helped test out our work more to ensure it was not just optimized for the previously built levels we already have.

John Boyd

As a member of the second team to work on this project, I feel it's important to clearly understand the groundwork done for us. We need to understand what our predecessors were trying to implement, which of their ideas worked and which of them didn't.

I want to analyze and interpret the results of the Institutional Review Board (IRB) testing that the previous team had planned but not conducted. Understanding how individuals with varying levels of sight interact with VR can provide valuable insights into the effectiveness of the tools developed so far and guide further improvements. Through this testing, I hope for us to engage with the broader community interested in accessible VR technology.

To ensure the project's success, I want us to be part of an iterative development approach. This means continuously refining the accessibility features based on feedback, testing results, and emerging best practices in VR accessibility. The goal is to create a solution that continually improves over time. In the pursuit of this, I hope we can continue to engage in detailed discussions with the previous team members to gain insights into the features they implemented and their perceived success. This collaborative knowledge-sharing process will help me understand the project's history, identify potential challenges, and build upon their work effectively.

Justin Morera

General

I think we can expand the testing to see how helpful the current model is and how we can improve each feature. I think time per level is a good metric, but we can also include more data such as the number of times each accessibility tool is used on each level and the rates at which each feature is used. I also think we can work on making these features importable to other virtual reality experiences so that they can be implemented into more developed and larger-scale environments.

Overall, I feel we need to focus on testing effectively first so that we know which features need the most improvement. This should be done before we begin working on making the application importable because we need to know which feature(s) will work best in other applications and should be prioritized to better showcase what the application can do.

As far as research is concerned, I think it would benefit us to better understand how individuals in the blind and visually impaired community feel about their condition and what challenges they face in the realm of virtual reality and everyday life. Understanding our users, and ultimately our stakeholders, will allow us to properly focus our energy on the aspects of the product that are most important. And beyond making a viable product, our knowledge of the problem we are working to address will also help us make a bigger and more positive impact on the blind and visually impaired community.

Testing Ideas

Heatmaps

The project's previous Project Manager suggested using a heatmap to track user movement; we believe this can be expanded further to not only track traversal, but to also track the location of each activation of the different tools. It would be useful to implement separate heatmaps for each tool showing both where tools are used the most, as well as the orientation of the user when using that tool. This will help us gauge not only the difficulty of the levels, but will also show us where the tools are the most

useful and what environments users are able to utilize them. Heatmapping would be most useful if we are able to implement new levels with more complex challenges for users to test; however, that will only be within the scope of this project if we determine that more data is needed to gain a fuller understanding of the tools' effectiveness.

For all testing metrics, the focus should be on determining which tools are most effective and versatile, as well as for what levels of sight impairment the tools are most assistive. Given the scope of this project in the current timeline, we are only able to optimize a limited number of tools for portability to other virtual reality environments; therefore, it is imperative that we make the most out of the testing phase of the project so that we are able to determine which tools are most likely to be portable, and of those, which are the priority as far as usefulness.

Haptics

For the haptics tool, it would likely be beneficial to track the number of uses per level; this could easily be implemented by adding a counter that increments by one with each activation of the haptics tool. The information gained from this metric would enable us to track how often this tool becomes useful and, combined with a heatmap, under what conditions the tool is useful. Tracking this same metric with the other tools would also enable our team to compare which tools are used most often, and again, with the use of heatmaps, which tools prevail under similar conditions. This is most valuable in allowing us to understand what tool or tools should be prioritized for porting into a development kit for other environments.

Another very important metric to keep track of for the haptics tools is the distance between target objects and the user. Knowing how far away users tend to be from an entity when they use the haptics tool will give us valuable insight into what ranges users tend to struggle with identifying objects; combining this with the identification of the target entities themselves will help us determine what types of objects are harder to identify at different distances. For instance, we could determine if smaller objects are less visible at longer distances, or if users struggle to find them at close range where there are other entities in close proximity.

Sonar

The sonar tool is unique and harder to interpret as it mimics an ability that humans do not naturally possess and is notoriously difficult to master; this makes tracking certain metrics extremely important in order to accurately gauge the tool's effectiveness in this project. As mentioned previously, tracking the number of activations will be an excellent metric to track comparatively with the other tools, and if we are able to determine in what situations users rely on the tool, we can see if it offers a unique strategy for navigating certain environments or if other tools are able to cover the same criteria that sonar would cover; if the latter is the case, then sonar may prove to be too advanced for the average user to utilize effectively. However, only our results from testing will allow us to make that determination.

Additionally, it could prove insightful to not only track the number of uses, but also the frequency of activations. Users may activate the sonar feature multiple times in a scenario, but we also want to know how often in a situation users feel the need to activate the tool. For instance, users may

activate the haptics tool less times overall, but the reason it is used less may be because they are able to obtain more information from it and navigate out of an area faster. If a user needs to activate the sonar feature multiple times in order to orient themself in an environment, that may suggest that they are not getting enough information from the tool they chose to rely on.

It could also be helpful to keep track of the distance between the user and the central entity of the sonar wave (whatever object triggers the return first) during activation, as well as the distribution of entities that return a sound with the tool. This seems harder to implement but can also be helpful in determining whether the tool accurately helps a user differentiate objects that are close together, or if it is more effective on objects that are farther apart. This is especially important when considering a series of columns that can be perceived as a solid wall, or a door that is hidden behind a corridor, with walls placed at either side. We want to ensure that users are able to find doors in tight spaces as well as thin openings in a virtual environment.

Partial Vision

The partial vision tool has a lot of potential to be efficient for visually impaired individuals but also has a number of metrics that are important to look at in order to determine its potential limitations. Again, the number of uses should be counted for each level, but counting the number of times a specific entity is targeted by the partial vision tool, across users and across sessions could also prove to be particularly valuable. By counting the number of times an object, say a button or a lever, is targeted with the partial vision tool, we will be able to determine how visible that object is to visually impaired users; we can tell whether the object is placed in an area where it is obvious, or whether the design of the object makes it easy for

users to determine that it is interactable. This information can help both with improving level designs as well as modifying the tool itself. Looking at how this tool's use relates to other tools will also help identify what makes an interactable object locatable.

Combining these metrics with the metric of distance, much like with the sonar tool, will allow us to understand when the average distances users of different levels of visual impairment are able to identify an object as in need of further information. And finally, another interesting metric that works in tandem with distance is the number of times an object is missed by a user. While harder to implement, if we are able to see how many times a user has to activate the tool in quick succession before ending the streak on a particular object, we will be able to understand the difficulty of using the tool on different sized entities, and at different distances; this is especially important when considering users with physical impairments in addition to visual impairments, and can lead to changes in the sensitivity of the tool, allowing easier targeting of farther or smaller objects.

High contrast

High contrast accessibility options are highly desirable in the blind and visually impaired community. This feature is easily implemented and its value can be tracked simply by counting the number of users that prefer having the feature active, and asking users for their thoughts on how much it improves the experience.

Menu manipulation settings

The project features a number of ways to manipulate the menus which help control text sizes and allow users to control how the text is displayed to them. Tracking which zoom levels and settings are most commonly used can be beneficial to understanding what challenges users face, as well as how we can improve the settings for future designs.

User opinions of each tool

The final area for testing metrics that we have deemed necessary to obtain would be in gathering our testers' thoughts on the tools presented to them and recording what went well and what did not. There are a number of questions we should be asking, before, during, and after the experience, which could shape how we move forward with this project. First, we need to understand our users' challenges with sight, and with virtual reality so that we can also understand how their condition affects virtual experiences, with or without accessibility features.

Specific information we should extract from user surveys includes complaints or concerns they come up with after exposure to the various tools and features; this could potentially allow us to become cognizant of challenges we, or the previous team, may not have considered during development and research.

We should also record what users found to be effective and what they found to be ineffective; as stated in the earlier sections, raw counts of tool activations may skew data in a way that makes a particular tool seem better, but in actuality, that tool may have required more activations to be as useful as another tool which would only need to be activated once. The previously

listed metrics will hopefully mitigate these situations, but a surefire way to understand what users in the blind and visually impaired community found effective is to ask them directly. Another situation to watch out for is when a tool is deemed to be most effective comparatively, but it is not effective in an objective sense; for instance, if we find that users maximize the text size in menus, we may assume that the max setting is the best for users, but it may be the case that the maximum text size allowed is still not big enough and is simply the best size available. Outcomes such as these are still valuable because they outline changes that need to be made to the tools and features in order to make them better and more usable outside the testing environment.

One final metric to prioritize in user surveys would be a question, or set of questions, to scale how realistic and natural these tools feel, both compared to navigation in real life, and in other digital settings. We need to know if the sonar tool is naturally interpretable by our users or if using it is an acquired skill; likewise, we need to know how realistic the haptic tool is to similar assistive technology in real life, such as a walking stick. And of course, we can’t ignore how the more conventional accessibility features, such as high contrast and menu zoom, compare to accessibility features on phones and computer screens.

Additional ideas

Add actions to Partial Vision tool

I thought it might be beneficial to improve the partial vision tool by making it also allow users to interact with selected objects when possible. Specifically, if the targeted object is a door, button, lever, or item that can be

picked up or used, etc., and is also within range to be activated in any such manner, the tool should have easily selectable buttons that the user can press to perform such an action. This would eliminate forcing a user to swing a door or pull a lever or other related action in situations where they are having trouble finding or accessing the object. This also strengthens the ability for users with motor impairments in addition to visual impairments because it allows them to perform more complex actions without fine motor skills as a requirement.

Project Timeline

Gantt Chart

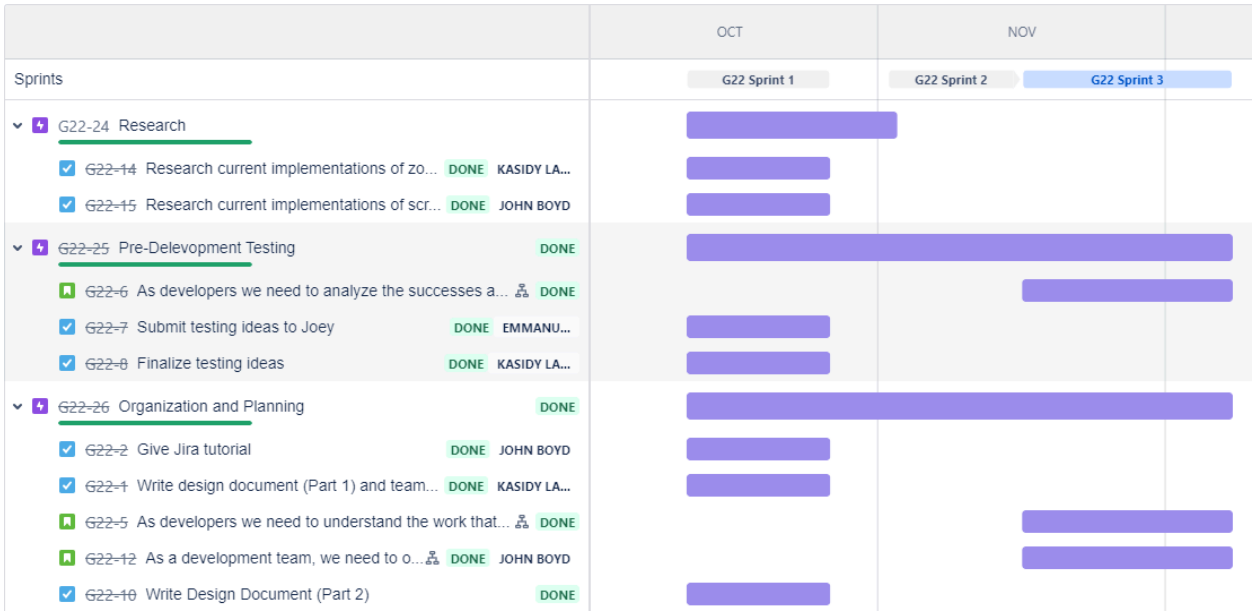


Figure 2: Jira Gantt Chart

Project Backlog

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↓	Updated
☑	G22-28	code magnifying glass	Kasidy Landry	Kasidy Landry	=	DONE	Done	Nov 17, 2023	Nov 27, 2023
☑	G22-27	Finish design document	Unassigned	Kasidy Landry	=	IN PROGRESS	Unresolved	Nov 17, 2023	Nov 17, 2023
↯	G22-26	Organization and Planning	Unassigned	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
↯	G22-25	Pre-Delevopment Testing	Unassigned	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
↯	G22-24	Research	Unassigned	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
▣	G22-23	Familiarize ourselves with use of the Oculus	Unassigned	John Boyd	=	DONE	Done	Nov 2, 2023	Nov 13, 2023
▣	G22-22	We need to make Jira represent our complete work in the form of a Gantt chart	Unassigned	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
▣	G22-21	We need to create use case and work distibution diagrams to represent our work	Justin Morera	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
▣	G22-20	Add an additional six pages to the Design Document	John Boyd	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
▣	G22-19	Integrate DailyBot with Jira to get standups in document form	John Boyd	John Boyd	=	DONE	Done	Nov 2, 2023	Nov 13, 2023
▣	G22-18	Code in testing metric logs	John Boyd	John Boyd	=	TO DO	Unresolved	Nov 2, 2023	Nov 13, 2023
▣	G22-17	Reorganize the Github repository	John Boyd	John Boyd	=	DONE	Done	Nov 2, 2023	Dec 3, 2023
▣	G22-16	Create new sprint	John Boyd	John Boyd	=	DONE	Done	Nov 2, 2023	Nov 13, 2023

Figure 3: Jira backlog

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↓	Updated
☑	G22-15	Research current implementations of screen readers in VR and non-VR contexts	John Boyd	John Boyd	=	DONE	Done	Oct 19, 2023	Nov 2, 2023
☑	G22-14	Research current implementations of zoom in VR and non-VR contexts	Kasidy Landry	John Boyd	=	DONE	Done	Oct 19, 2023	Nov 2, 2023
☑	G22-13	Set up a Discord bot to do our daily stand-ups	John Boyd	John Boyd	=	DONE	Done	Oct 12, 2023	Oct 19, 2023
▣	G22-12	As a development team, we need to organize our work	John Boyd	John Boyd	=	DONE	Done	Oct 12, 2023	Dec 3, 2023
☑	G22-10	Write Design Document (Part 2)	Unassigned	John Boyd	=	DONE	Done	Oct 12, 2023	Nov 2, 2023
☑	G22-9	Code in testing metric logs	Unassigned	John Boyd	=	DONE	Done	Oct 12, 2023	Dec 3, 2023
☑	G22-8	Finalize testing ideas	Kasidy Landry	John Boyd	=	DONE	Done	Oct 12, 2023	Nov 2, 2023
☑	G22-7	Submit testing ideas to Joey	Emmanuelle Rebosura	John Boyd	=	DONE	Done	Oct 12, 2023	Nov 2, 2023
▣	G22-6	As developers we need to analyze the successes and failures of the existing application	Unassigned	John Boyd	=	DONE	Done	Oct 12, 2023	Dec 3, 2023
▣	G22-5	As developers we need to understand the work that was done before us	Unassigned	John Boyd	=	DONE	Done	Oct 12, 2023	Dec 3, 2023
☑	G22-2	Give Jira tutorial	John Boyd	John Boyd	=	DONE	Done	Oct 12, 2023	Nov 2, 2023
☑	G22-1	Write design document (Part 1) and team contract	Kasidy Landry	Kasidy Landry	=	DONE	Done	Oct 5, 2023	Nov 2, 2023

Figure 4: Jira backlog continued

Starting Foundation

As stated previously, we continued this project where the previous group left off. That group developed the existing project from scratch and were able to produce a fully functional prototype ready to be tested. They created accessibility tools for low vision as well as a simulation to use those tools. Their simulation includes multiple rooms to put the accessible tools created to use. In their development, they utilized partial vision, haptic, sonar, and auditory feedback to communicate the user's surroundings to them while interacting with it. Those methods work together to allow the user to get a better sense of the obstacles surrounding them in the environment, as well as what objects are able to be interacted with, and where they can go in the virtual world. The previous team created multiple rooms so that they are able to change the challenges in each room allowing for varying difficulty. The objective for every room is to be able to navigate from the entrance to the exit using the accessible features created for assistance when needed. The goal for the accessible tools in the simulation is to give the users autonomy in the virtual world, meaning that they will not have to rely on other people to help them navigate and experience virtual reality. Therefore, the four of us will be building off of the existing project handed down to us by optimizing their features as well as adding completely new features.

As mentioned above, the previous group utilized partial vision, haptic, sonar, and auditory feedback in the project. As a collective, we will continue to work on, research, and develop these tools. We split the topics between the four of us to begin researching. Kasidy has chosen to focus on partial vision feedback. In the current state of the project, the partial vision tool is used to give information about items in the virtual world to the user. It is also used to determine the distance of an object so that they could overlay the object with a color filter based on the distance so that its distance can be determined by color gradient. They encountered a few obstacles while developing this feature but were able to successfully implement it. Kasidy thinks something we could do to make it better for the user would be to give them the option to have the partial vision tool constantly display item info on the screen somewhere. For example, if a user likes this tool and knows they will use it a lot, they could go into a settings menu where they can choose to always display the tool. Instead of having to activate it when needed, it will be displayed as a translucent text box on the side of their vision with the exact location being customizable. This means that hovering over any object with the raycasts(laser pointer-like visual markers that show the user where they are pointing in the virtual environment) will automatically show the information about the item in the persistent text box.

Previous Project

The previous senior design group created the project on Unity, coding everything in C#. Everything that they made through Unity is found on their Github repository, where everything can be downloaded and used.

The main components of the original version of the project are the testing levels to traverse and orientation tools to aid the user in traversing the levels. All of these tools are designed to allow the user to navigate in virtual reality without having to rely on vision, which much of virtual reality is currently dependent on.

There are four main orientation tools that were developed by the previous team:

- Sonar tool
- Auditory tool
- Haptics tool
- Partial vision tool

Each tool gives the user different information about their virtual environment depending on what they are interacting with or what is around them. The tools allow the user to rely on other senses, such as sounds or vibrations in their controllers, each meaning something different for the user to understand and process about their virtual environment.

The eight testing levels were designed specifically with the four orientation tools in mind. They are meant to be navigated and completed using the orientation tools for assistance.

- Levels 1 through 4 were made to slowly introduce the user to the different tools one at a time, each room focusing on a different tool.
- Levels 5 and 6 combine aspects of the first four levels, allowing the user to practice using all the tools at once.

- Level 7 is just a simple room that shows the user how much time it took them to navigate through the previous levels.
- Level 8 is an extra level that simulates an office environment. Similar to levels 5 and 6, this level combines aspects of the first four levels for the user to navigate through using all four tools, within an office setting. This is meant to showcase how the user can use the different tools to navigate a semi-realistic environment in virtual reality.

All the while, the levels slowly increase in complexity as the user progresses and becomes more accustomed to the different orientation tools.

Auditory Orientation Tool

The auditory tool created by the previous group tells the user how tall an object in front of them in virtual reality is using different pitches for the user to listen to; when the pitch is lower, the object in front of them is shorter, and when the pitch is higher, the object in front of them is taller. If there is nothing in front of them, nothing will play. This feature is also used in combination with the sonar tool, both of them activated at the same time using the same button. While both tools use audio feedback for the user to determine the environment around them, they use audio in different ways, making it easier to distinguish the information the sonar tool and auditory tool provides for the user.

The previous team looked into using audio and sound for this project, as they learned that sound is very important for those who are blind or visually impaired. As they state in their original design document, hearing is considered one of the two most important senses along with vision, and since those in the blind and visually impaired community lack vision whether completely or to a certain extent, they learn to rely on their hearing. Audio is also an important aspect of virtual reality to allow the user or player to be fully immersed in the virtual environment.

Something that the previous group had to consider and work with was determining what sounds would be useful for their tools so that they would not be too distracting to the user nor would they be too overwhelming. Many different sound styles in virtual reality were considered and looked into, especially when most of the orientation tools rely on sound. Whether to change pitch or volume for the Auditory Tool specifically was also something that had to be determined, settling with pitch in order for the Auditory Tool's sound to be heard, as the user will need to be able to hear it to gain information.

One big decision made was to put in sounds that would be triggered by the user by the push of a button. This would be most useful as the ultimate goal was to be able to guide someone who is blind or visually impaired in virtual reality, as it would provide them the most feedback about their surroundings.

Haptics Tool

This tool is meant to mimic a walking stick in real life; while the user holds down the right trigger on their right controller, the controller's raycast emits vibrations when it is oriented toward objects at different proximities from the user. The controller emits a constant vibration in this state if an object is detected within the maximum range (2 units) which increases in intensity as the distance between the user and the object (the end of the raycast) decreases. The haptics tool also has had multiple additional features added onto it after testing the initial performance. These features all work seamlessly with each other to improve users' navigation and orientation in the level.

There are three intervals for the tool:

- 1.33-2.0 units away – low intensity vibration
- 0.67-1.33 units away – medium intensity vibration
- 0-0.67 units away – high intensity vibration

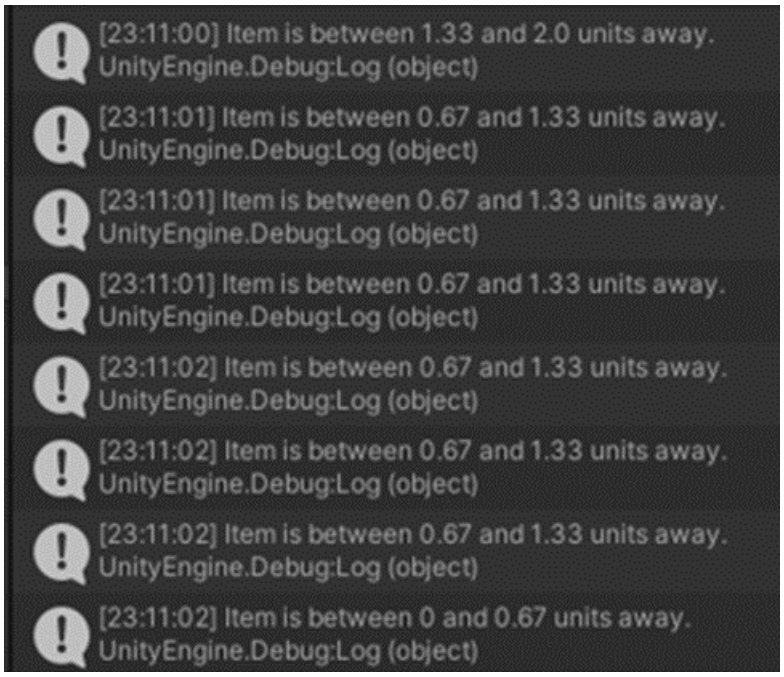


Figure 5: Debug Menu with distances between user and targeted entity.

The tool also features a waypoint system which assists users in navigating through levels via periodically placed rectangular checkpoint objects the user can walk through. When the controller is oriented toward a checkpoint and within range of it, the feedback will change from a constant vibration to a heartbeat-like vibration and the user will also hear a “ding” sound when they reach the checkpoint to indicate it has been reached and is now disabled. These checkpoints are placed in a way that marks a path to the teleporter exit on each level of the testing application; however, for this tool to be implemented in full for other experiences, these waypoints need to be made available to developers, and those developers additionally need to place the waypoints diligently in order to define a clear path for blind and visually impaired users.

The third aspect of the tool is a sensor for interactable objects, mainly doors, which are given their own unique vibration pattern when within range of the haptics sensor. This vibration is similar to the heartbeat-like vibration given off by the waypoints but is still discernible enough to not be confused with a waypoint. The tool triggers this vibration pattern when oriented toward a door's handle, or another interactable part of an object.

The final aspect of the tool is simply a means to tell a user that they are navigating through a menu; in this case, a vibration is sent to the user's left controller, which is the controller that controls menu navigation, and allows the user to determine whether the menu is open or closed so that they are able to avoid accidentally changing settings or lose their in-game orientation while trying to navigate through the level if the menu is open.

Sonar Tool

Blair et al. (2023) highlights the concept of echolocation, where blind and visually impaired individuals use sound, like the way bats use sonar, to understand their surroundings. By emitting clicks and listening to the way sound reflects off objects, they can determine not only the distance to objects but also their shape and size (see Figure 6).

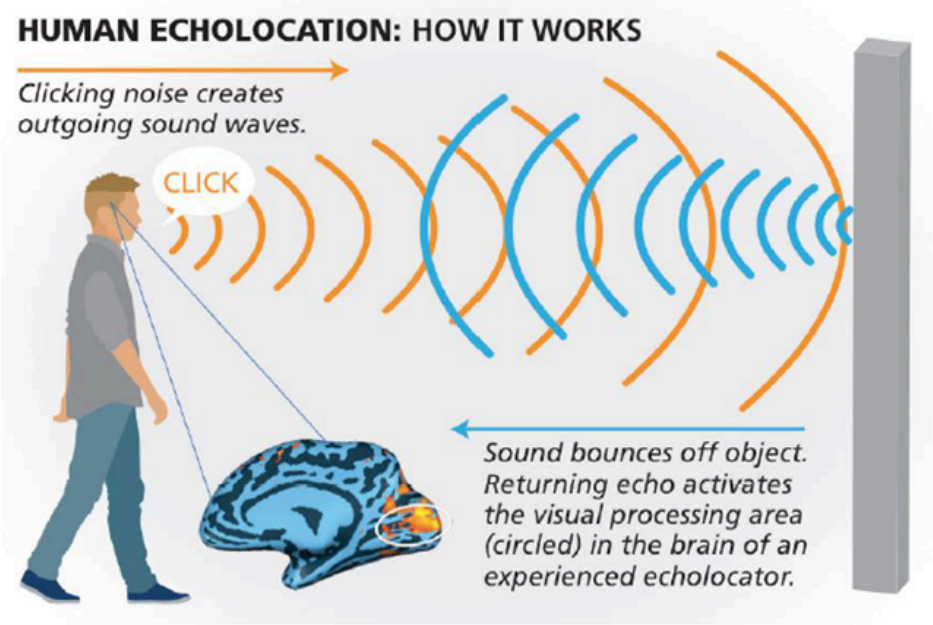


Figure 6: An example of echolocation (Blair, Lugo, Aronne, Le, & Quintana, 2023)

They also delve into the idea of implementing echolocation in virtual reality (VR) environments, where users can explore and navigate spaces through auditory cues. It discusses the choice of sounds to use and their potential impact on VR experiences (Blair, Lugo, Aronne, Le, & Quintana, 2023).

The team discusses their ideas by which they could implement sonar as a VR navigation tool in depth. The feature that was the cornerstone of this tool's development was Unity raycasts. Raycasts are rays calculated in the Unity engine with developer specified origin, direction, and magnitude. The team decided to use these raycasts to simulate the physics of sound reflection. The following means of application were considering how to simulate the production of sound from the mouth:

- An invisible cone mesh:
    - The user emits an invisible cone in which each collision with an object emits a single raycast toward the object.
    - When the raycast hits the object, it generates a sound in the direction of the user, creating a realistic echolocation effect.
    - This process repeats for each object within the cone, allowing the user to determine the presence of objects in the direction they emitted the sound from.
    - This method did not account for sound reflections off of obstructing objects, and the sound plays from the center of the object rather than considering the precise reflection points.
- A cone of raycasts:
    - Multiple raycasts emit from the user's perspective, and each raycast detects if it hits an object.
    - When a ray hits an object, a sound is played at the point of contact on the object, creating a more realistic simulation of echolocation. Unlike the previous idea where only one raycast was emitted, this approach generates multiple sounds from various positions on the detected objects, adding to the realism.
    - The advantage of using multiple sounds is that it allows users to differentiate between objects and understand their relative positions. For example, if two objects are in front of each other, the raycasts can hit different sections of these objects, providing more detailed information.
    - This method did not account for cases where some raycasts do not hit any objects. This can occur when there are gaps between objects or no objects within the cone's range.

    - The team explored a potential solution of creating a cone of rays using a for loop that emits raycasts at various angles. While this method forms a cone shape, it only detects objects along the edges of the cone, leaving the center of the cone empty.
    - To address this issue and create a more uniform set of rays, the research considered a random spread of rays within the cone. However, the team deemed this method less effective due to the lack of uniformity and the potential for varying results each time the program generates the cone.
- Sweeping raycasts across an angle:
    - In this concept, the user's model sends out multiple rays, and each ray detects objects in its path.
    - To implement this, the idea is to make individual rays sweep across a 60-degree angle. These rays space out evenly to ensure accurate detection of objects within the angle (see Figure 7).
    - When a raycast detects an object, a sound plays at the point of contact on the object.
    - The main advantage of using multiple rays is that it allows for a more detailed and realistic simulation of sonar. It overcomes the limitations of previous ideas, particularly in scenarios where objects are in front of each other. The use of multiple rays helps differentiate between objects and provides information about their positions.
    - The challenge in this approach is that the rays create a flat layer, and stacking these layers to resemble a cone shape is complex.
    - To address this, the program would use multiple sweeps of rays that rotate off each other, creating a cone-like shape.

The sweeping raycast method was the one chosen for implementation into the final product, as it solved the issues with obstructions and gaps that the other two ideas had. However, for the sake of performance, the team reduced the number of raycasts in the sweeping pyramid from 20 to 10. They also reduced the angle of the cone from 60 degrees to 45 degrees, to reduce the number of objects detected and prevent sensory overload for users.

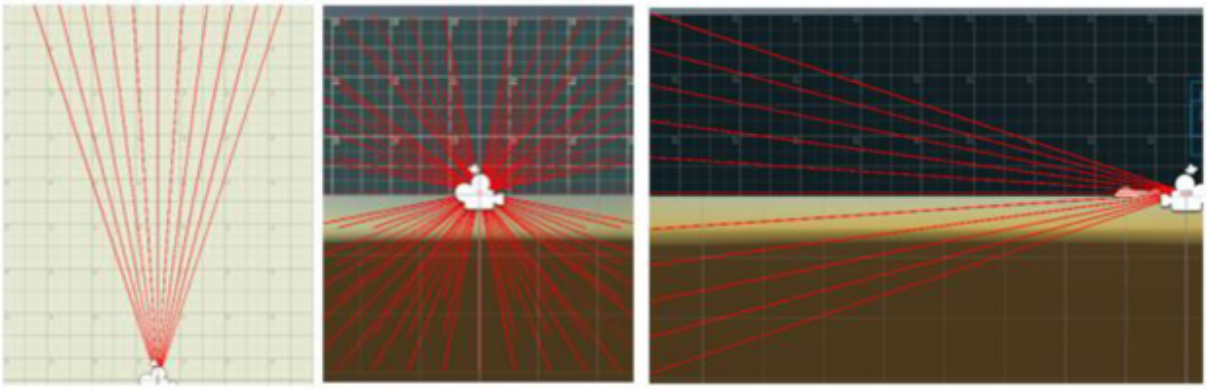


Figure 7: Pyramid of raycasts from the top, front, and side angles (Blair, Lugo, Aronne, Le, & Quintana, 2023)

After deciding how the sound cone would work, their next task was to determine how the rays would bounce off objects they encountered using their original angle and the angle of the surface they contacted. They decided that upon collision the audio would play at the point of collision in the reflected direction.

The current implementation of the sonar tool in this research works like a metal detector in a 3D environment. It utilizes a pyramid of raycasts emitted from the user's viewpoint.

The tool activates when the user holds down the left trigger, ensuring consistency and preventing overstimulation. While active, the pyramid of raycasts detects the closest object to the player, and the tool generates a series of notes. The frequency of the notes varies based on the distance to the object: closer objects trigger more frequent notes, while more distant objects produce less frequent notes. If no objects are within the maximum range, no sound plays. The pitch of the notes is determined by an auditory tool.

Research by John Boyd

Text-to-Speech Technology

History of Speech Synthesis

Speech synthesis is not a new field of technological research. Scientists have been looking into the production of artificial speech technology for a long time, as evidenced by John Larry Kelly and Louis Gerstman’s vocoder in 1961 and Homer Dudley’s Voder (see Figure 8) in 1939 (Weitzman T. , 2023). The emulation of human speech has been an undertaking of science for a long time, even if it is now much more prominent in the public eye thanks to recent advancements.

Figure 8: Homer Dudley's Voder (Crab, 2013)

Text-to-speech (TTS) technology is exactly what it sounds like: a computer receives information in the form of human-language text, and it outputs the language as human-understood speech.

Over the past 50 years, there have been many notable advances in text-to-speech technology that the public has witnessed, including:

- Kurzweil Reading Machines for the Blind are introduced in libraries in 1976 (Weitzman T. , 2023),
- DECtalk’s introduction in 1983, which would allow variation in the pronunciation of words (Weitzman T. , 2023) and go on to be the first vocal synthesizer used by Stephen Hawking,
- Microsoft’s Narrator, first introduced in 1999, which would become a standard accessibility feature in Windows devices (Weitzman T. , 2023),

- Vocaloids, which debuted in 2004, which is considered one of the first pieces of vocal synthesis software to sound like human speech (St. Michel, 2014), and
- Siri, which was released originally as an app in 2010 and has been a standard feature of Apple products ever since (Weitzman T. , 2023).

Current Usage of Text-to-Speech

Today there are a myriad of programs that offer text-to-speech features, including upgraded versions of the products mentioned above. TTS has come to the foreground of public attention recently due to improvements (see Figure 9) in artificial intelligence (AI) and machine learning, which have allowed them to sound more natural, more human than ever before.

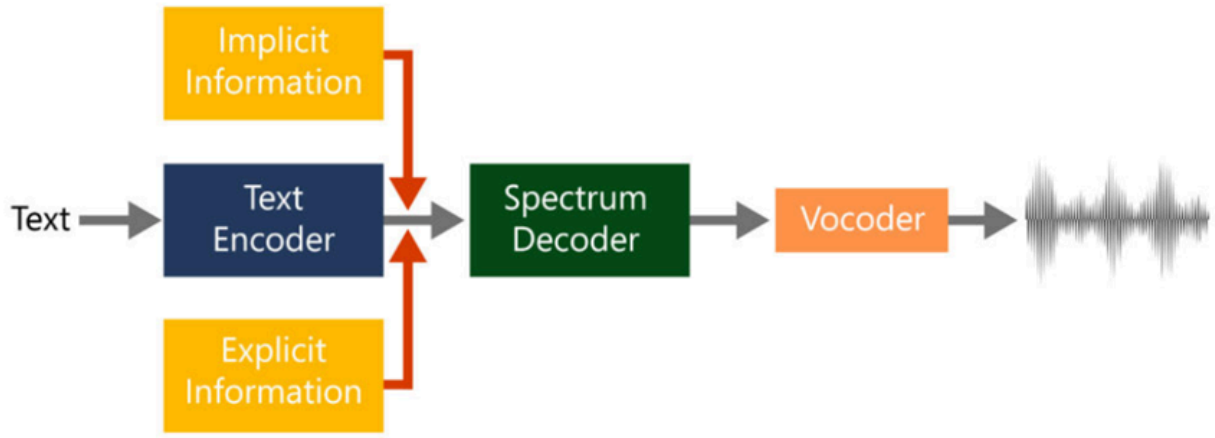


Figure 9: A diagram showing the simplified process by which the Azure Neural TTS model processes text input and outputs speech (Sheng, 2021).

Today, we see TTS used as one of the first tools applied as a means of making applications more accessible for those with visual impairments. Screen readers use text-to-speech to allow BVI individuals to get the

information on-screen without needing to rely on their vision. Audiobooks and news articles often employ screen readers to give users the option to listen to the content on their screens, expanding the accessibility of their information and the options to interact with it available.

As text-to-speech begins to produce more natural output, they are also incredibly useful in localization and translation, moving closer to how native might speak rather than clinically and robotically. The improvement in speech has also seen the use of digital assistants become more widespread.

Modern TTS APIs

Developers looking to implement text-to-speech technology into their software have a plethora of choices available to them. These choices include:

- Google Cloud Text-to-Speech API
- Amazon Polly
- Microsoft Azure Text-to-Speech
- IBM Watson Text to Speech
- Meta’s VoiceBox
- Nuance Text-to-Speech

These TTS APIs provide developers with the tools they need to incorporate spoken language into their applications. The choice of API often depends on factors such as the specific use case, target audience, desired voice qualities, language support, and the developer's familiarity with the platform. Each of these APIs has its strengths and unique features, making them suitable for a wide range of applications in various industries.

Current Limitations of TTS

Digital speech synthesis and TTS technology have certainly come a long way for where it first began. However, despite the recent advances we've seen in text-to-speech thanks to research in speech AI, there remain many drawbacks to this technology.

- Limited expressiveness

  TTS technology has made significant strides in delivering natural and expressive speech, but there are still limitations in conveying certain emotions and nuances.

  While TTS systems can mimic basic emotional states like happiness or sadness, achieving the subtleties of human expressiveness, such as irony or sarcasm, remains a challenge. This is because emotional context and intent recognition are complex tasks that require fine-tuned models and extensive training data (Kuligowska, Kisielewicz, & Włodarz, 2018).

- The ambiguity of language

  Words and phrases often have multiple meanings depending on context, making language inherently ambiguous. TTS systems may struggle to disambiguate these meanings accurately (Kuligowska, Kisielewicz, & Włodarz, 2018).

  While context-based models have improved the disambiguation, there are still challenges in ensuring that the synthesized speech accurately conveys the intended meaning. This issue becomes

even more pronounced when dealing with puns, metaphors, or homophones.

- Unnatural speech

  Despite significant advancements, synthesized speech can still sound unnatural, especially when compared to human speech. Listeners may detect robotic or monotonous patterns in the voice, which can hinder the immersive quality of the experience.

  Achieving a truly natural cadence, tone, rhythm, and pronunciation for all words in all languages is a complex task. The limitations are partly due to the challenges of accurately modeling the diverse nuances of human speech (Kuligowska, Kisielewicz, & Włodarz, 2018).

The Strengths and Limitations of Screen Readers

As mentioned previously, screen readers are one of the most common implementations of TTS technology in modern software. Screen readers put text on screen through the conversion process to describe the words on screen.

However, the challenges of screen readers arise when dealing with images and other objects that the computer does not recognize as text. There are two common ways to address this.

For images, pdf files, and several other file types may contain text that the computer does not recognize as anything more than an image. The text in this image first needs to be extracted using optical character recognition

(OCR). OCR algorithms identify characters in pictures in documents using pattern recognition and allow the computer to interpret and interact with the words in the file. It will also analyze the structure of documents to separate text from other elements of the document to ensure that only text is extracted. When boosted by AI and machine learning algorithms, OCR technology is even more effective and can identify languages and process different handwriting styles (Weitzman C. , 2022).

OCR is a good tool for use with screen readers when reviewing scanned documents. Documents scan as images and the text is not often readable by a computer. However, OCR technology allows computers to recognize the text within a document and thereby allows a screen reader to convert the text to speech for the user.

Although the technology is powerful, OCR alone is often not useful if the images themselves are what’s important for the screen reader to describe, as OCR ignores content in images aside from text. In HTML and other frameworks with similarities to HTML, this problem is addressed by alt text, text hidden from the user that provides a description for images and other objects (Sun, et al., 2023). Alt text is written by the developer or generated automatically, making it an extremely useful accessibility tool (see Table 10).

	Alt Text/Alt Sense	Information to Consider	Examples
Object	Visual properties	Size, shape, colour, etc.	"A tree with round orange and red leaves, 2m high."
	Behaviours	Animation, movement, effects, etc.	"The leaves are ruffling in the wind and some are dropping to the ground."
	Interactions	Grasping, lifting, opening or closing parts, haptics, etc.	"Upon touching the tree with your controller, it will further shed its leaves."
	Relationship w/ users	Ownership, meaningfulness, traces of use, etc.	"John Doe had previously carved a heart into the bark."
	Semantic relationship	Does it belong to a parent object or group of identical objects?	"The tree is part of a forest shaping a clearing."
	Spatial relationship	Distance, angle, height, etc.	"This tree is the closest to you, the others are in a circle around you, approximately 2m away from you."
	Accessible vibrotactile feedback	Type of object, type of interaction, relative distance, etc.	Vibration pattern indicates a portal interaction or relative distance
Scene	Setting	Surrounding, general description, etc.	"You are in a meadow."
	Weather	Weather conditions	"It is sunny with a light breeze."
	Season	Spring, summer, autumn, winter	"It is late summer."
	Light Conditions	Intensity, colour	"It is the golden hour for sunset."
	Mood	peaceful, threatening, exciting, etc.	"It is a cosy scene."
	Action	Main happening within the scene	"Many people are sitting around a campfire."
Avatar	Identity	Gender, cis or trans, age, disability or neurodivergence, race or ethnicity, skin colour	"I am a curious person in my thirties."
	Appearance	Hairstyle, height, clothes	"My black hair falls onto my shoulders and I am wearing sunglasses and a gray shirt"
	Belongings	Personal items, digital collections such as NFT	"My rainbow-coloured handbag lies next to the campfire."
	Social Cues	Attention, facial expression, hand gestures, etc.	"John Doe turns towards you and opens their arms in a welcoming gesture."
	Spatial relationship	Distance, angle, height, etc.	"An avatar is approaching diagonally from your right."
	Accessible vibrotactile feedback	Type of object, type of interaction, relative distance, etc.	Vibration pattern identifies the collision object as an avatar or indicates the distance

Table 10: Alt attributes that developers and designers should consider when developing alt text for their BVI users (Kuligowska, Kisielewicz, & Włodarz, 2018)

It is not, however, without drawbacks. Not all frameworks distinguish between useful and decorative images or whether an object can be interacted with or not. A-frame, an open-source framework for building WebVR applications, includes tags for 3D objects, such as <a-sky>, <a-plane>, and <a-dodecahedron>. However, these tags do not provide any indication for what these objects represent in the virtual environment; they say nothing of whether these objects are important for storytelling, user interaction, or are just there for decoration (Sun, et al., 2023).

Screen Readers and TTS in VR

The implementation of text-to-speech in VR is very interesting. TTS technology is available primarily as a means of communication for someone who doesn't want to use their voice. ElevenLabs, a prominent player in the AI-enhanced text-to-speech sector, allows users to design a voice for themselves in virtual and augmented reality. Mati Staniszewski (2023) from ElevenLabs claims that their TTS "mimics the natural cadence, emotion, and inflection found in human speech, providing strikingly lifelike voices that feel indistinguishable from human speech." They provide options to create a synthetic voice or to clone your own voice use in virtual reality and the Metaverse.

This is a powerful tool for accessibility as it can give the mute the means to speak audibly in VR and AR settings. However, Staniszewski only mentions TTS as a means for a user to communicate with their virtual environment, and never the other way around, meaning that the system is not useful for improving the experience for BVI individuals.

By extension, screen readers are not a common feature implemented into virtual reality systems. Unity and Unreal Engine support for alt-text in their editors, which is a very useful tool for BVI developers (Unity, n.d.; Craven, 2023). However, right now there are no common means for developers to integrate screen readers into their programs for their users to interact with.

Development of Unity Editor Tools

Unity is primarily a game development engine. People from across the globe have used Unity to create some of the most creative pieces of interactive media in history. It is a very well acclaimed development environment that provides a versatile platform for not only game developers but also for those working in virtual reality, augmented reality, and other immersive technologies.

Unity's intuitive interface and extensive asset store have contributed to its widespread adoption, empowering developers to bring their visions to life with relative ease. The ability of developers to create their own tools to fit purposes, both general and niche, has made the platform even more popular.

This extends to the creation of custom tools within the Unity Editor itself. Unity provides a robust framework for developing custom editor tools, enabling developers to streamline workflows, enhance productivity, and address unique challenges.

To delve into the realm of Unity Editor tools, developers can harness the power of the Unity Editor API. The API grants access to the inner workings of the editor, enabling the creation of custom windows, inspectors, menus, and other features (Unity Technologies, 2023). This level of customization is invaluable for tasks such as automating repetitive processes, visualizing complex data, or integrating third-party tools seamlessly into the Unity workflow.

Developers can use C# scripting to extend the editor's functionality. This scripting language is not only integral to Unity game development but also serves as the foundation for creating custom tools. By tapping into the Unity Editor API and employing C# scripts, developers can design tools that cater to their specific project requirements.

The process involves defining custom editor windows, handling user input, and manipulating assets or scenes within the Unity Editor. This flexibility allows for the creation of tools ranging from simple utilities to complex systems that significantly enhance the development pipeline.

The process by which you develop tools for the Unity Editor is slightly different for how you would develop your average game script. For starters, the average Unity C# script inherits from the MonoBehavior class, and its code is primarily meant to execute during runtime.

Editor scripts, on the other hand, have to be in a folder titled “Editor”. This is so that when you build the program, Unity knows not to try and compile these files (Koder, 2019). The scripts themselves do not inherit the MonoBehavior class, but instead inherit either the Editor or EditorWindow class.

Building a tool that inherits the Editor class will appear in Unity’s inspector, and is best suited for tools made to help with the functionality of certain aspects of the scene. On the other hand, tools that are inheritors of EditorWindow will receive their own view window, like the Scene or Game view windows. These tools are good for general functionalities to begin creating or changing large aspects of the scene, but do tend to affect the scene as it is executing.

Research by Kasidy Landry

Before they began their development, the preceding group did research to better understand the struggles that blind and visually impaired people face. They did this so that they could implement features that will most positively impact the user experience. My group and I continued this practice to ensure anything that we added was beneficial to the user as well as a way to find new features to add. The previous group also focused on learning techniques and proper etiquette for programming in virtual reality in the pre-development stage. I, having no experience with virtual reality software development as well, also did this. I continued to do this to develop my knowledge and skills so that I could develop acceptable code for our project.

APIs and SDKs

API stands for application programming interface. This is a set of rules that allow different applications to communicate between one another. An API is essentially a layer between disconnected systems that processes data transfers between them. APIs allow developers to make their project’s functionality and data available to other developers, business partners, others within their company, etc. Application programming interfaces can have several operations that developers can outsource and use in their own work so that they can access specific features from other systems without worrying about how they work.

APIs make a developer’s job easier by providing the foundations for different things that they can use to enhance the program they are creating without spending a lot of time on the minor details of code development. For example, if you are developing an IOS app for apple iPhones and want your app to use the camera on the phone, you do not have to create your own camera interface because the cameras on phones have an API that you can use in your app. By integrating the already existing API, you don’t have to spend time writing code to implement what the API does because it is already done for your convenience and you can move on with your app development.

APIs can simplify integration and management of applications by providing a foundation for communication between different applications. API data exchange is done within the application and is not shown or made known to the user directly and therefore provides the user with a streamlined user interface experience.

SDK is an acronym that stands for software development kit and is a compilation of tools to assist in building software. These tools let a developer integrate a separate program into their application. SDKs have prebuilt features that a developer can simply tack onto their project to implement these new features in their own work.

SDKs can include a compiler, code libraries, code samples, testing tools, documentation, and debuggers. SDKs also usually include an API.

There are two types of SDKs, either open source or closed source. Open source being an SDK that the general public has access to and closed source being an SDK that is locked up and less accessible.

SDKs are similar to APIs in that they are outsourced and integrated into code to save time, add new features, and enhance user experience. However, they both exist for different purposes.

An API is a collection of routines, tools, and protocols that define how components in software should interact. They allow developers to access certain features or services from other applications. An SDK is a collection of several different tools for software development and can include APIs. An SDK can add whole features to an app while an API communicates with outside software.

Magnifying glass/zoom

In one of our weekly meetings, our sponsor, Joey, proposed two new features he would like added to the project. He asked us to research them to find out if they are able to be implemented in virtual reality and, if so, how they can be implemented to the existing project. Joey himself is visually impaired and suggested these features out of personal experience and interest. He believes they will be beneficial to the accessibility of the project and I agree so I am interested to see if they are possible and excited to hopefully implement them. His ideas were a “text to speech” feature and a “zoom” feature.

The text to speech feature would do exactly what it seems; it will be able to narrate any speech in the user’s line of sight in the virtual world. I’m thinking it could be either automatic when hovering over text, or could be activated by clicking a certain button, depending on the user’s settings that

they have chosen beforehand. As for the zoom feature, Joey described it to be similar to the magnifying glass feature on an iPhone. This feature allows a user to have a pop-up box on command that is able to be moved across the screen that will zoom in on the contents behind the box. This is helpful if a user only needs to zoom in temporarily. For example, if they need to see only one object in the virtual world more clearly, or if they need to zoom in to navigate to the settings window to change the more permanent zoom settings. This would be extremely beneficial if a user needs everything to be zoomed in more but cannot see well enough to find the settings menu to change the zoom to their desired percentage.

As a group, we assigned two people to each of Joey's two ideas. Emmanuelle and I chose the zoom magnifying glass feature and Justin and John chose the text to speech feature.

Upon starting my research for a magnifying glass feature for our project, I found that many games created with unity have implemented the same exact feature. This means that it is definitely possible to execute. There are many instances of them being used in 2D and 3D games. I even found examples of it being used in a 3D virtual environment.

In addition to just finding instances of this feature, I also discovered that there are a few tutorials and explanations on how to implement it. There are many ways to get creative with this feature. I saw developers use it as a scope in shooting games, as a camera with the ability to zoom in and out to take photos, as well as just a regular magnifying glass.

During my research, I had ideas for a magnifying glass in our project. I thought our magnifying glass object needed to be different from the

stereotypical idea of a magnifying glass that a detective like Sherlock Holmes might use. Although a magnifying glass like that would be good for theming if it was being used in a mystery game, our goal is to enhance accessibility in virtual reality. My idea on implementing the magnifying glass in our project included a somewhat large rectangular object that the user can move around. I also wanted it to be able to easily zoom in and out on command based on the user's needs. The magnifying glass window should also be easily accessed. The user should be able to deploy it whenever needed with a simple action.

One of the most important parts of our project is its ability to be easily added to an existing application or game by a developer. This allows any virtual reality application to be more accessible. The examples and tutorials of magnifying glass features I found were directly embedded in the application or game and are a part of the system as a whole. In order for this feature to work with our SDK, I needed to find a way to make our magnifying glass feature something that can stand alone and be overlayed into an existing project. The feature will need to be able to interact with what it is being added to in order to be effective. This may have been more difficult than just adding a magnifying glass to any virtual reality application because it needs to be its own separate entity that must be able to work with anything it is added to. This means it has to be able to work with any code environment, no matter the differences in coding style. In order to do this effectively, the magnifying glass must be an object with its code in a separate package that is able to be added and accessed anywhere.

Magnifying Glass Implementation

I began the development of the magnifying glass that Joey requested. I got to work on it for a few days. I viewed a few tutorials on YouTube and tried to follow along and implement what they were building into our project. A simple magnifying glass is straightforward but can get complicated as you go along and want to add more features and make it more streamlined.

I began by creating a cylinder object in one of our virtual reality scenes and flattening it to make it resemble a lens. This will be the main part of the magnifying glass that will display the zoomed in background behind it. Next I put in a camera on the lens so that the camera's movement follows the lens. Then, I lowered the field of view on the camera and modified the camera's position to create the magnification effect. I then rendered the texture on the camera and applied it to the lens. This made it so that whatever the camera sees will be displayed on the face of the lens creating the transparent look of it. I played around with the settings on the camera and moved its position around and tilted it a bit to see the difference each move makes on the magnification and clarity of the world behind the magnifying glass. I moved things around until I was satisfied with the magnification percent, clarity, and size of the lens. After creating the lens and magnification effect, I added a handle and frame to the existing lens to make it look like a traditional magnifying glass. Next, I tracked the field of view on the lens and tested it in the various interactive levels we have in our project. While doing this, I once again tinkered with the magnification effect to see what changes and to figure out what percentage I think is best, though moving forward I will get outside opinions on this as I am not visually impaired or blind.

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

As a group on November twenty-seventh, we met with our sponsor as well as Dr. McMahan who is a professor that specializes in and does research in virtual reality. We discussed many things about features we wanted to implement in the project, mainly the magnifying glass and an alt text to speech feature. He explained that he thinks it would be best if we focused on one feature and made it as effective as we could. Since these features are supposed to help blind and visually impaired people, having many lackluster features is not as helpful as one very well executed feature.

Dr. McMahan also spoke about a possibility of increased motion sickness with the magnifying glass. Since the magnifying glass more or less stays stationary in relation to the headset and the users head movements, looking at the magnifying glass, then suddenly looking away could cause severe disorientation. This is because you are going from looking at an object and/or its surroundings in increased size and immediately looking at something that is normal sized and the magnification discrepancies would not be easy on the eyes or brain. Though games have been able to successfully execute features like this, it would just be another thing we would have to worry about and we do not want to put unnecessary discomfort on individuals, especially those who are testing our project for us since they are doing us a favor.

Having said this, Dr. McMahan asked Joey, our project sponsor, which feature he would like to see implemented most. Joey understandably chose the alt text to speech feature. Because of this, we agreed to table the magnifying glass development and focus on the development of the alt text to speech feature. This is a very complicated and layered feature that will

require a lot of work so splitting it between the four of us instead of the original two we planned is probably for the best.

Research by Emmanuelle Rebosura

Much of the first semester required us to do research: learning and understanding what the previous team has done and accomplished before us, using IRB testing in regards to what has already been established by the previous team to learn what we would be improving on during the next semester, and research on how we would improve these already made features or add new ones that would help increase accessibility in virtual reality.

Blind and Visually Impaired Community

The group we spent both semesters focusing on helping and bringing accessibility to in virtual reality is those who are blind or visually impaired in some way. To describe being visually impaired covers a wide range of visual disabilities, not just blindness, as having a visual impairment is still different from being blind, though blindness is a type of visual impairment.

Being visually impaired is defined as when one's eyesight is unable to be corrected to what is considered a "normal" level, affecting their daily life. It is defined specifically by how it affects one's daily life rather than one's *visual field*– how much one is able to see with their central and peripheral field of vision when they look in a certain direction– or their *visual*

acuity– the ability of one's eye to distinguish shapes and details from a certain distance (think the prescription level of a pair of glasses)– as most people who are visually impaired in some way still have some vision rather than being completely blind, which most people may not realize initially.

Someone who is visually impaired can be someone with very poor vision, or it can refer to someone who is completely blind. Blindness itself only refers to the inability to see– some people who are blind may still be able to see a little bit, which is known as "partial blindness" or "low vision." To be legally blind, either one must have 20/200 vision at best in their better eye, or their visual field must be reduced significantly (about 20 degrees or less in their visual field). People who are completely blind, or have what is referred to as "total blindness," lack any kind of light perception and make up about 15% of those with some kind of eye disorder.

Visual impairment is very common throughout the world. According to the World Health Organization, there are at least 2.2 billion people worldwide with some form of visual impairment, with almost half of them having been able to prevent the development of the visual impairment. Of those people, roughly 43 million people are estimated to be blind.

Eyesight is seen as one of the most important senses– most people without any of their senses impaired tend to rely most on their eyesight more than any other senses (hearing, touch, taste, and smell). With vision impairments, those who are in the Blind and Visually Impaired community require other means of gaining information about the world without having to rely on eyesight. Due to this, all the other senses become heightened and much more important in order to gain information about the world around us.

There are several ways people who are blind or visually impaired are still able to navigate daily life without vision, such as using a cane, having a guide dog, or having someone who is able to guide them. Physical text such as signs, papers, or books can be read using Braille, a language made up of raised dots in certain patterns that can be read using one's fingertips. Hearing is also often heightened in those with poor and low vision, as hearing is considered the second most important sense.

With technology, though, there are many ways blind and visually impaired people are able to navigate it.

Technological Accessibility for the Visually Impaired

For blind and visually impaired people, there are two categories of technology from their perspective:

- General technology (Computers, smart phones, etc., the technology people most often use without thinking twice)
- Assistive technology

Assistive technology is technology specifically designed *for* people with different disabilities to use. In the case of those in the Blind and Visually Impaired Community, assistive technology includes screen readers to relay to the individual through audio what is on the screen so that they do not have to see or read it themselves, or screen magnifiers to magnify and enhance what is on the screen for those with low vision.

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

When designing digital content, such as websites, developers should take care to make sure their content is accessible to a wide audience. When it comes to making digital content accessible for those in the Blind and Visually Impaired community, there are several ways this can be done. These include:

- The ability to adjust brightness
- Color contrast that allows for more readable text
- The ability to adjust text size and element size, or making text large enough for those with impaired vision to read
- Readable fonts (for example: avoiding using “fancier” or “curly” fonts
- Being aware of blocks of text in certain text styles being easier to read than others (bold text is easier for people to read compared to underlined text and italicized text)
- Avoiding an excessive use of typing in all capitalization, as this can also be difficult for users to read
- Being aware of spacing so that it is easy to read
- Creating a way to distinguish different elements (e.g. making it easy to tell the difference between a header and the main text on the same page)
- Rewriting any stylized texts in regular fonts so that it is more readable for both users and screen readers
- Adding alt text to important images

Why should developers care about making their websites and software accessible for those who are blind or visually impaired? Developing *any* kind of software to allow for accessibility is important for many reasons:

- By making software accessible, the content made is able to have a much broader reach to allow more users and possible customers.
    - If accessibility is not kept in mind, many people are not able to interact with the software. Revenue and business may even be lost if people find it is not accessible, including loss of business from government organizations.
- If accessibility is kept in mind during the initial development, then developers do not have to worry about rushing to add accessibility features.
    - The Americans with Disabilities Act, or the ADA, legally requires commercial websites that are published to conform to certain specific standards. Failure to have an accessible website may lead to lawsuits.
    - While this law applies specifically to websites, keeping accessibility in mind from the start makes implementation easier than having to work it in after everything is developed.
- While accessibility features are important for those with disabilities, people without disabilities can also benefit from accessibility features. There are many reasons one may need, or even *want*, to use accessibility features within software, which only helps benefit the user.

There are simple tests and questions developers can use and ask themselves to ensure they are including blind and visually impaired people in their audience, such as checking if there is a focus on all the important elements or if all the controls can be accessed using just a screen reader. In fact, there are basic guidelines a developer can follow, known as the Web Content Accessibility Guidelines, or the WCAG.

The WCAG standards explain to web developers how to make websites more accessible to those with disabilities, with certain guidelines and success criteria that developers should follow when developing websites. The current latest WCAG standard is WCAG 2, which has been developing since it was first published in December of 2008; the latest updated version, WCAG 2.2, was published in October of 2023. As of then, the latest standards include thirteen guidelines and twenty-seven success criteria.

As stated by the ADA Site Compliance, developers should understand the inclusive design principles as follows, as based on the WCAG 2 web accessibility standards:

- Perceivable: information should be able to be perceived from how the components are presented
- Operable: users should be able to use and operate with the interface with ease
- Understandable: the layout should be intuitive and easy for the user to navigate
- Robust: content should be designed to fully utilize platform accessibility services to ensure it works well with assistive technology

While many of these standards are designed for websites and web development for computers and mobile devices, websites are not the only technology that should apply and implement accessibility. We are progressing technology in entertainment as well, as seen with Virtual Reality, and those in the blind and visually impaired community should be able to keep up with those with sighted people when it comes to this.

One example of entertainment that now has accessibility for visually impaired people is television– while visually impaired and blind people can hear the lines and dialogue from the television, it does not necessarily convey everything that happens on screen. However, there are devices now that allow for a video description to be read for the visually impaired to listen and follow along with what is happening on the screen.

Every person is different, so one person who is visually impaired may have different needs compared to someone who is also visually impaired– for example, one accessibility feature may be useful and necessary for one visually impaired person, while another may not need it all. It is important to have a wide range of features in order for technology to be accessible to as many people as possible so that any group of people do not end up left out as our technology continues to advance.

People who are blind or visually impaired can also use screen readers to read out loud text on a screen or the alt text of an image on a website, assuming alt text is added. Screen magnifiers also exist that can be used to enlarge the content on an electronic screen, making it easier to view.

With our project, while we are using Virtual Reality rather than making a website, there are still many aspects of ADA Site Compliance that apply to our goals since virtual reality still uses screens– some of what applies to our project include color contrast and large font size or the ability to change font size.

We had originally planned to include a zoom function that would allow users to magnify parts of their surroundings, but we had eventually learned of some problems with using a zoom feature in Virtual Reality that would have been more detrimental to the user rather than helpful. The ideas that we are now currently aiming to add to make Virtual Reality more accessible to those who are blind and visually impaired are alt text for objects and a type of screen reader to read text and alt text, which we hope will function just like the tools used by many people in the Blind and Visually Impaired Community.

Zoom Function

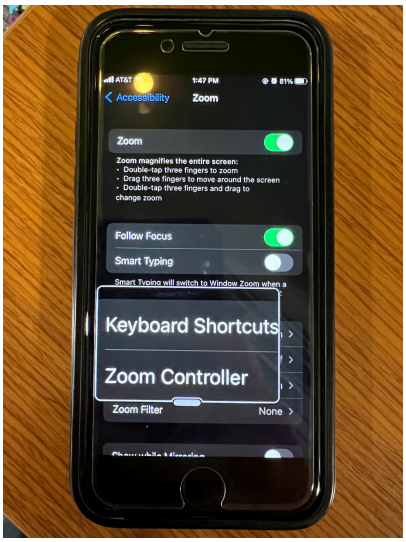
One feature we were originally looking into adding to our project is a zoom feature, similar to the zoom accessibility feature on the iPhone.

This is a very useful accessibility tool on the iPhone for those with impaired vision, magnifying either the entire screen or part of it to make texts and images on the screen easier for the user to see. To turn on this feature on the iPhone:

- Open the settings app
- Tap Accessibility
- Tap Zoom
- Turn on Zoom

When zoom is turned on, the user can double tap the screen with three fingers at once, which will activate the Zoom accessibility feature. By doing this, the zoom feature will either zoom in the full screen that the user

can move around using three fingers, or have a little window pop up that the user can drag around the screen to zoom into a certain spot on the screen, which can be seen in the image below.



(Figure 11) *An iPhone 7 using the zoom accessibility feature with Window Zoom*

There are also several different settings that the user can adjust that may make the zoom accessibility feature more useful to them. These include:

- **Follow Focus:** When switched on, this will have the window zoom follow the user's typing, text selections, and text insertion points.
- **Smart Typing:** If the user is using Full Screen Zoom, this will switch to Window Zoom when the user starts to type.
- **Keyboard Shortcuts:** These are different shortcuts for the user to adjust or toggle the zoom feature using an external keyboard.
- **Zoom Controller:** Turning this on allows for quick access to zoom controls and set certain actions.

- **Zoom Region:** This allows the user to choose whether to use Full Screen Zoom or Window Zoom.
- **Zoom Filter:** This allows the user to put on a filter when using the zoom feature, such as inverted, grayscale, grayscale inverted, or low light.
- **Show while Mirroring:** Turning this setting on will have the zoom appear while sharing the phone screen.
- **Maximum Zoom Level:** This allows the user to drag a slider to adjust how much the zoom feature will zoom in on the screen.

The type of zoom feature we initially thought to have is one similar to the windowed zoom. We wished for a user in virtual reality to be able to drag around a small window in their environment that will allow them to zoom in on certain areas of their vision, making this environment easier for them to see. While our zoom accessibility feature was not planned to have been as comprehensive as the iPhone's, we had originally expected it to be very impactful and useful for those using virtual reality.

Our initial idea was that creating a zoom accessibility feature seemed very easy and possible for our project. Many video games made in Unity also have zoom in features, such as games that use a zoom in for a sniper scope for players to narrow in on and view targets from far distances, which is how we originally thought we would implement this feature. There are also several tutorials that can be found on how to code these using Unity.

This can also be done for virtual reality games as well– one tutorial describes how to create a sniper in a three-dimensional virtual reality environment with this feature attached to an object the player can hold in virtual reality. While this tool was not completely what we were searching

for, we thought that we could use the idea presented from this tutorial with any necessary adjustments to work with our goal to create a zoom accessibility function in virtual reality, before realizing why a zoom function in virtual reality would not be as helpful as we believed.

Research by Justin Morera

Text-to-Speech/Screen Readers:

One of the proposed new features to implement in the kit is a text-to-speech function that will allow visually impaired users to have menu text, and possibly object descriptions (such as the ones from the partial vision tool) read aloud to them, just as how screen readers do on other devices, and it is also possible that this feature can be made to work with pre-existing screen readers.

Preliminary research already shows a strong need for this type of feature in virtual reality, as expressed by Jesse Anderson with Equal Entry, “Magnification and menu narration features are often critical for blind and low vision users. Some low vision users, like myself, often use a combination of magnification and narration / screen reader. Blind users not only require a screen reader or menu narration but also a control method for locating and navigating VR controls” (2022). It was also noted that although these types of accessibility features are quite new, research is being done and technology is moving toward improving accessibility in virtual reality by way of features such as screen reader support.

From an implementation standpoint, it is possible that the best way to make this type of feature work is to provide adequate alternate text or "alt text" to all objects and entities in a virtual environment. This alt text is a sort of metadata that a screen reader can pick up on and use to generate audial descriptions of the objects or menus being interacted with.

The challenge is adding sufficiently descriptive alt text to objects, and also applying it to as many objects as necessary, without overloading the user with too much auditory feedback. One of the previously mentioned research groups stresses the risks of engaging a user with auditory overload (Sun, 2023); they implemented a version of alt text for screen readers tied to a haptics tool similar to the one in this project where only the alt text of objects that collide with a user's controllers' raycast will be read by the screen reader. This suggests that limiting the text-to-speech feature only to menus and objects targeted with the partial vision tool may be an important limitation to protect overstimulating users.

Environments like Mozilla Hubs seem to be a great model for accessible virtual environments, as reviewed by Rhea Guntalilib (2020). The virtual meeting room hosting application is noted to have a very easily accessible feature both in virtual reality and on their website, which work very well with a screen reader and other visual impairment settings. This could be a good environment for us to test in order to see how similar accessibility features can mesh together in an effective manner, inspiring our team to see what is critical for visually impaired and blind users in different virtual settings.

Furthermore, the generation of descriptive and plentiful alt text is not completely unguided; because this feature has been around in other media

for a while and idea itself is not new, there are guides, such as Microsoft's guide for writing effective alt text, that outline how to write properly descriptive alt text, while also demonstrating when it is necessary to include alt text, and just as importantly, when alt text should be omitted (Microsoft, 2021). Even though these guidelines are made for web development, the principles behind the practice should be very easily translatable to virtual reality development. Of course, research will certainly continue into the feasibility and the implementation of this feature, and as our knowledge of the matter grows, we will be able to move closer to implementing another beneficial tool designed to aid blind and visually impaired individuals effectively.

Implementation Plans

As mentioned briefly before, our projects has the opportunity to capitalize on C#'s Object-Oriented properties; this can be utilized maximally when considering how to tag virtual entities for text-to-speech as well as for modularizing the other tools such as the haptics and partial vision tools. By consolidating scripts for the tools into their own respective classes, we can implement methods to interact specifically with objects of a pre-defined type. The idea is that we create an interface (or perhaps multiple interfaces) that any other object or entity in a virtual environment can inherit to universally gain the tags required for each accessibility feature. Each tool's interaction will be explained in the following paragraphs.

Sonar

The sonar tool utilizes a generic object's physical dimensions to return auditory feedback to the user upon collision; therefore, any physical object in the environment should be able to return this feedback without additional tags. This makes integrating an interface with the sonar feature in mind technically unnecessary; however, it is certainly possible to add additional functionality such as combining verbal descriptions or changing sounds for specific objects when targeted. For instance, if a developer wants certain character models or objects, such as doors, to return a unique sound with the sonar device, adding the custom accessibility interface to that entity’s class will add the option to apply additional tags to the object that the sonar tool will automatically check for.

This feature was planned for integration with the accessibility tags, but we chose not to continue developing it in favor of ensuring the portability of the partial vision tool and tagging systems.

Haptic

The haptic tool is easily one of the more obvious candidates for a tagging system via interface due to the vast number of features it includes. By inheriting an interface with optional fields for various customizable tags, developers can mark objects in their environments as “waypoints”, signifying milestone objects or locations for story or environmental progression; label interactable objects like doors, buttons, or levers; or even apply additional haptic tags to objects if developers desire alternative means to differentiate objects of a certain type. For each tag, an object will output a different

pattern of haptic feedback when targeted by the tool, removing the need for specific waypoint objects and premade doors that users need to import with the kit.

This feature was planned for integration with the accessibility tags, but we chose not to continue developing it in favor of ensuring the portability of the partial vision tool and tagging systems.

Preparation for Testing

The plan for testing was to code in a way to get quantitative metrics for how often the Sonar and Haptic features are used.

Whilst using the existing solution, there was a point where we tried to navigate the level with our eyes closed. We were about as unsuccessful in this endeavor as one might expect, as we all lack the experience in using our senses other than sight to navigate that a BVI individual would have built up over a significant period of their life.

However our sponsor, who is a member of the BVI community himself, said that despite his own experience, he had areas in which he got stuck in without the use of his eyesight.

We decided that it would be useful to have a visual representation of areas where features were used the most. It would give us a clearer picture of:

- What features are used the most
- What kind of obstacles give the most navigational trouble
- What features are commonly used together (and therefore pair well)

Testing Questions

To assess the usefulness of the existing product, Joey and the rest of the team prepared the following questions for our testing volunteers:

- How impaired is your vision?
    - Not impaired at all
    - Colorblindness
    - Low vision
    - Moderate vision impairment
    - Severe vision impairment
    - Blindness
    - Other (Specify)
- Have you used virtual reality before
    - Yes
    - No
- If yes, how difficult was it for you to use virtual reality?
    - Very easy
    - Easy
    - Neutral
    - Difficult
    - Very difficult
    - I haven’t used virtual reality before
- What are some challenges you face when using virtual reality?
- How easy was the sonar tool to use?
    - Very easy

    - Easy
    - Neither easy nor difficult
    - Difficult
    - Very difficult
- Did you find the sonar tool to be effective?
    - Yes
    - No
- Did you find the sonar tool's feedback to be easy to interpret?
    - Yes
    - No
- What did you like about the sonar tool?
- What did you dislike about the sonar tool?
- How easy was the haptics tool to use?
    - Very easy
    - Easy
    - Neither easy nor difficult
    - Difficult
    - Very difficult
- Did you find the haptics tool to be effective?
    - Yes
    - No
- What did you like about the haptics tool?
- What did you dislike about the haptics tool?
- How realistic and natural does the haptics tool feel compared to assistive technology in real life (such as a walking stick)?
    - Very realistic and natural
    - Realistic and natural

    - Neutral
    - Not realistic or natural
    - Not at all realistic or natural
- How easy was the partial vision tool to use?
    - Very easy
    - Easy
    - Neither easy nor difficult
    - Difficult
    - Very difficult
- Did you find the partial vision tool to be effective?
    - Yes
    - No
- What did you like about the partial vision tool?
- What did you dislike about the partial vision tool?
- How similar does the partial vision tool feel compared to navigation in other digital settings (mobile or web browsers)?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - Not at all
- How easy were the menu accessibility features to use?
    - Very easy
    - Easy
    - Neither easy nor difficult
    - Difficult
    - Very difficult

- How useful did you find the menu accessibility features to be?
    - Extremely useful
    - Very useful
    - Somewhat useful
    - Not very useful
    - Not at all useful
    - I didn’t use the menu accessibility features
- What are your thoughts on the high contrast feature?
- What are your thoughts on the font size adjustment slider?
- How similar do the menu accessibility features feel compared to navigation in other digital settings (mobile or web browsers)?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - Not at all

Automated Data Logging

To glean some extra information from our testing, we thought that it would be best to implement some level of data logging and analysis into the existing code. The primary tools that could be analyzed using such data are the sonar and haptic tools. Knowing where and when these tools are in use would give a decent picture of how useful these tools are in navigation.

Method 1: Custom JSON Logs

We created a class with the ability to write and read logs of when the sonar and haptics tools were firing. This was a simple enough process; all that was needed was to add a function call in the RayCone.cs and WayPointHaptics.cs files which would take care of logging the information as the tools are used.

We collected the following information:

- The logged action (sonar raycast, sonar rayhit, haptic impulse)
- The time of logging
- The position the logged action took place in

All this data would be stored in JSON format for easy interpretation.

Data Visualization

Once we had the logged data the next step was deciding how to format the data in a way that would be easy for us to interpret and use.

Method 1: Heatmaps

It seemed like the simplest way to represent any data received in testing would be to use a heatmap. Using this method, we would have a clear, visual representation of all the points at which a feature was used.

Method 1.1: kDanik's Unity Heatmap

In terms of existing solutions, the closest tool we found was a project under development by Daniil "kDanik" Kurachkin on Github. In its completed form, the tool would use the Unity event system to store logs of data of the activation of custom events. These logs would then be used to show an overlay of the most often places that these events occurred.

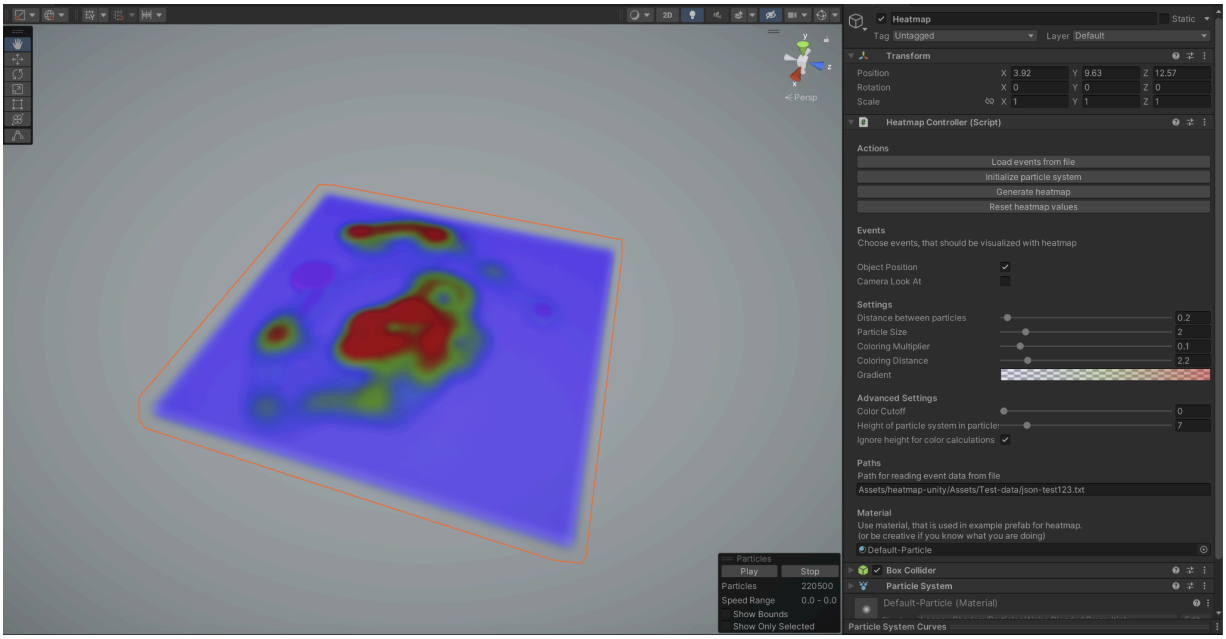


Figure 12: *A heatmap describing the frequency of a logged event's occurrence in a specified location. Red areas depict high frequency. (Kurachkin, n.d.)*

Our thoughts for this method was to use this project to log testing data to see where and how often the haptic and sonar tools are being used. We would then display this data over the levels using a heatmap.

However, we decided that the effort required to make the tool usable was not worth the amount of time that it would take. So we took ideas from the project to create a more crude version of the heatmap tool.

Method 1.2: Custom Logging and Pseudo-heatmap Generation

We used the aforementioned JSON logs to store the data of sonar and raycast usage. We then created a script that would read these logs and use a Unity gizmo (official term) to visually display that data within the scene.

We inserted a script into the scene that would read all the data and store them in IEnumerable lists. The script would then draw color-coded, semi-transparent cubes in the space these tools were used, creating a pseudo-heatmap. The information could be toggled on or off using the script or by disabling gizmo draws.

The issue with this method was that it was very resource expensive. It used too much memory in the Unity editor, likely because it would keep trying to reread the data and redraw the pseudo-heatmap. We came to decide that the heatmap was costing us more time and effort than it was worth, and decided to take a more traditional approach.

Method 2: Scatter Plots

Rather than continuing to work on developing a heatmap system, we decided that it would be easier to export the logged data into Google Sheets

format. From there the data can be made into different scatter plots representing a top down view of the level.

While we have data for the y-position of the logged events, it is largely irrelevant for our analysis purposes and so it will not be represented in our scatter plots.

Testing Results

Although we only had four testers, we were able to obtain substantial results that will help us immensely in the development process. The respondents consisted of one colorblind individual with no other visual impairments, two low vision individuals, and one individual with moderate vision impairment.

Of the survey responses received, it was clear that the sonar tool was the most difficult tool to use, but all users found it to be effective once they got used to it, and most users found it to be simple to interpret. In fact, all the tools and features tested were found to be effective and helpful by all users.

The haptics tool was found to be generally easy to use, but 100% of users gave a “neutral” response when asked if the tool felt comparable to real-life assistive technologies, such as a walking stick. The partial vision tool was less easy to use, according to respondents, but none of them found it to be difficult overall; however, all users found the tool to be at least moderately similar to navigation tools in other digital settings. And finally,

the menu accessibility features were found to be very easy to use and all users found them to be the most helpful they could be, while also stating that the features were very similar to navigation in other digital environments.

These findings helped us determine which tools were not yet ready to be added to a development kit, and which tools appeared to be the most promising and ready for the next stage. Of course, more testing would have been beneficial, but considering the time constraints tied to this project, even the gentle nudge provided by this data was enough to put us in the right direction. At the very least, we knew which tools to prioritize during development. Pie charts showcasing some of the major findings are present in the Diagrams section.

Diagrams

Testing Results

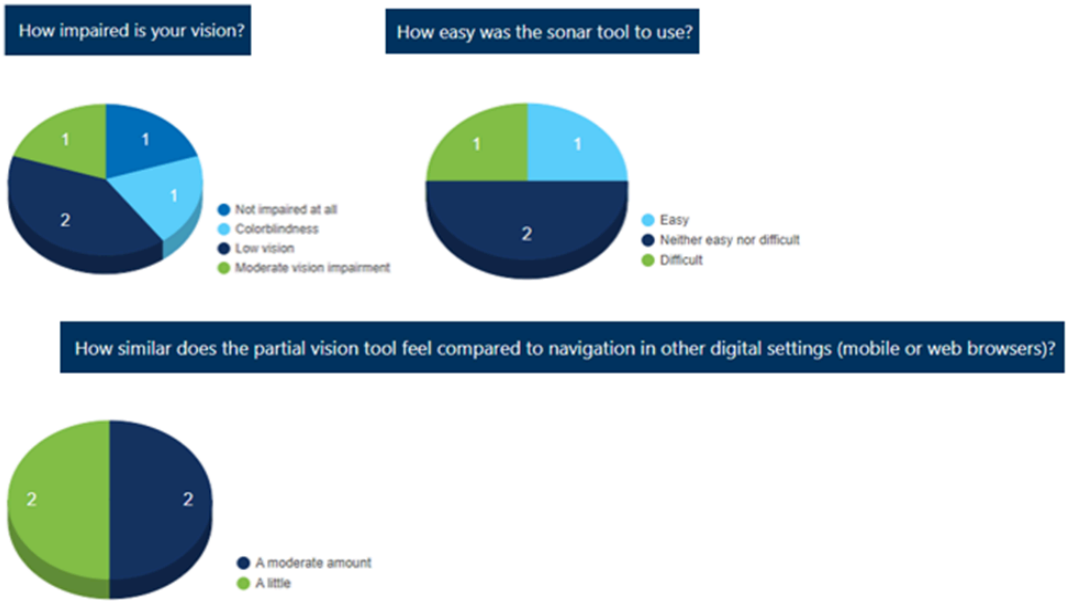


Figure 13: Pie charts generated based on our survey data using SuperSurvey.

Division of Labor

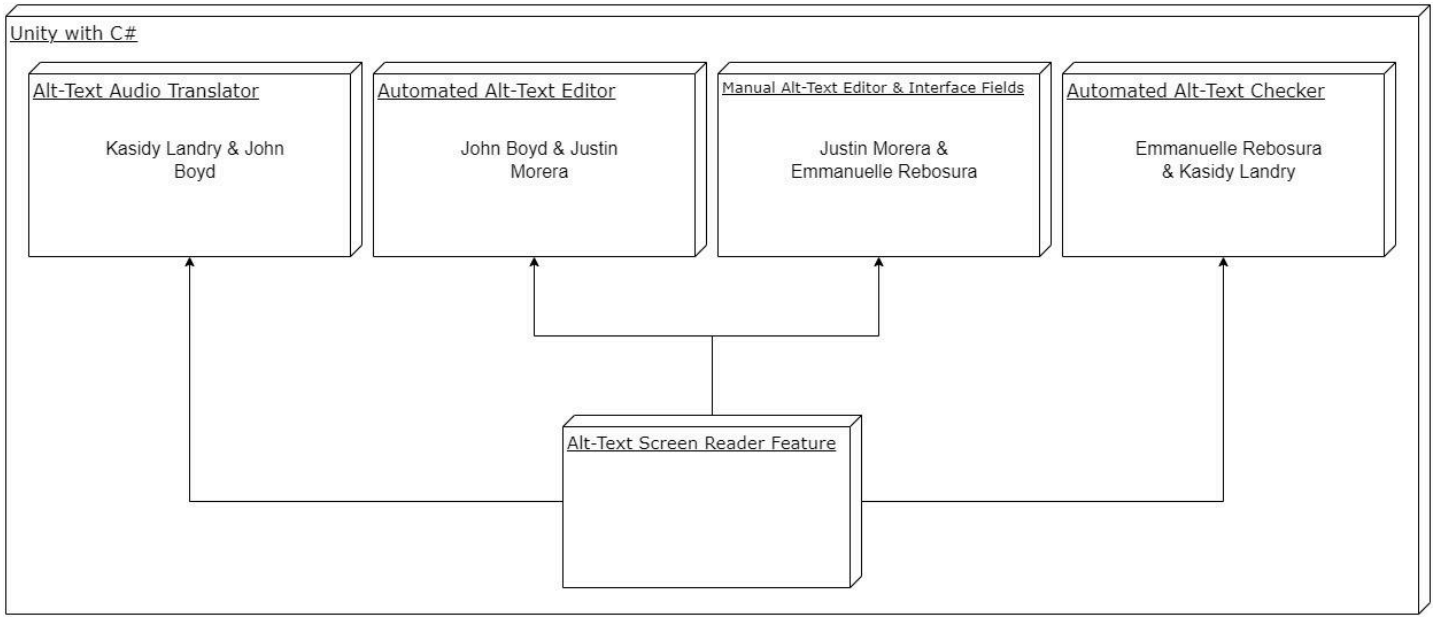


Figure 14: Division of Labor Block Diagram

This diagram illustrates how our team distributed the four main aspects planned for this project; each team member chose to split their focus between two parts, so that each part had two developers that could communicate with each other and “tag-team” their development.

Furthermore, by having each aspect of the tool connected to two members, and not allowing any two members to work together on the same subprojects, we ensured that every team member had at least some hands-on knowledge about at least three major aspects at any given time.

For instance, John and Justin were both planned to have worked on the Automated Alt-Text Editor, but John also worked on the Alt-Text Audio Translator and Justin also worked on the Manual Alt-Text Editor and Interface

Fields. In the event that these two developers needed further insight into the Automated Alt-text Checker, they were both in direct communication with Kasidy and Emmanuelle, respectively, who both worked on that subproject. This put any given part, at maximum, one degree away from any other.

Development Process

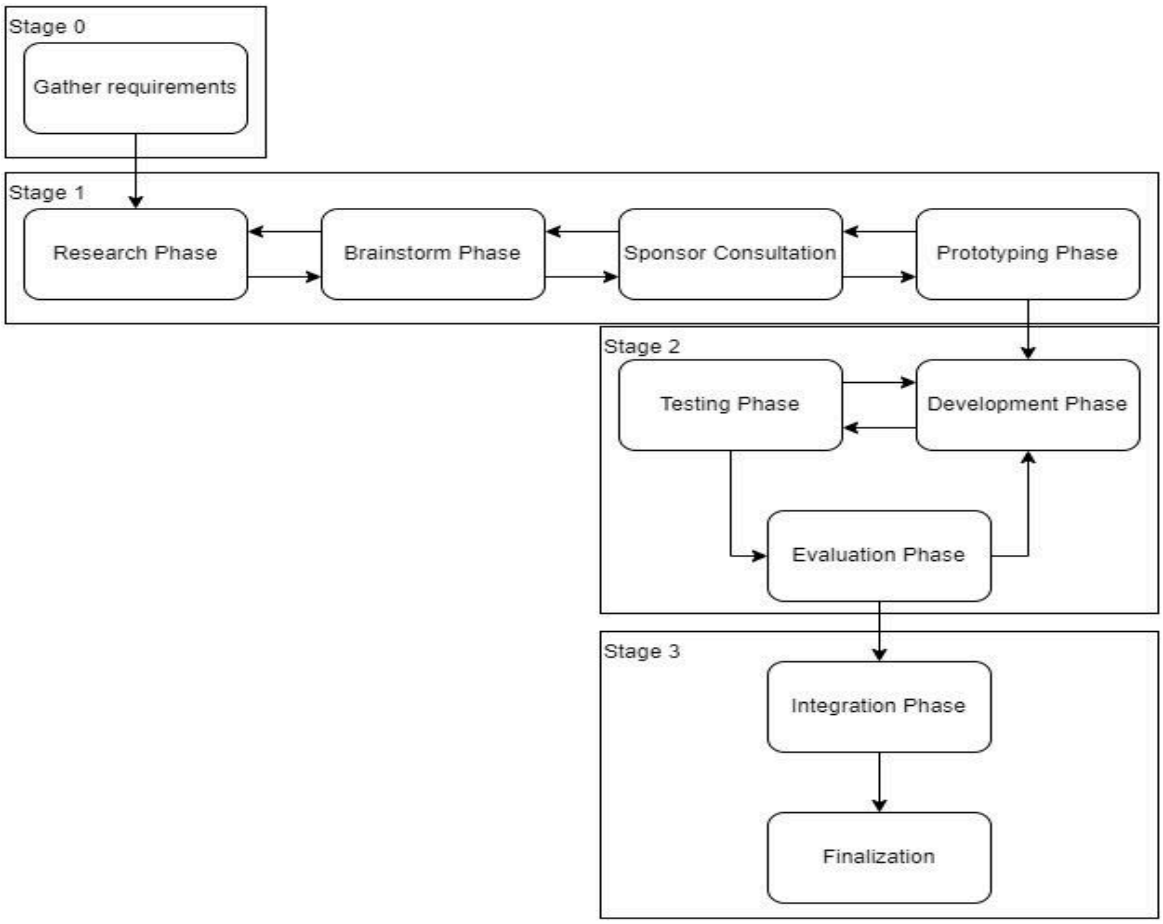


Figure 15: Development Process Activity Diagram

This diagram shows the general path we planned to take when developing new features for this project. We always strived to adhere to this

ordering as feasibly as possible. Stage 0 is arguably the most important phase in which we made sure we understood fully what was expected of us from our sponsor and professors. Stage 1 was all about understanding and planning the implementation of the feature. Stage 2 was tailored toward actually putting together the product and seeking approval from our sponsor. And Stage 4 was naturally implementing the final version with the rest of the application and ensuring there are no outstanding conflicts or unforeseen bugs that arose after setting the new feature out into the open environment.

Use Cases

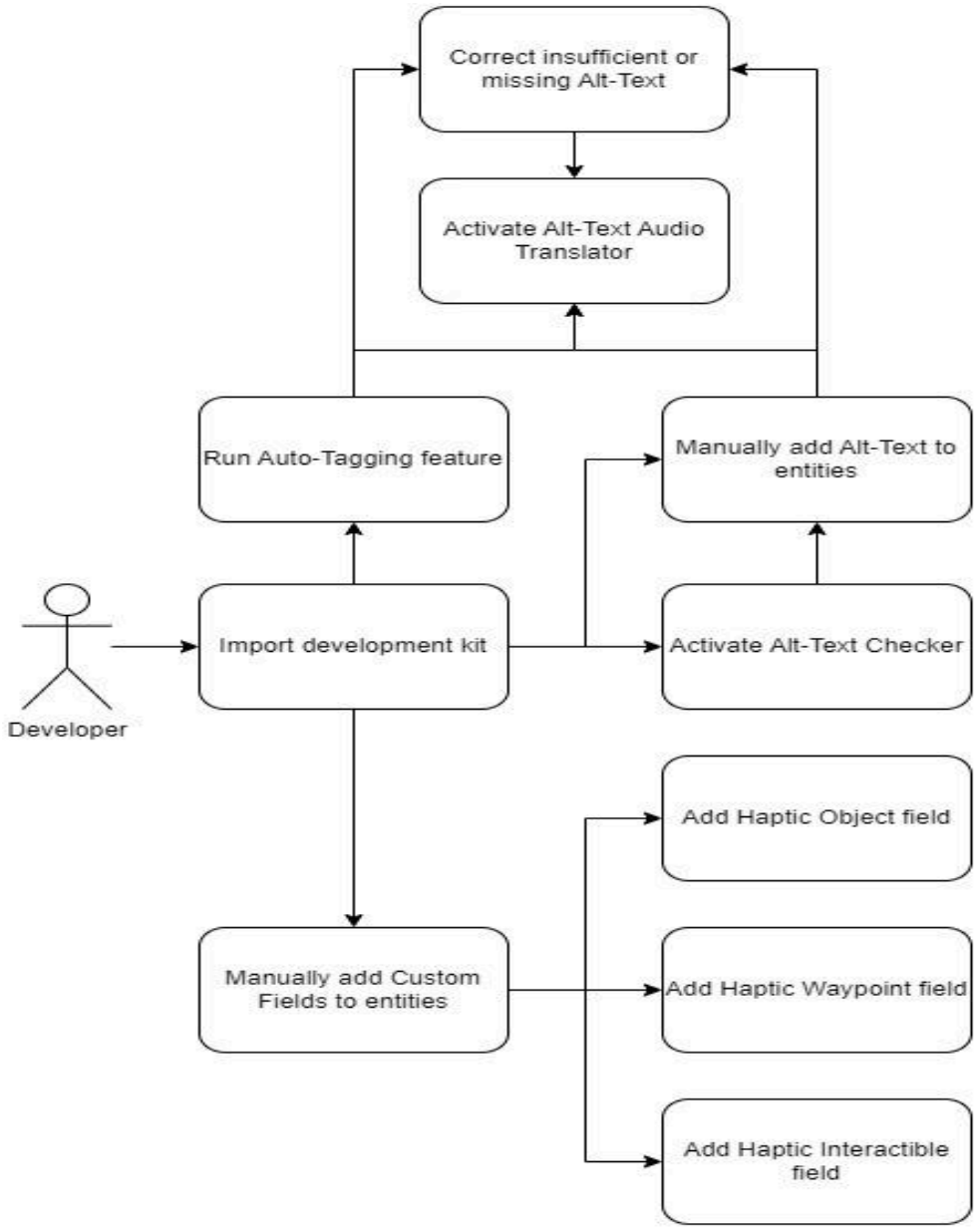


Figure 16: Use Case Diagram

This diagram demonstrates the features a developer obtains by importing our development kit. The other previously developed tools can be ported over by adding custom fields to objects that the tools look for and interact with. The priority feature, the alt-text translator, has multiple modes of implementation in which alt-text can be generated automatically and manually. Additionally, there is a feature that checks every object for insufficient or missing alt-text. Once sufficient alt-text is added to an object, the translator reads it out loud during runtime.

Weekly Contributions

Kasidy Landry

October 16 – October 20

My group's main objective-per our sponsor-is to await the results of the IRB testing being conducted to find out what works and what doesn't in the existing project before we start working on it. Joey, our sponsor, asked us to come up with specific questions and deliverables we want to get out of the testing being conducted. Therefore, this week, I came up with ideas of what I want to know from the IRB testing, typed them up, and added them to a google doc with my teammates' ideas as well. We sent this doc to Joey so he could look over it before we discuss it in our meeting on Thursday.

On Thursday October 19, I, as the project manager, led the weekly meeting. This meeting included Joey and we discussed our testing ideas and then Joey introduced ideas he had. Since we are still waiting on the IRB testing results, Joey suggested we research two features he would like to add to the project. He explained that he wanted a text to speech feature and a feature similar to the magnifying glass that iPhones have. He wants us to research the possibilities of these features and how we can use them in VR. After Joey left our meeting, our group discussed the two features, and we assigned two people to each feature. I chose to research the zoom/magnifying glass feature, so that's the main thing I will be working on this coming week. Also, every meeting I maintain a spreadsheet that includes the itinerary for the meeting, the attendance, a list of things to do the week after the meeting, a link to the minutes that Emmanuelle types during the meeting, and the date of the next meeting all in one place. Therefore, after this week's meeting, I recorded the attendance, added items to the to-do list, and added the date of the next meeting which we all agreed upon during the meeting.

October 23 – October 27

This week we were tasked by our sponsor to research two features that he would like us to add to the project. We each took one of them and researched them. I researched a zoom feature. Our sponsor wants us to be able to add a zoom feature that resembles that of the iPhone, essentially a magnifying glass. It will be a popup window that can be moved around to zoom in on certain parts of the field of visibility instead of zooming in on everything. I researched how to do this in Unity and found references for 2D

and 3D unity projects that used a similar feature. I also found tutorials that walk you through developing a magnifying glass. During our meeting, I reported my findings to Joey, explaining that this feature is possible and probably a quick and easy thing to implement.

I also created the agenda for the weekly meeting, ran the meeting, and spent a lot of time writing my individual design document.

October 30 – November 3

This week I continued researching a new feature to add to our project. It will be a magnifying glass type zoom feature. I researched how it can be done in 3D in unity. I also researched how we can make it something that can be put over another game's code. Our project is an API to make VR games more accessible so having a feature that only works in our testing game simulation is not that helpful. It needs to be flexible with the ability to be overlayed on any 3D unity project, so I am finding ways to do that.

Additionally, since I chose to work on the partial vision tool that the previous group created, I have also been researching that. I looked at their existing code to understand what they did, then I thought of ways to improve it. I did this by brainstorming and researching partial vision tools.

November 6 – November 10

This week I focused on the previous group's repo. They did not have good organization and hardly any comments, so I went through it to try to

understand what everything does. I tried to organize things a bit better by adding comments and moving things around. I also played the VR game that they created so that I could get a better grasp of the code in action. Additionally, I did some practice problems using C# as I am still learning, but getting better. I also watched some YouTube tutorials on a magnifying glass feature I want to add to the project soon. I did not get a chance to add the feature yet but I am hoping to start that this weekend.

November 13 – November 17

As per our sponsor's request, I began adding a zoom feature to our VR simulation. This feature will look and act like a magnifying glass when it is completely finished. I first created a flat cylinder object in our VR scene that will be the magnifying glass lens. I added a camera to the lens so that when the lens object moves, the camera moves with it. I lowered the field of view on the camera and modified the camera's position to create the magnification effect. I then rendered the texture on the camera and applied it to the lens so that what the camera sees will be shown in the lens. I changed the settings on the camera and moved it around, and tilted it a bit to see the difference it makes and to find a look that I like best for the magnifying glass. I have a good foundation for this feature but it is not completely finished yet.

November 20 – November 24

I worked on our unity project by continuing a magnifying glass I started last week. I added a handle and a frame to the existing lens to make it look like a magnifying glass. I also tracked the field of view on the lens and tested it in the various levels we have to find a magnification percentage I think is best. I also went through the project to add alt text to objects in the environment so that we can use that info when we begin adding the text to speech feature later on.

November 27 – December 1

I was going to continue implementing my magnifying glass in the project, but we met with a professor this week who specializes in VR per our sponsor's request and got some feedback and ideas from him. He suggested with the other features we want to implement being complicated and lots of work/research that we should focus on those instead of the magnifying glass, because the other features are more practical and useful. So, this week I shifted and I began researching one of the ideas he gave us. This being an AI to automate alt text for in game objects. Since we are developing an API, we cannot just manually type the alt text, so having an AI be able to recognize objects and describe them would be helpful. This feature will be an addition to the text to speech feature my other group members have begun working on.

February 5 - February 9

This week I narrowed down the information we will be putting in our presentation and began creating the presentation for our group. I also began writing the script for the CDR. For the project, I began implementing the automated alt text checker for our text to speech feature. I also worked a little on the speaker part of the tts feature.

February 12 - February 16

I didn’t do much this week for our project because I was focused on my other classes and personal stuff unfortunately. However, I was able to work on our presentation slides and the script for our CDR. I also coded a little bit for our automated alt-text checker. I hope to really focus on that feature next week and hopefully get it completed.

February 19 - February 23

Most of the time I spent on this class this week was spent on the CDR presentation. I was cleaning up the slides and practicing what I will say. We were supposed to be presenting next week so I was focused more on that than the actual project for the time being.

February 26 - March 1

This week I worked on the auto checker for the alt text in our vr project. I wrote code to check the alt text of game objects to make sure that they have alt text. It also checks if the alt text is the same as other objects. And checks to make sure the alt text is not too short. I did a little bit of testing but not much. I'll continue testing next week.

March 18 - March 22

This week I worked more on the auto alt text checker feature. I wrote more code to check for different things. I checked for repeating objects or for objects with lots of similarities and notifying the developer to make sure this is what they intend. I also checked to make sure the alt text is long enough to be accurate and helpful. I also tested each part of the alt text checker and continued this afterwards.

March 25 - March 29

This week I finalized everything for the demo I was working on. I made sure everything I worked on was working in the whole project without error. After the demo I helped my team try to figure out what went wrong with the demonstration on the vr headset. After tinkering with the headset we got it working again and are prepared for our rescheduled demo.

April 1 - April 5

This week I worked with my group to fix the problems we were having with our code and vr headset. We tried finding open source games we can add our SDK to but a lot of them were incomplete or had errors so we couldn’t add our SDK in without error. We have one game that is able to have our SDK added to it so we tested that a lot this week. We also found out why we had the problem with our headset glitching and figured out how to stop it.

April 8 - April 12

This week I worked on the design document, adding what I’ve done since we finished it last semester. I also wrote my script for our video submission and began filming it. I also edited my group’s video to be turned in for showcase judging.

Emmanuelle Rebosura

October 16 – October 20

This week, I emailed our sponsor, Joey Down, a document of the testing ideas our group came up with during our last group meeting for IRB testing regarding the original project. I also emailed the previous project manager, Logan Blaire, asking for documentation or a guide on the Github repository for the previous version of the project. All the information he gave me about it I shared with my team. After our sponsor meeting this week, I was assigned to research the possibility of adding a zoom function in virtual reality, one similar to the zoom function on an iPhone. While I haven’t been

able to do as much research as I'd like so far due to personal life, I *have* begun research for this.

October 23 – October 27

This week, I did more research into the iPhone's Zoom accessibility feature to understand what my group's sponsor is thinking of for a zoom function in VR, the possibility of doing so in VR, and on people who are blind or visually impaired, as that is the group we are working to help with our project. I also did a substantial amount of work for the individual design doc, which I will be finishing tonight.

October 30 – November 3

This week, I looked over the repository with the team, as well as getting batteries for our VR headset's controllers, as we found that the batteries already were dead when we attempted to test using it. I also decided that for my role, I will focus on the auditory tool, until we hear updates on the IRB testing and learn the results and figure out how that'll affect next semester. My group also decided we want to have a meeting with the previous project manager so that he can guide us personally through the repository, so I emailed him asking if he is available for a Zoom meeting with us. I am currently still waiting for a response.

November 6 – November 10

This week, I set up several Zoom meetings for the team, including our usual bi-weekly meeting. I also finally got a response from the previous project manager, and while we weren't able to meet up with the original time our team wanted, we were able to find a time that worked for everyone and managed to get some good information, especially when I asked about how the previous team did version control in Unity, as that was my biggest concern given my current version control problems with Unity this semester in another class. I also set up a Zoom meeting for November 27 to meet with someone who can help us and give us tips and advice for our project.

Right now, we can't do much as we're still waiting for the IRB testing date to be settled, so the most we can do right now is research. I did a bit more research on the Blind and Visually Impaired community in order to clean up and fix what I had originally written in the Individual Design Doc, as it was pretty messy and poorly written when I wrote it, and I want it to be neater for the Final Design Doc.

November 13 – November 17

This week, I have not been able to get as much done due to another major project in another class being due very soon that I have been putting more

focus into to complete, which my team is aware of. However, I still did what I could for Senior Design 1.

I set up one more meeting for the group with our sponsor this week since we are not meeting next week due to the Thanksgiving holiday. Since we are meeting with a professor on November 27th who has more knowledge on Virtual Reality and how what we want to implement might work or not work, our sponsor shared some links for us to look over to help us regarding accessibility features, which I have started looking over. The links he has sent us discuss more on accessibility for websites and apps, but I know this still applies to us for Virtual Reality since we will be using some common accessibility features for visually impaired people using websites and apps, along with the fact that Virtual Reality is essentially another screen one would be looking at and using.

November 20 – November 24

Over Thanksgiving break, I took the time to finish going over the links our sponsor had sent us and consider how it applies to our project. Since we are focusing on accessibility for the Blind and Visually Impaired community, I believe much of the info from what our sponsor sent is useful and important to keep in mind. While the links focused on website accessibility, there are still aspects that can apply to our project, such as making content easy to read for those who are visually impaired or even what we planned to focus on with creating a zoom feature and an alt text feature. I also went over this information to prepare for the meeting with Dr. McMahan; since he is helping us with our project, especially since he works with virtual reality, I wanted to

have the accessibility information in case that was something he wanted to discuss in terms of virtual reality.

November 27 – December 1

Now that we've had our meeting with Dr. McMahan, we gained a lot of useful information as well as more concerns we have not initially considered. As of now, the idea of implementing a zoom feature is being scrapped due to concerns and complications that Dr. McMahan had brought up, which means research I had done before no longer applies to our project. Since we are now focusing more on the alt text idea instead, roles have been reassigned for how we will work and implement our features. I am still on the auditory tool as of now, but in regards to alt text– which we've split into four basic parts– I decided that I will be responsible for making a system that will flag the VR developer and warn them if their game objects are poorly named or not descriptive enough for alt text. New research has been started on this, and while I know I probably will not get too much information before the design doc deadline for Gerber groups, this research will continue into the winter break.

February 5 – February 9

This week, I got some work done for our Autochecker, which will alert developers if they are lacking alt text or have insubstantial alt text for visible game objects. The outline for it is done at least, and I know how I expect it to run, it is just a matter of writing the actual code in C# for Unity. However,

it does have to work with the two scripts for adding alt text so that the developer doesn't get overwhelmed or alerted for things that don't need alerts, which Justin and I are making sure to keep updates on as he has been writing and working on the scripts to add alt text.

Progress is a little slow due to having to take care of my sibling who has gotten sick.

February 12 – February 16

This week, I've worked on most of the autochecker, which should be done soon hopefully, as I've completed most of what I've outlined for the script. However, my updates to it haven't been tested yet, which I need to do to make sure it works as intended. Depending on if it works as we expect it to, the autochecker should almost be finished.

I've also started preparing for my group's CDR that we will be presenting in less than two weeks, since I want to make sure that goes well.

February 19 – February 23

For this week, most of my efforts was spent helping to put the presentation for my group's CDR together as well as preparing to present so that I would know what to say, as originally our CDR was supposed to be this upcoming Wednesday and I wanted to be ready for when we presented.

However, it seems class for Wednesday is canceled, so we have more time than expected to prepare for our CDR.

February 26 – March 1

This week, I worked on fixing my code for our autochecker, as I found it is unable to actually run as intended. This is still being worked on to be fixed.

March 4 – March 8

For this week, I figured out what I will be saying for our CDR presentation now that my group finalized who is speaking during which slide as well as finishing our slides, since we are presenting on Monday. I also now have a solution for working on the project despite my broken computer screen, so I have been able to get back to fixing the autochecker like I originally wanted to work on last weekend.

March 11 – March 15

This week, now that our CDR presentation is done, I am now focusing completely on finishing the autochecker. With spring break next week, since I will not have other classes to worry about for a while, I expect to get a lot of progress done and even hopefully get it done and working as it should be.

March 18 – March 22

More progress was made on the autochecker, until I found I don't seem to be able to actually test the alt text editors on my computer, which gave me concern as to how to actually test the autochecker. Time was spent trying to figure that out, and I have found so far a way it can be tested, however it can't seem to be tested on my computer on our original project we've been using.

More work will be continued over the weekend before our group's Gerber demo.

March 25 – March 29

This week was spent finally getting rid of the remaining errors from the alt text checker before our Gerber demo, as well as finding out why exactly it did not seem like it could be tested on my computer. It was also spent sending out emails for my team for possible faculty we could ask for our presentation, which we are still trying to figure out, especially when we so far we have gotten one rejection and no other responses. Since we are having our Gerber demo rescheduled due to technical issues that we did not anticipate nor understand why it happened, I will continue to optimize the autochecker as well as clean it up so that there are not as many repeats it provides to the developer.

April 1 – April 5

This week was spent fine tuning the autochecker so that it is more polished and refined, finding more people for our faculty for the final presentation, and working with the team to fix most of our major errors before our rescheduled demo, especially after what had happened during our original demo. I was able to contact two people who agreed to be our faculty members, and I am not stressed with the autochecker since it functions as intended, as everything else with it is really minor.

April 8 – April 12

This week was just for cleaning up the code of the Autochecker, confirming with my group's committee how they plan to attend the presentation, and recording my part of the Senior Design video due this Saturday. I have also been mentally preparing for our final presentation that is happening on Monday morning.

John Boyd

October 16 – October 20

Last week I attempted to set up a Discord bot so that we could do our daily scrum meetings asynchronously. The bot was Orli PM, and while the features seemed useful, it did not matter because the bot has been offline

since adding it to the server. As a replacement, I added DailyBot, which has been serving our purposes this week.

Other than that, I have been researching testing techniques in Unity and the use of text-to-speech in VR contexts. It's important that we have logging capability when we do our IRB testing so that we have quantitative data that is easy to analyze. From testing, we can put in a text-to-speech function as our sponsor requested.

October 23 – October 27

I did research on the text-to-speech technology at the request of our sponsor. This research included the history of TTS and the current state of it. The main goal of it was to find current usage of TTS in virtual reality software. While TTS implementation is very important to the blind and visually impaired community, and is thus implemented widely across desktop and web technology, I found that it is rather scarcely used in the context of immersive experiences such as virtual reality and augmented reality. I communicated this to our sponsor on Thursday.

October 30 – November 3

This week was the beginning of our second sprint. I ran our second sprint setup meeting, organizing the tasks that we needed to complete before our TA check-ins next month.

Most of my time this week was spent trying to analyze the code that the previous research team had already written for the project. The project is split into two main folders: an application folder containing dll, c, and cpp files and a simulation folder containing the majority of the previous team's work as a Unity project. I know that the application folder was automatically generated, so I focused my effort on the simulation folder. The main feature that I was in charge of analyzing was the sonar feature, the functionality of which was contained in the RayCone folder.

Now that we have our hands on the Oculus I have also gotten (minimal) hands-on experience with the previously developed tools. The sonar and haptic tools are the main navigational features, with the audio system aiding them. What I've noticed is that there's never a moment that makes sense to not use the sonar system without the haptic system or vice versa.

November 6 – November 10

I'm learning how to implement heat maps into Unity code to see where users are getting stuck, and where users are using most tools. I'm doing research to see if it would be better to find a tool online to implement the heatmaps or if it would be a better idea to create a custom tool myself. The most relevant tool that I've found online is Unity Heatmap by kDanik on Github, as it allows for decent collection and visualization of data. The plan is to start implementing the heatmaps over the weekend or early next week.

November 13 – November 17

This week, I've been coding in the ability to log the usage of the existing features. I did this by adding a function that will log the usage of the sonar and haptic tools to a JSON file. This file can then be read from for analysis.

With data logging implemented and sampled data created, I then tried to create a custom tool to create a heatmap of the sonar and haptic tools' activations. However, when I tried to draw the heatmap gizmo the editor faced massive slowdowns. The solution that I've created uses too much space in memory, and changing how often it runs does not do much to help. I'll continue trying to optimize it, but I may seek another solution for data analysis next week.

November 20 – November 24

I tried to further develop the heatmap, but the amount of effort it was taking to attempt to optimize it was beginning to outweigh the benefits of having a working heatmap. It kept lagging the Unity editor too much, so I am going to analyze the log data manually. The program already has the ability to log data, so I'm going to take the data from testing and put it into a Google Sheets spreadsheet. From there I can create a scatterplot of the data using the x and z-coordinates of when the tools are used.

November 27 – December 1

A lot of the project that we're developing is going to be making a tool that developers will be able to use. So I've been researching the Unity Editor API, which we'll be using to write code for Unity Editor tools. Most Unity C# scripts inherit the MonoBehavior class, and are compiled and built to run during executing. Editor code, however, inherits from two different classes, Editor and EditorWindow, to create tools for use in the scene inspector or a separate editor window, respectively. I've been looking through the documentation of the Unity Editor class and how to make tools such as buttons, dropdowns, graphs, etc., that we can use to make our alt-text generation tool.

February 5th - February 9th

I began implementing text-to-speech to the project. We are using Wit.ai along with Meta's Voice SDK for Unity. The way we're using text-to-speech would, in it's full implementation require another part of the project that a groupmate is testing to be merged. In the meanwhile I am providing the text through a sample object and having it output. I set it up to use the partial vision (psuedo-alt-text) system we have right now. It currently works when interacting with the object. I'm trying to customize it so that the user has control over some elements, like pitch, speed, and cadence.

February 11th - February 16th

I am behind where I want to be, but base functionality is present for the text-to-speech feature. The feature triggers upon the activation of our partial vision tool and reads out a description of the object. I have begun looking

into how to implement unit testing so that we can get a continuous integration thing going.

February 19th - February 23rd

I've updated the CDR presentation with everything that I can up to this point. This includes the group difficulties, a list of soft deadlines, and a description of the work that was needed for the TTS tool. I have done a little bit of individual practice on delivering my slides.

I'm also writing tests and documentation for the TTS tool. I should be working on one of these at a time. I'm not.

February 25th - March 1st

This week has been more practice for the presentation since it got delayed. I have the delivery I'm satisfied with whilst practicing by myself and I'm generally not stuttering or using filler words as I speak. My group and I set a time to practice together on Monday.

March 4th - March 8th

I practiced the presentation that we will be giving to our peers with the group. Due to being unsatisfied with the flow of some slides, I made edits to them and my script. The development process was greatly slowed due to this and other responsibilities.

March 11th - March 15th

I've done some work to refactor the tts tool to work based off the AccessibilityTags script rather than off of the PartialVision script. The

refactoring has been simple so far, I'm hoping it stays that way so I can get on to testing.

March 18th - March 22nd

I made a mistake in refactoring the TTS to fit the AccessibilityTag thing, so I fixed that. Justin had trouble making the partial vision tool work in RocketMan, so I helped him figure out what was going wrong. The way that the menu "disappears" is by teleporting out of bounds and there was no means to do that until today.

March 25th - March 29th

This week I prepared for the demo by installing the apk on the headset with the original simulation. I still haven't worked out the package manifest issues to see why I can't get the dependencies to work. Hopefully, progress over the weekend.

April 1st - April 5th

This week I refactored the prefab to match Unity standards and then fixed the bugs that arose as a result of that. I'm still working on getting TTS to work in the package, but progress is very slow and the lack of docs from Meta aren't helping.

April 8th - April 12th

The TTS works. There was a Unity version error and an api request error and it took a lot of time but I got it to work.

Group 22: Accessible SDKs in Virtual Reality

Final Design Document

Justin Morera

October 16 – October 20

This week I contributed a bulleted list of metrics that would likely be useful for the testing phase of our project. This list was approved by our sponsor and my other team members. The next step of these ideas would be to implement them into the code to track these metrics. Additionally, I met personally with our sponsor and obtained the VR headset that was made available to us for use on this project. Finally, I continued working on documentation and expanding on my ideas for testing metrics in formal writing.

October 23 – October 27

This week, I did substantial research on text-to-speech for screen readers in virtual reality. I found multiple sources outlining the importance of such a feature as well as some information pertinent to the actual implementation of it. Additionally, I completed my portion of the 15-page Individual Design Document assignment, and met with my team and our sponsor to discuss current progress and future goals for the next sprint. I was able to accomplish a decent amount of research and also gain a better understanding of the challenges blind and visually impaired individuals face in digital environments (and everyday life), while also further appreciating the value of our work on this project. Finally, I continued to brainstorm and develop my ideas for furthering the project goals and improving its features.

October 30 – November 3

This week, our group collectively went over the GitHub repository from last year's team and began understanding the file organization and structuring of code for each of the project's features. We also charged and tested out the sponsor-provided Meta Quest 2 virtual reality headset and will begin testing the project tools and levels ourselves next week. I also started learning C++ so that I can begin coding the test metrics into the existing files. I also started searching for the files that belong to the haptics tool so that I can make sure I have identified all of them in order to insert the proper code into each one.

November 6 – November 10

This week, I learned C# and became familiar with the syntax and the structure of classes and interfaces. I used my newfound knowledge to come up with some major ideas for how we implement test metrics into the previous team's code to properly gain quantitative information for each tool during testing. Additionally, I installed Unity and am beginning to use that to understand the structure of the project's code better after realizing that the Git repository was cluttered with automatically-generated files and the scripts were not organized in a meaningful way; Unity organizes the scripts and attaches them to the features that they apply to, which makes locating the necessary files for each tool immensely easier. Finally, I continued

researching text-to-speech accessibility and also began writing out a detailing of our project's tech stack and concept of operations.

November 13 – November 17

This week, I wrote extensive documentation covering the project's Technology Stack and an overview of the Concept of Operations in preparation for our second check-in of the semester with the course's Teacher's Assistant, Lexy. I also continued practicing C# and familiarizing myself with Unity and its Graphic User Interface, Unity Hub, to improve my usefulness when I start coding. Additionally, I continued planning implementation ideas for how we can make the selected tools easily portable to other developers' virtual reality applications. And finally, after our meeting with our sponsor, I volunteered to make a user survey for our testers over this coming weekend so that when our sponsor starts the testing process, we will have a streamlined way to track the more opinionated and qualitative metrics we need. The first draft of the survey is already developed and awaiting approval from the team and our sponsor.

November 20 – November 24

This week, I finalized and sent out the survey for our testers and went over the results from the testing period that occurred over Thanksgiving break. I began turning some of the user feedback into quantitative metrics for statistical analysis of the responses to gain some insight into the average approval and feedback of each tool that was tested. I also continued to look

into text-to-speech and screen reader implementation for virtual reality and I practiced going over the C# syntax and better understanding the structure and implementation of interfaces and other object-oriented concepts in C# as I believe those will be paramount in our development later. Things have slowed down as we near the end of the semester and the break, but now that we have testing data to look over, we will be able to deliberate and better identify the route we will take when we reach the development phase of our project.

November 27 – December 1

This week, we got insight and direction from Dr. McMahan about possible routes to take for the development of our selected tools. Specifically, out of our two main tools, the magnifying glass/zooming feature and the text-to-speech/screen reader feature, it was advised that only one tool would be feasible to implement in our allotted time frame. This is due to concerns about the scope of the features and the direction of development required to produce the desired effect for each tool; the zoom feature would need to be implemented as its own entity in the environment and attached to an object, likely the user's controller, while the screen reader feature would need to be implemented as an application-programming interface tool that takes place primarily during development. We agreed that the screen reader was the most important tool and that we would focus on implementing an automated alt-text creation process but would also use my previous idea of an interface with alt-text and other fields as a manual alternative to fill any gaps left by the automatic process. I have volunteered

to flesh out the implementation of a manual in-development process for adding alt-text to objects, and I will also continue to examine the other tools to see if their implementation can be combined with this process by adding specific fields that interact with those tools.

Next, I analyzed the testing results from our four testers and made some basic charts to display overall opinions about the tools to help us quantify the tools' current performance. I believe these simple metrics can help direct us toward the most beneficial outcome of our development by identifying the most promising features for further development, even with such a small amount of testing data. Finally, I emailed our sponsor to return the Oculus/Meta Quest 2 virtual reality headset back to us before the break so that we can continue to use it for testing and increased familiarity.

December 4 – December 8

This week, I finished writing my 30 pages of the Final Design Document (this document). My portion included a Block Diagram containing the division of labor for the team, illustrating the tools each member would be working on initially; it also included a Use Case Diagram, showing a high-level and generalized workflow a typical user might take when using the application in their project’s development; and finally, it included an Activity Diagram which demonstrates the general structure of our design process.

Additionally, I collected each member's pages for the design document and organized and reformatted them to create a coherent and well-structured final product for submission. I deleted redundant headers and moved sections around so that everything flowed naturally and stitched together well. The finished product is satisfactory and showcases what we plan to work on; however, it is far from complete as our development is likely to change as we make progress. Some tools and features may prove easier or harder to implement in a transportable development kit style, but our minimum viable product is sound, regardless.

February 5 – February 9

This week, I improved the code for the alt-text tagging features, both automated and manual, to include descriptions of GameObjects as well as their names; I also installed an open-source virtual reality environment, called Rocket Man, that we will use to test our code in. I have begun looking into two things: how to determine if an object is interactable to add that information to the alt-text, and how to reinstall the PDK from the vanilla version downloaded from GitHub to the version with our custom scripts. Despite these milestones, progress has been mildly slow due to contracting the flu as well as a large assignment from another class; however, these developments are exciting and next week, I will likely have achieved a major breakthrough in progress.

February 12 – February 16

This week, I cloned the Rocket Man repo and added our code, as well as the previous team's partial vision tool code to it, revealing some dependency issues with the partial vision tool. This weekend, I am working on resolving that issue so that developers do not need to install third-party packages to use our SDK, and instead can use our code on its own to add functionality. I also created a new repo for our finished code to go, where I will develop a detailed README.md explaining to developers how to download and insert our project into their Unity environment, and any requirements it may have. All of our features will be there eventually, but currently it has the editor scripts that allow developers to add alt-text to objects, as well as the current version of the partial vision tool (dependency fix still pending) and the basic accessibility script utilized by the aforementioned editor scripts.

February 19 – February 23

This week, I mostly tailored my efforts toward finishing the Critical Design Review presentation. I worked on solidifying the details of the Partial Vision Tool, SDK portability, Manual Alt-Text Editor, and Automatic Alt-Text Editor slides, as well as the broader topics such as goals and requirements, so that our slides were concise and clear on what the project's focus is. Aside from presentation-related accomplishments, I also made a breakthrough in getting the Partial Vision Tool to work in the RocketMan environment; while I haven't gotten to add the tool to the scene, I did come to the realization that I had not transferred all the parts of the tool to the

new environment, rather I only transferred the script for the tool itself. Now that I have a better understanding of the supporting scripts associated with the Partial Vision Tool, also created by the original team, I believe I can deliver the package of scripts to RocketMan and allow it to actually function and accept input in the environment. Lastly, I sent screenshots of the working Editor scripts in Unity to our sponsor to give him a visual of what the finished feature looks like from a developer's point of view.

February 26 – March 1

This week, I worked on fixing the dependency issues with the partial vision tool and was able to remove an entire train of errors the environment, leading me to believe that the code should be able to be inserted to nearly any virtual reality Unity project without adjustment, so long as that project uses the most up-to-date version of Unity's InputActions module. If not, I have added a step to the README.md explaining that the code relies on the module so the InputActions module must be installed to the project if the project was previously using the outdated method of inputs. This can possibly cause a limitation to the use of some projects but will ultimately streamline the SDK's seamless insertion into other environments. I also further tested our features into RocketMan to ensure the newly added scripts behave accordingly, then shifted my focus to our Critical Design Review presentation as it is scheduled for the following week. I added my roles and accomplishments to the presentation as well as gave timeline estimates for the four aspects of the project I have contributed to.

March 4 – March 8

This week, I used the extra time granted to us by scheduling delays to expand and polish the Critical Design Review presentation slides. I was able to run through my parts multiple times and I also helped flesh out multiple slides to further expand on the material we will be discussing. Additionally, I had some ideas about how we can rework the AccessibilityTags implementation to have multiple buttons, one for each tag, that developers can use to add functionality for each specific tool into their environment; I also added a slide discussing these changes for the Critical Design Review. Along the lines of project development, I finally started working toward dissecting the newly discovered objects and scripts used by the previous team and am almost at the point of fully implementing the partial vision tool into the Rocketman environment completely.

March 11 – March 15

This week, we did our Critical Design Review which was a major milestone for our group. Just before the presentation, I came to a major discovery for how to implement our SDK and was able to add that update to our presentation. I found out that we can leverage Unity's Package Manager feature in order to make our SDK hosted in the Github repository even easier to use. I am still working on properly setting up the Unity package so that when a developer imports it from Github using the URL, it functions without error. There are a lot more steps with fine details that are not explained well for setting up a Unity package for distribution, but I am

confident that once I get past this barrier, implementation will be streamlined and headache free for developers.

March 18 – March 22

This past week, over Spring Break, I was able to fully integrate the Partial Vision Tool into RocketMan via the SDK. Barring the planned changes to the way the accessibility tags work and any tweaks to the documentation, the SDK has been successfully implemented and is ready for the other features to be inserted. I have now begun integrating John's work on the text-to-speech feature to add to the Partial Vision Tool's functionality in the SDK and will test that in RocketMan today. I am also working on recording a test in the environment through my headset to show our sponsor and to add to the video submission for the Senior Design Showcase.

March 25 – March 29

This week, was not a good week. I prepared our SDK for our demo by reproducing the steps one would take to turn a vanilla version of RocketMan into an accessible project using our package. I built and sideloaded the project onto my personal headset in order to ready it for the demo, and I tested it thoroughly, multiple times over the course of the week. I also worked with John to attempt to add the text-to-speech functionality to the partial vision tool before building it onto the headset but we encountered a roadblock in which the main Meta dependency of wit.ai, which actually performs the text-to-speech, could not be shipped in our package, but it

could also not recognize the package previously installed in the project; that feature is still in progress. I also worked Emmanuelle and Kasidy to get the Alt-Text Checker to work in the SDK and it was able to be included without any errors just before the demo time. However, during our demo, we were able to showcase all aspects of the project except for the pre-built Rocket Man instance because my headset started to malfunction at the most crucial moment; I did look into the issue after the demo and determined that it is a common error that Meta Quest 2 headsets experience that really doesn't have a reliable fix. It seems to be due to the sensor inside the headset that recognizes if the headset is being used or is idling (it basically looks at your face to tell if you are wearing the headset or if it's sitting on a table), then shuts off the screen if it is idle. Although we experienced a major setback by having to reschedule our demo for next week, we learned a valuable lesson in presenting a product: prebuild your showcase project on more than one headset, and also prebuild multiple projects, just in case one headset, or one APK malfunctions for any reason. Next time, we will have four avenues to take for demoing purposes.

April 1 – April 5

This week, I added polished up the alt text generators and the Partial Vision Tool; I added the ability for the alt text generators to check for all types of Colliders instead of just Mesh Colliders as it did originally, and I also added the ability for the generators to check for Rigidbodies on objects to mark them as interactible or not. These capabilities are connected to the Partial Vision Tool so that the new alt text and the Interactible tag are visible

when selecting all affected objects. Finally, I restructured the repo to fix some breaking errors caused by an attempt at adding the Voice SDK, and then we redid our demo and were able to show off the SDK in RocketMan without any hardware issues.

April 8 – April 12

This week, I worked on editing our CDR to be in the past-tense for our final presentation and also recorded my part of the Senior Design showcase video. The project is functional and nearly fully completed and I am working on preparing everything for our presentation on Monday.

Implementation

AccessibilityTags Script

This script was created to house the “altText” string and “interactable” boolean fields for the accessibility tools, as well as set the foundation for other accessible fields to be integrated with the “Add Accessible Fields” scripts easily. It is a simple script that simply holds the wrapper class with each relevant field for the accessibility tools. There is no underlying logic within the script.

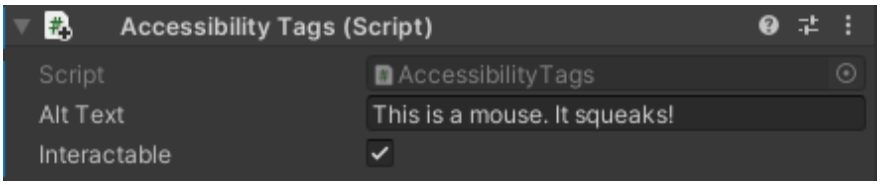


Figure 17: The Accessibility Tags script attached to a game object in Unity

Automatic Alt-Text Generator

This Editor script adds two buttons to the Unity Editor's Tools tab. When the first, "Add Accessible Field(s) to entire scene", is pressed, the script iterates through every GameObject in the scene and attaches the AccessibilityTags script to it. Then, if and only if the object has both a Renderer and an active Collider, it will generate alt-text for the object, using its name and any description field it may have on any of its other scripts. It also checks if the object has a Rigidbody script attached and will set "Interactable" to true if it does, false otherwise. The second button will cause the script to iterate through all GameObjects and remove the AccessibilityTags script if it's present. These actions can be undone by pressing Undo or CTRL+Z.

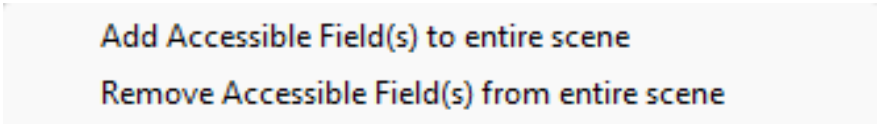


Figure 18: The two buttons added to the Unity Editor Tool Tab

Manual Alt-Text Generator

The Manual Alt-Text Editor is another Editor script that gives developers the ability to right-click on an object and add the AccessibilityTags script, as well as generate the altText as before. The main difference is that the selected object doesn't need to have a Renderer or Collider, which allows developers to interact with the altText property in different ways using their own code. It can also be used to simply reset the altText of an object for any reason. The script also checks for a Rigidbody and sets the "Interactable" field accordingly. It can be undone, as well.

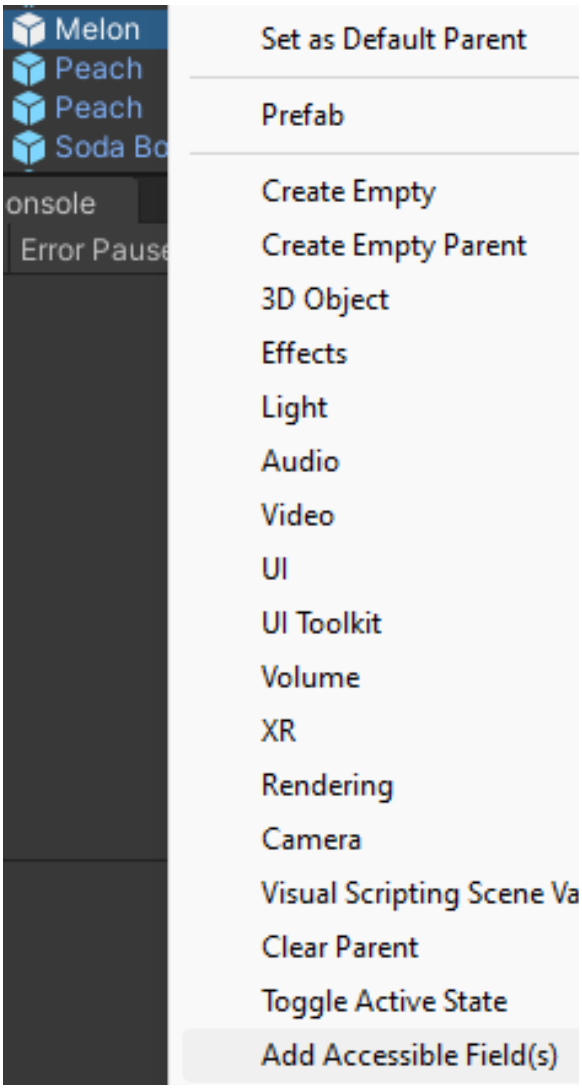


Figure 19: Add Accessible Field(s) button under right-click menu

Alt-Text Linter

The Alt-Text Linter, or the Automated Alt-Text Checker as what the team referred to it as (or "autochecker" informally), is a tool that checks if game objects with a collider and renderer have the Accessibility Tags script and alt text. This can be found under the Unity Tools tab and activated by clicking "Check Alt-Text." If there are objects that do not have the

Accessibility Tags script or alt text, this autochecker will alert the developer in the Unity Console as Log Warnings to let them know that certain objects are lacking them and that they need to be added. It also checks if the alt text is substantial enough— since the Automatic Alt-Text Generator adds the object's name into the alt text when used, the autochecker also checks to see if there is more to the alt text than what is initially added. If not, it will alert the developer, telling them to add more so that there is a description of the object for the user.

The autochecker also flags the developer if the names of different objects are not specific or unique, such as if they are only one word or are duplicate names of another object, including names with numbers such as Box1, Box2, Box3, and so on, since the names are already added to the alt text. The naming conventions are not necessary like alt text is, but we still alert the developer just in case it is not intended and to see if these objects need to be differentiated in some way or if there can be a more accurate description. These show up in the Unity Console as Log messages instead. It is recommended that the developer clears the log each time before using the autochecker to see a more updated and accurate count of their issues regarding the alt text of game objects.

Partial Vision Tool

The Partial Vision Tool was made portable for the Software Development Kit by transporting the script from the original simulation and inserting it into the SDK repo after removing the hardcoding. The script now

utilizes the fields in the AccessibilityTags' script to set the floating menu text and for the Voice SDK. The Input System implementation in the code was also updated to be more proper.

The prefabs from the original simulation were also transported with their preset settings into the SDK. The only prefab that needs to be used directly in a developer's scene is the Partial Vision Assistance prefab which requires the developer to select the Head object in their hierarchy for camera positioning for the floating menu, the Raycast Origin to determine where Raycasts are emitted from, TTSspeaker after setting up Wit.ai, as well as input buttons for tool activation, showing, and hiding the menu.

Figure 20: Partial Vision Tool floating menu

Wit.ai - Voice SDK

We chose Wit.ai as the service by which we would offer text-to-speech capabilities as it was free and integrated as part of Meta's VoiceSDK package. The package requires this SDK as a dependency, and the TTS systems will not function without it.

The TTS functions alongside the partial vision tool closely, as the tags that the tool reads are passed to the reader to be spoken aloud.

Unity Package Portability

The SDK is an organized Github repository containing all our finished scripts for the tools and tags, as well as the Editor scripts for devs to add into their environment. The key feature is a detailed README.md file that concisely instructs devs on where to put the files in their project, how to use them, and any requirements to fulfill. Developers are able to simply download the repository and insert the files into their project or install it directly through Unity Package Manager (UPM) using the Git URL. Setup is as simple as dragging-and-dropping the desired prefabs into place and following minimal configuration steps. Everything is detailed in the README and the Git URL is https://github.com/JustinMorera/VR-Accessbility-SDK.git.

Since the project is a Unity Package, every file within has a .meta file and every directory has a .asmdef file. These files contain the necessary metadata for allowing Unity to recognize the organization and script placement, as well as holding the references between any related scripts. These files are extremely important and any new files that are added by future groups need to contain their corresponding .meta files, as well as any updates to the .asmdef files they may need.

Acknowledgments

Joey Down is the sponsor for our project. He is the founder of Polaris Technology Inc. which is an immersive technology company whose mission is to develop inclusive technology that can help people live more independently. Joey is visually impaired so he has been very involved and helpful with the project's progression. He has had good suggestions for new features and how to improve what we implemented. He has always been available when we needed him, as well. Joey made sure that we would get our testing results. Even when the IRB testing was delayed and eventually fell through, he got some people he knew in the blind and visually impaired community to test our project and give helpful feedback.

Dr. Ryan McMahan is an associate professor at the University of Central Florida. He is also a researcher whose research interests are eXtended Reality (XR) & Virtual Reality (VR), training & cyberlearning, human computer interaction (HCI), and computer graphics. He worked with Joey and the previous team to give guidance, advice, and insight for virtual reality. Joey wanted us to meet with him as well. We met with him and ran our ideas by him so we could provide his input. We explained our magnifying glass feature and our text to speech feature. He gave us good advice on our next steps regarding these features as well as what may work and may not work with implementing them. With this meeting, we were able to get a clearer idea of what the features we are creating needed to look like.

References

Allan, J., Kirkpatrick, A., & Henry, S. L. (Eds.). (n.d.). *Accessibility Requirements for People with Low Vision*. W3. https://www.w3.org/TR/low-vision-needs/

The American Foundation for the Blind. (n.d.). *Digital Inclusion*. The American Foundation for the Blind. https://www.afb.org/talentlab/talent-lab-client-solutions/afb-accessibility-resources/digital-inclusion

The American Foundation for the Blind. (n.d.-b). *What is Braille?*. The American Foundation for the Blind. https://www.afb.org/blindness-and-low-vision/braille/what-braille

The American Foundation for the Blind. (n.d.). *"Watching" TV with a Visual Impairment*. The American Foundation for the Blind. https://www.afb.org/blindness-and-low-vision/using-technology/using-technology-entertainment-guide-people-visual-0

Blair, L., Lugo, J., Aronne, D., Le, M., & Quintana, H. (2023, Spring). Multi-Modal Orientation for BVI Individuals in VR Design Document. Orlando.

Blair, L., Lugo, J., Aronne, D., Le, M., & Quintana, H. (2023, Spring). M-MO-VR. Github. https://github.com/loganblair314/M-MO-VR/tree/main

Blindness (vision impairment): Types, causes and treatment. Cleveland Clinic. (n.d.). https://my.clevelandclinic.org/health/diseases/24446-blindness

Codemeariver. (2017, June 30). *Making a sniper scope in VR*. Code Me A River. https://codemeariver.wordpress.com/2017/06/30/making-a-sniper-scope-in-vr/

Georgetown University. (2019, February 13). *Visual impairments*. Health Policy Institute. https://hpi.georgetown.edu/visual/#

Lee, S. Y., & Mesfin, F. B. (2023, January 21). *Blindness*. National Center for Biotechnological Information. https://www.ncbi.nlm.nih.gov/books/NBK448182/

Marsden, J., Stevens, S., & Ebri, A. (2014). *How to measure distance visual acuity*. Community eye health. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4069781/

Oliver, D. (2021, November 22). *"I am not my blindness": What the blind community wishes you knew*. USA Today. https://www.usatoday.com/story/life/health-wellness/2021/11/17/what-blind-visually-impaired-people-wish-you-knew/8637271002/

Scott. (2022, June 10). *Website compliance for visually impaired*. ADA Site Compliance. https://adasitecompliance.com/website-compliance-visually-impaired/

The University of Pittsburgh. (n.d.). *What Is Vision Impairment?*. Department of Ophthalmology. http://ophthalmology.pitt.edu/vision-impairment/what-vision-impairment

World Health Organization. (2023, August 10). *Vision Impairment and blindness*. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

Zoom in on the iphone screen. Apple Support. (n.d.). https://support.apple.com/guide/iphone/zoom-in-iph3e2e367e/ios

Jesse Anderson. (2022). How Can a Blind Person Use Virtual Reality?. Equal Entry. https://equalentry.com/how-can-a-blind-person-use-virtual-reality/

Microsoft. (2021). Everything you need to know to write effective alt text. Microsoft Support. https://support.microsoft.com/en-us/office/everything-you-need-to-know-to-write-effective-alt-text-df98f884-ca3d-456c-807b-1a1fa82f5dc2

Rhea Althea Guntalilib. (2020). Screenreader Experience of a Virtual Reality Conference. Equal Entry.https://equalentry.com/screenreader-review-of-virtual-reality-conference-technology/

Sun, Y., Stellmacher, C., Kaltenhauser, A., Wagener, N., Neumann, D., & Schöning, J. (2023). Alt Text and Alt Sense in VR: Engaging Screen Reader Users within the Metaverse Through Multisenses.

Design and Deploy. “Unity 2018 - Creating Magnifying Lenses and Scopes.” *YouTube*, 16 Nov. 2019, youtu.be/TNbRjTaXH3I?si=9RElni81QcllHNSd. Accessed 18 Oct. 2023.

Garlic Suter. “Unity Create with VR Challenge 1.3: Magnifying Glass.” *YouTube*, 17 Aug. 2021, youtu.be/it5aZuiWCPA?si=iInM1k7bPBg_GqH7. Accessed 18 Oct. 2023.

Grady, Christine. “Institutional Review Boards: Purpose and Challenges.” *Chest*, U.S. National Library of Medicine, Nov. 2015, www.ncbi.nlm.nih.gov/pmc/articles/PMC4631034/ . Accessed 23 Oct. 2023.

“Lesson 3: What Are IRBs?” *HHS.Gov*, U.S Department of Health and Human Services, 28 June 2021, www.hhs.gov/ohrp/education-and-outreach/online-education/human-research-protection-training/lesson-3-what-are-irbs/index.html. Accessed 23 Oct. 2023.

“Magnifier: API Reference: Arcgis Maps SDK for JavaScript 4.28: Arcgis Developers.” *API Reference | ArcGIS Maps SDK for JavaScript 4.28 | ArcGIS Developers*, developers.arcgis.com/javascript/latest/api-reference/esri-views-Magnifier.html. Accessed 17 Oct. 2023.

Education, IBM Cloud. “SDK vs. API: What’s the Difference?” IBM Blog, IBM, 23 Aug. 2023, www.ibm.com/blog/sdk-vs-api/.

Hoffman, Chris. “What Is an API, and How Do Developers Use Them?” How-To-Geek, 27 Dec. 2021, www.howtogeek.com/343877/what-is-an-api/.

“Motion Sickness: Symptoms & Treatment.” Cleveland Clinic, my.clevelandclinic.org/health/diseases/12782-motion-sickness. Accessed 27 Nov. 2023.

Thompson, Sophie. “Motion Sickness in VR: Why It Happens and How to Minimize It.” VirtualSpeech, VirtualSpeech, 27 June 2023, virtualspeech.com/blog/motion-sickness-vr.

“What Is an Application Programming Interface (API)?” IBM, www.ibm.com/topics/api. Accessed 17 Nov. 2023.

“What Is an SDK?” Adjust, www.adjust.com/glossary/sdk/. Accessed 17 Nov. 2023.

Kurachkin, Daniil. (2023, July 10). *kDanik/heatmap-unity: Unity3D tool for recording and visualization of big datasets in form of heatmap*. GitHub. https://github.com/kDanik/heatmap-unity

Unity Technologies. (2023, December 1). *Manual: Custom Editor tools*. Unity - Manual. https://docs.unity3d.com/Manual/UsingCustomEditorTools.html

Koder, K. (2019, July 26). Making Your Own Unity Tools - Introduction to Custom Editor Windows - Unity Tutorial. YouTube. https://www.youtube.com/watch?v=34736DHWzaI