Team

Muskaan Narula (muskaannarula22@vt.edu)
Samridhi Roshan (samridhi18@vt.edu)
Bao-Tran Thai (tranthai@vt.edu)
Emily Newton (emilyrn18@vt.edu)

Code

https://github.com/SamyCoder/theArtMetaverse

The tools we used were primarily node.js, javascript, html, css, .x3d and .gltf models.

To run the code:
Go to the lib folder and type the following in the terminal: node mw_server.js
Once the node runs, a server URL will be generated.
Click on the link or copy and paste the link into a web browser and then the mirror world can be accessed.

The Art Gallery is the main world we have been editing in, some positioning that we altered for this world, may have the user spawn in an unideal location within the other worlds (aka the Moss Arts Center world or example world). So, if it looks like the world is a light gray with nothing else, try moving around, it might turn out you are somewhere inside the MAC or just looking at a wall.

Our Phase Reports:

Phase 1 Users, Tasks, and Data: Phase 1
Phase 2 Ideation: Phase 2
Phase 3 Abstract and Poster: AbstractFinalVTURCS
Poster: link

Map/Navigation Feature

This feature aims at providing users an overview of the gallery as well as critical “checkpoints” they can use teleportation to reach. This enhances the navigational abilities of users as they can identify key parts of the gallery and navigate to specific artworks much more easily. When one clicks on the make icon on the bottom left corner, one can see a birds eye view of the gallery appear. Along with pins marking the “checkpoints.” When the user clicks on a checkpoint, they will be teleported to the respective area.

What we have

The files we worked on for this feature include:

- world.html
- map.html
- map.css
- map.js
- checkered.x3d

world.html

In this file, the world is declared and loaded. Here, the button for the map is added as well as a modal for the map which pops up when the map button is clicked. This page also has “viewpoints” which users teleport to those respective points.

map.html

This file has the code for a basic modal and map setup. The code in this file is added to the world.html and this file can in the future be used as a template which is added into the world.html page.

map.css

This file includes stylings for the map.html page and the map components in world.html.

map.js

This file includes functionality for the map button as well as the checkpoints within the map.

checkered.x3d

This file is a “checkpoint” pin object which is used on the map. Users can click on this in the map to be teleported to their respective points.

What we need

While this is functional for the art gallery, we need a way to store viewpoint information in a file separate from the world.html. It would also be good to move the map modal into a proper component and then import it into the world.html. This clean up will improve the code quality significantly. A minor bug that would also need to be fixed is that when a user teleports, the avatar detaches from the user. Another addition to this feature would be the ability to view the checkpoint others are near. This can help users find their friends or others to discuss the art world with. This, in addition to the proximity chat feature would create more interaction. A unique feature that could be added is the ability for users to create their own bookmark or checkpoint in the world so that if they desire to return to that specific spot in the world they can.

Artwork and Audio Feature

The feature

One world we have focused on is where artists can display their artworks in the virtual world allowing them to also have an audio feature if they want a narration or storytelling to accompany their artworks. For each artwork, the user can have the ability to teleport to that artwork upon a single click and be able to pop up a mini screen that displays an audio menu and player as well as a miniature picture of the artwork. They also have the ability to close the artwork menu and keep the audio player open as they continue exploring the world or examining the artworks in depth.

What we have

The files we worked on for this feature are:

- world.html
- webclient.js
- artworld.x3d
- artworkAudio.html
- artworkAudio.css
- artworkAudio.js
- Mack

world.html

This is where the artwork panel is with the audio feature. We are able to get this div in webclient.js and dynamically change the artwork and the audio attached to it by clicking on the specific artworks. The primary gui features here is the image of the clicked artwork in the center and two arrows on either side that can direct the user to go to the next artwork in the gallery. There is also an audio button that can be pressed that will open up the audio controls and allow the user to listen to the narration of that specific artwork.

webclient.js

In webclient we worked on the function configureArtworkOverlay(). This function links the .x3d model of the artwork to the proper information that should be displayed in the artwork panel or overlay that is toggled. We do this by adding event listeners on each of the artworks within the gallery and setting the correct information within the artwork overlay or panel. This also sets up what artworks are going to be to the left or the right of this one, and makes sure that when the user presses the right arrow button, they are teleported to the correct artwork. The arrow key functions are changeToRightImage() and changeToLeftImage() and they are also within this file.

artworld.x3d

This is the art gallery that we are using and it is within the TheArtMetaverse/public/glTF_X3D_HTML_DOM/Sponza directory. We took it out of the AP_NP.html file within this same directory and modified it to include our own artworks. Using the .x3d file rather than just the .gltf file is extremely important because it allows us to include id's on .x3d models within the world that we can get by id in the javascript. This make it easier for us to interact with items in the world. The building itself is a .gltf file and is something we are not able to interact with, so this .x3d world we created allows people to add more items and interact with them.

artworkAudio.html

This file is a stand alone document that we used as a sandbox container to work on designs for the artwork overlay when the artworks are clicked on. Our hope was this could be inserted into the world.html file using javascript, but that became a bit more complicated than we first thought. Therefore we ended up copying and pasting all of the code that is in this file to the world.html upon completion of the design. We keep this in the code for testing purposes and also it is a great way to avoid merge conflicts when we are working on separate files.

artworkAudio.css

This file holds all of the styling for the artwork overlay and it also allows us to use the toggle functionality. This means that we are able to add and remove classes to the overlay div which allow it to appear and disappear just by clicking on an artwork or a button.

artworkAudio.js

This JavaScript file we defined the function configureArtworkOverlay() allowing us to add an event listener to toggle the artwork and audio panel upon clicking on an artwork. This function within this file is not currently being used in the project, we ended up adding it to the webClient.js file because we found out that we needed the world to be completely loaded before this function would be able to get any elements by id. Therefore we added it in webClient which make this equivalent run within its init() function onload of the body of the world.html

Mack

Mack is an avatar that we created and we are very proud of him. Please take care of him, he is sensitive.

What we need

Although the current version of displaying audio and artworks is what we had envisioned, some additional features that could be added would be to have are:

- Have a central bulletin board with a list of all the artworks
- Give the user the ability to upload mp4 files rather than just static artworks
- Be able to form smaller private groups to view the art gallery

Similarly to the bug in the Map/Navigation feature section, one bug that would need to be fixed is that when a user clicks on an artwork, they teleport to the artwork but they would detach from their avatar.

Proximity Chat Feature

The feature

The idea behind this feature is to be able to communicate with any user closer to us in the virtual space so that we can interact, appreciate, and view art together. The feature is willing to promote social interaction and networking in the metaverse.

What we have

At present, we have a private chat UI at the lower right corner of the screen. You can click on the “messenger looking button” at the lower right corner and it will open a pop up which contains a dropdown menu for selecting the user you want to chat privately with. The list of users are intended to be sorted based on the distance from the player user with the closest user being on the top of the list and the farthest user being at the bottom. The pop us can be closed by clicking on the button again clearly indicated by the x on the button itself.

The HTML code for this is present in world.html file and the CSS for the private chat UI is in chatProximity.css file (both in the public/mw folder).
As for the feature implementation for the private chat, a new socket has been created in the webClient.js file (in the public/mw folder) called privateSocket. The connections for io.emit and socket.on have been defined in the mw_server.js file in the lib folder.

[We would like to mention that the private chat feature is currently not working as intended. Due to time constraints, we had to go for the Wizard of Oz method. While the messages are being transmitted correctly, the private aspect of the feature is not working. The messages show up in every user connected. Also, the users are not currently sorted]

What we need

1. The private aspect of the chat feature needs to be correctly implemented. The issue we were facing was that the change in the dropdown was not being recorded and the variable we needed to implement that private aspect was reading null. (specifically userName variable in the socket.on(‘privateMessage’...) in mw_server.js)
2. The users need to be sorted by distance in the dropdown.
3. Other Potential Design Considerations:
    a. We can click the avatar and a pop up for chat requests can pop up and then after accepting it we can do the private chatting with the user we clicked on. However, it could be annoying if the user misclicks the avatar and the pop up shows up, so that could be something to consider against the current design.
    b. If the list of online users is a lot, then it could be difficult for the user to click on it. An option for this could be to only include the users in the online list in the same gallery room or a certain bounded distance.

Notes Feature

The feature

This feature can be found at the bottom right hand corner of the users screen. There is a green Notes button that a user can click and it opens up a window that has a text box the user can type in, a submit button, and a scrollable list of notes that they have saved. The idea behind this feature was to allow users to be able to take notes about artworks as they viewed them and listened to their narration. They should also be able to save them for later.

What we have

The files we worked on for this feature are:

- world.html
- webClient.js

world.html

In the world.html file it contains the code for the notes window. The button itself is placed in the footer of the page near the map and private chat button. When clicked it opens a window that holds a text area for the user to type in. There is an add button that the user can click which adds the button in the users local storage and displays it in a scrollable section of the window underneath the textarea.

webClient.js

In this file we worked in the setUpNotesApp() function in which it adds an event listener to the add button in the notes window. When a person decides to add a note, this function will add it to the user's local storage within the browser and then pull all existing notes within the local storage into the scrollable list within the window. This function is being run in the init() function onload of the body of world.html

What we need

The main thing we need for this feature is to allow users to actually save their notes. Currently it is being saved in the localstorage of the users browser, but it would be nice if the user can download these notes to save for later or save them to an attached database. One of the bugs is if the user has notes in their local storage already, they won’t show up in this window until a user adds a new note upon just opening a new browser. This is a problem that needs to be fixed as well.