Immersive Archeology

CS 4624

Multimedia, Hypertext, and Information Access

Report

December 14, 2022

Virginia Tech, Blacksburg, VA 24061

Instructor: Dr. Edward Fox

Client: Dr. Todd Ogle

Team: Sam Williams, Enmu Liu

List of Contents

List of Figures

List of Tables

1 Abstract

For this project, our job is to create a website that allows users to upload 3D scanning data from archaeological sites and artifacts found in a particular dig site for virtual analysis of corroborating evidence from the results of fieldwork. In archeology, the post-excavation analysis phase is typically the most time-consuming aspect of the archaeology process. The proposed Immersive Archaeology System would primarily contribute to this post-excavation phase, by creating an immersive environment for archeologists to connect. In the immersive environment, archeologists can gather potentially relevant data about a site and its artifacts from a library, for archeologists to analyze and interpret. This enables more flexible archaeological practices since both field-based and lab-based archeologists can corroborate via the immersive environment. Additionally, educators can make use of this system to teach learners about dig sites and their artifacts remotely. For this semester, our goal is to build the foundation of the project by creating a static, one-user, web environment. This semester, we created a website that allows users to upload artifacts and dig sites. With the artifacts and dig sites uploaded, the user can then view them in a 2D and XR mode. Future work can include functions such as multi-user interaction and discussion.

2 Introduction

2.1 Background

Archaeology is the study of the ancient and recent human past through material remains. Hence, the main job of an archeologist is to analyze these physical remains. When a dig site is found, archeologists generally have to be physically present to make an accurate analysis of the artifacts found. This is because archeologists not only need to see the artifact itself, but also its surroundings such as the sector or the depth where the artifact is found. (4) This project aims to combine 3D scanning and reconstruction, immersive virtual reality, and analytics in order to enable a new workflow for archaeological research. This enables archeologists to deduce the identity of an artifact other than just from photos. Our project would primarily contribute to this post-excavation phase by creating an immersive environment for archeology to corroborate and identify artifacts. In the immersive environment, archeologists around the world can connect together and draw together potentially relevant ethnohistoric data from archival sources to analyze and interpret artifacts without having to be physically present. It also creates an archive for 3D models, digital photos, and field notes, to be stored and labeled. Archaeologists performing analysis in the virtual environment can analyze an artifact together with its relative location in the virtual dig site. They can also have discussions with other archeologists. Moreover, they are able to use a time slider tool to view different phases of excavation and view the artifacts discovered in the same phase.

2.2 Objectives

For this semester, our project aims to build the prototype version of the immersive environment and achieve the following functionality:

- Create a working website that allows clients to upload artifacts and dig sites.
- Create a platform for users to comment on the artifact.
- Create a 2D display and 2D interaction with the dig sites and artifacts.
- Create a VR/XR interaction with the dig sites and their artifacts.
- Create a time-brushing function for the artifacts and dig sites.

2.3 Deliverable

A working website that enables users to:

- Upload artifact and dig site metadata.
- Dynamically view and interact with artifacts and dig sites in immersive environments.

2.4 Client

Our client, Dr. Todd Ogle, investigates strategies and methods for making the unseen seen via immersive experiences. Over the last decade, Dr. Ogle has used augmented and virtual reality to bring hidden histories to life, engage young learners in inquiry-based learning, and improve decision-making ability in spatially and contextually authentic immersive simulations. (4)

3 Design

We encapsulate our deliverable into a website that 1. stores a client's dig sites' and corresponding artifacts' metadata in a database that users can access via an upload form and 2. dynamically displays them in immersive and interactive 3D environments (Fig.1).

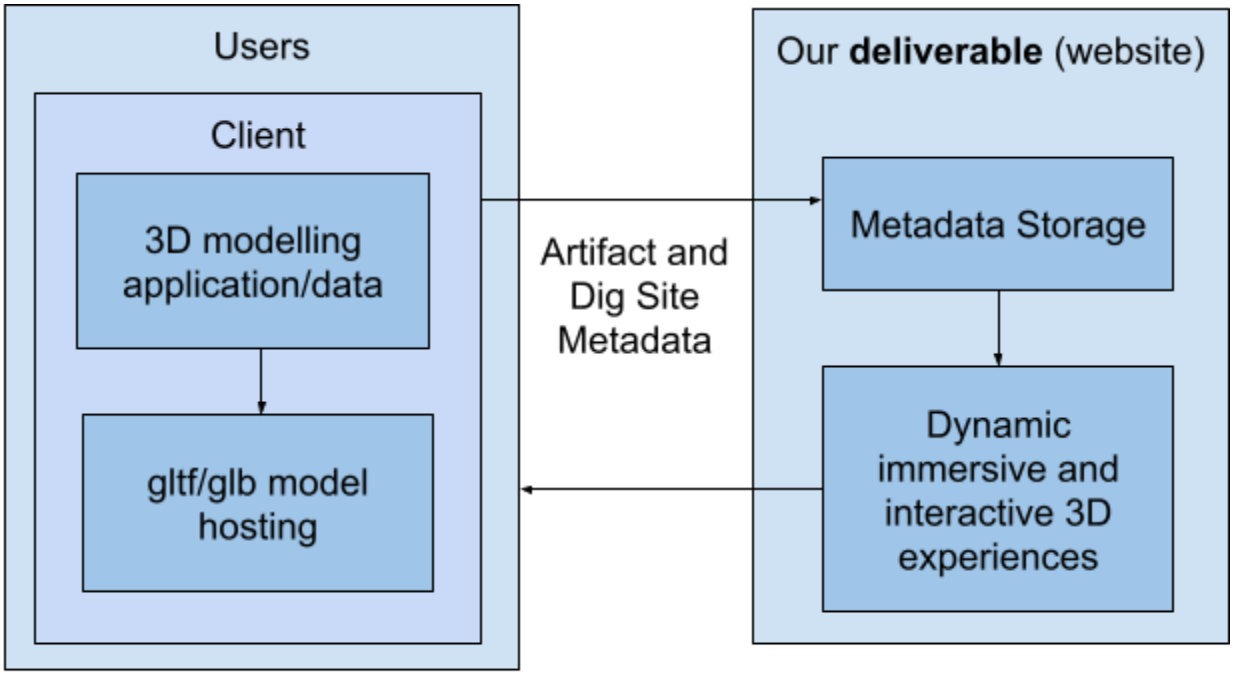

Figure 1: Our deliverable (website) abstraction

3.1 Clients

Not to be confused with our project client, Dr. Ogle; clients, as shown in Fig. 1, are a subset of users who have dig site and artifact metadata they would like to upload for other users and themselves to interact with. They satisfy the following criteria:

1. Clients are in possession of a 3D modeling program that allows them to export gltf/glb (9 & 10) models, ideally that retain coordinate data.
2. Clients publicly host the corresponding models.

3. Clients are aware of all artifact and dig site metadata that they wish to upload.

3.1.1 Criterion 1

Clients should be in possession of a 3D modeling program that allows them to export dig sites and artifacts as gltf/glb files, ideally that retain coordinate/location data. According to our client, Dr. Ogle, dig site model data is generally recorded via drones and LIDAR (11) and stored as point cloud data. The client's 3D modeling application should support loading this data and exporting it to gltf/glb files. If the location data of the dig sites is retained, artifact coordinate/position data can simply be copied into the artifact upload form (see Fig. 2). Otherwise, the client will need to do work to ensure artifacts will be correctly positioned in the immersive environment.

3.1.2 Criterion 2

Clients should publicly host the gltf/glb models. Our website does not support storing the gltf/glb files, only a link to wherever it is publicly stored. A free and easy way for a client to store a gltf/glb model is on Github or Zenodo. (16) The raw URL can be used as the model URL.

3.1.3 Criterion 3

Clients should be aware of the metadata of the artifacts and dig sites they wish to upload. The necessary data can be found in Tables 1 and 2, for artifacts and dig sites, respectively. According to our client, Dr. Ogle, clients will generally have artifact and dig site metadata stored in spreadsheet files that they can just copy over to the upload forms.

3.2 Users

As previously mentioned, clients are a subset of the user category, so they are included in this terminology. Users utilize our website to immersively experience and interact with the artifacts and dig sites uploaded by clients. Users can access these immersive environments via desktop web

browsers or the Oculus Quest (12). Mobile devices are not officially supported. Other XR devices besides the Oculus Quest are not officially supported but may function properly (i.e., devices that have 2 controllers).

3.3 Our Deliverable (Website)

Our website allows clients to upload artifacts and dig site metadata, and allows users to view and interact with the artifacts and dig sites based on a client's request for a particular dig site or artifact.

Users can perform any of three activities on our website:

1. Interact with individual artifacts, which have their own dynamically generated web pages.
2. Interact with individual dig sites, which also have their own dynamically generated web pages.
3. Interact with a particular dig site and all of its artifacts in an immersive environment. They can brush artifacts by particular excavation dates with a slider bar to display which artifacts are uncovered at that time. Additionally, this can be done in either a desktop web browser or an Oculus Quest.

All three activities will be generated dynamically, i.e., the individual web pages will not be stored statically, but instead queried dynamically from boilerplate pages with particular parameters specifying which artifacts, dig sites, and comments to fetch.

4 Implementation

4.1 Website Hosting

The hosting scheme we are aiming for (and currently using) is an AMP stack, composed of Apache, MySQL, and PHP. (1, 3 & 5) We are currently using 000webhost (https://immersive-archeology.000webhostapp.com/) (13), but would prefer to switch to a more secure and official hosting platform affiliated with Virginia Tech. We are testing locally with a program called WAMP, which provides a loop-back AMP stack for us to test with locally. (2) We are also using Github to synchronize and share our work.

4.2 Database

The database we are using is named “archeology,” and has 3 tables: dig_sites, artifacts, and comments. These tables hold the artifact, dig site, and comment data respectively, and are queried by the client (through the website) when appropriate.

4.2.1 Dig_sites

The dig_sites table (see Table 1) holds the dig site data. Each dig_site entry is comprised of the following fields:

Field Name	Description	Data Type
id	Dig site ID	PRIMARY_KEY int
title	Title of the dig site	VARCHAR 128
description	Dig site’s description	VARCHAR 256

model_url	URL to the dig site glb/gltf model	VARCHAR 256
date_begin	Beginning date of excavation	DATE
date_end	End date of excavation	DATE

Table 1: Database Table for Dig Sites

4.2.2 Artifacts

The artifacts table (see Table 2) holds the artifact data. Each artifact entry is comprised of the following fields:

Field Name	**Description**	**Data Type**
id	Artifact ID	PRIMARY_KEY int
site_id	Artifact's dig site's ID	int
title	Title of Artifact	VARCHAR 128
description	Artifact's description	VARCHAR 256
model_url	URL to the artifact glb/gltf model	VARCHAR 256
date_excavated	Date of excavation	DATE

location	Coordinates of the artifact	VARCHAR 64

Table 2: Database Table for Artifacts

4.2.3 Comments

The comments table (see Table 3) holds the comment data. The way that comments are queried is by selecting all comments from this table that correspond to a particular thread ID (thread_id) associated with a dig site or an artifact. They are then sorted by their timestamp to display in the comment section on dig site and artifact pages. Each comment entry is comprised of the following fields:

Field Name	**Description**	**Data Type**
id	Auto-incremented ID	int
thread_id	ID of thread with the comment	VARCHAR 64
name	Commenter's name	VARCHAR 64
email	Commenter's email	VARCHAR 64
comment	Comment itself	VARCHAR 512

date	Timestamp	DATETIME

Table 3: Database Table for comments

4.3 Artifacts/Dig Sites Upload

Currently, any user can upload artifacts and dig sites. In other words, users are equivalent to clients. Another limitation is that currently artifact and dig site metadata cannot be deleted or removed by clients. For future work, these may be areas of improvement.

4.3.1 Uploading Artifacts

Users can upload artifacts by visiting the upload artifacts page and filling out the provided form (Fig. 2). All provided fields correspond to the database's *artifacts* table (see Table 2), as listed above.

The model's URL should be a link to a publicly accessible glb/gltf file. The file is not uploaded to our site, instead it is downloaded from a host. An easy and free way to host a glb/gltf file is to upload it to a public GitHub repository and include the raw URL to the file, or via Zenodo.

The coordinates should be taken directly from the client's artifact spreadsheet or similar document (see Criterion 3 of the Design section).

Any error in input will be handled and reported in a red font underneath the submit button. If successful, the artifact will be viewable from the Browse All page and accessible in the corresponding dig site's immersive experience.

4.3.2 Uploading Dig Sites

Users can upload dig sites by visiting the upload dig site's page and filling out the provided form (Fig. 3). All provided fields correspond to the database's *dig_sites* table (see Table 1), as listed above.

The model's URL should be a link to a publicly accessible glb/gltf file, just like an artifact's.

The coordinate/location data of the dig site should be preserved while exporting the gltf/glb model (see Criterion 1 of the Design section). This is not a requirement, but otherwise more work will be required on the client's end to position artifacts correctly.

Any error in input will be handled and reported in a red font underneath the submit button. If successful, the dig site will be viewable from the "browse all" page and accessible in the dig site's immersive experience.

4.4 Comments

Users can post comments on both artifact and dig site web pages. The comments are dynamically downloaded from the *comments* table corresponding to either the artifact or dig site. This key is a string that is prefixed by either an 'a' or 'd' for artifact or dig site, respectively. The remainder of the key is the ID of the artifact or dig site. This avoids the issue of conflicting IDs for an artifact and a dig site. This key is labeled as the thread ID.

As shown in Figure 5, comments are displayed in increasing order by date (timestamp), where the most recent is shown first (an arbitrary design choice). The user can leave a comment by leaving their name, email, and a message less than 512 characters. If there is an error with the form data, an error will show underneath the submit button in a red font, otherwise the word "Success" will print in a green font, indicating that the comment was successfully uploaded, and the comment section will refresh.

If the user does not submit anything, the user will have to refresh the page to refresh the comment section. An issue with our comment section is that anyone can post anything and it can't be removed by clients or other users. This may be an idea for future work.

4.5 Immersive 3D Environments and Interaction

In our implementation of this project, we are using the THREE.js API (6) to create our immersive 3D environments and interactions. THREE.js is an open source 3D rendering library that provides an API for WebXR, which we allow to be run on the Oculus Quest.

Since we are implementing a time-slider to brush artifacts based on their expiration dates, we needed some form of UI control to allow interaction with 3D UI objects. This is not natively implemented in THREE.js. So we are using a 3D UI library from one of Dr. Ogle's previous projects to implement this functionality. (17) For example, this library allows users to hover on and click artifacts, as well as hover over and drag the time-slider in both the desktop web browser and Oculus Quest experiences.

4.6 Artifact Web Pages

Artifact web pages are dynamically generated from boilerplate PHP files for displaying artifacts in the web page */artifact.php*. Since this is a PHP page and we are running an AMP stack, if we add an ID query to the end of this URL, e.g., */artifact.php?id=123*, the web server (via PHP code) will return artifact data corresponding to the ID. If the ID is invalid, an error message will be displayed indicating that no artifact matching the queried ID can be found.

The artifact pages are only accessible via the "browse all" page or from inside the immersive environment, which we will discuss later. So a user will not usually have to deal with typing the *?id=123* portion themselves.

The artifact webpage has 4 main sections, in order from highest location on the page to lowest:

1. The artifact title and description
2. The interactive viewer
3. Other artifact metadata
4. The comment section

4.6.1 Section 1

The artifact title and description section simply contains the title and description of the artifact.

4.6.2 Section 2

The interactive viewer section is an instance of THREE.js that loads the artifact model from the response data queried by the artifact's ID. The user is provided with controls, allowing the user to rotate it and zoom in and out on it. See the User Manual section for more information.

4.6.3 Section 3

Other artifact metadata, such as the ID, site ID, and excavation date are shown here.

4.6.4 Section 4

The comment section is loaded here at the bottom of the page. See the Comment section for more information on comments.

4.7 Dig Site Web Pages

Dig site web pages are also dynamically generated from boilerplate PHP files for displaying artifacts in the web page */dig_site.php*. Similar to the artifact web page, if we add an ID query to the end of this URL, e.g., */dig_site.php?id=123*, the web server (via PHP code) will return dig site data corresponding to the ID. If the ID is invalid, an error message will be displayed indicating that no dig site matching the queried ID can be found.

The dig site pages are only accessible via the "browse all" page or from the home page in the bottom panel.

The dig site web page also has 4 main sections, in order from highest location on the page to lowest:

1. The dig site title and description

2. The interactive viewer
3. Other dig site metadata and the Visit in XR button
4. The comment section

4.7.1 Section 1

The dig site title and description section simply contain the title and description of the dig site.

4.7.2 Section 2

The interactive viewer section is an instance of THREE.js that loads the dig site model from the response data queried by the dig site's ID. The user is provided with controls, allowing the user to rotate it and zoom in and out on it, just like the controls for artifacts. See the User Manual section for more information.

4.7.3 Section 3

Other artifact metadata, such as the ID and the excavation begin and end dates, are shown here. Additionally, a "Visit in XR" button is placed at the end, which provides the user with the ability to access the immersive dig site environment, which we will discuss later.

4.7.4 Section 4

The comment section is loaded here at the bottom of the page. See the Comment section for more information on comments.

4.8 Immersive Environment

The immersive environment web pages are also dynamically generated from boilerplate PHP files for displaying artifacts in the web page */xr.php*, in the same manner as the artifact and dig site PHP files. If the ID is invalid, an error message will be displayed indicating that no dig site matching the queried ID can be found.

Regardless of whether the user wishes to use the Desktop Web Browser or Oculus Quest experience, they must first arrive at this page, generally from the corresponding dig site's web page's *Visit in XR* button.

The immersive environment web page will load a Three.js scene different from the artifact and dig site viewer, that allows the user to navigate and interact with artifacts and the time-slider to brush artifacts based on their excavation dates. It will display "loading" text while all the data is loaded (the artifact models, teardrop models to physically mark the artifact's locations, the dig site model, and a controls menu model and its components (including the slider). Once all is loaded, the text will be removed and the XR experience will begin.

The user, referenced as the player in the context of the immersive environment, will be able to navigate the environment and interact with artifacts (in the desktop viewer). A controls menu is also a child object of the player, which displays visual controls such as the time-slider, with which the player can interact. The player will also have a visual green guide-line, representing what they are pointing at. In the Desktop experience, it is controlled by the mouse position, and in the Oculus experience, it is controlled by the right-hand controller's position and orientation. For example, this is how the user will visually know that they are hovering over the slider or an artifact.

For information on how the controls change in the Desktop Web Browser vs. Oculus Quest experience, see the User Manual section.

4.8.1 Homepage

On the homepage, users can select a dig site from the bottom panel or click on the "Browse All" button to view all dig sites and artifacts.

4.8.2 Browse All Page

On the browse all page, users can see all uploaded dig sites and artifact metadata, and access the corresponding dig site and artifact web pages.

5 User’s Manual

5.1 Website Walkthrough

From the browse all page, users can observe properties of uploaded artifacts, such as their corresponding dig sites, date of excavation, and more information like their ID numbers.

In the web browser, the artifacts can be manipulated by clicking and dragging (rotating) and scrolling (zooming).

Users can also leave comments on each artifact's page, located at the bottom of the page. This is helpful for conversing with others to help understand each artifact better.

5.2 Dig sites

Similar to artifacts, the dig site pages contain much of the same information as artifacts, including the interactive viewer, metadata (though there are some slight differences, such as the excavation date range), and comment section.

But perhaps most notably, there is a "Visit in XR" button that will transport users to an immersive experience in the web browser (Figure 7).

5.3 Immersive Experience

The immersive experience can be enjoyed in both the desktop web browser and Oculus Quest. In the experience, users can navigate around the dig site and examine and interact with artifacts found there. Additionally, users can selectively filter artifacts by date if they wish.

5.3.1 Desktop Web Browser

After clicking on the "Visit in XR" button (Figure 7) on the corresponding dig site's page, users will be taken to the interactive viewer page. Once

the viewer is done loading, users can begin to interact with the environment.

5.3.2 Controls

- Movement
    - **W**: Forward
    - **S**: Backward
    - **A**: Left
    - **D**: Right
    - **Z**: Down
    - **X**: Up
- Rotation
  Click and drag with the left mouse button.
- Interaction
    - **Slider (Excavation Date Brush)**: Hover over and left click and drag the slider knob on the controls menu to brush (selectively filter) visible artifacts by their excavation dates. The left-most (default) setting is to view all artifacts, but moving the slider knob to the right will brush artifacts on particular dates in ascending order.
    - **Artifacts**: Click on the 3D artifacts to be taken to their artifact pages (in a new tab).

5.3.3 Oculus Quest

Users can access this website from the Oculus Quest's native web browser. After they arrive at the immersive experience page and it has loaded, a button will appear at the bottom of the page underneath the

viewer. In the Oculus Quest Web Browser (15), users can click this button to access the unique XR Oculus Quest experience.

5.3.4 Controls

- Movement
    - **Left Joystick (Forward)**: Forward
    - **Left Joystick (Backward)**: Backward
    - **Left Joystick (Left)**: Left
    - **Left Joystick (Right)**: Right
    - **Right Joystick (Forward)**: Up
    - **Right Joystick (Backward)**: Down
- Rotation**:** N/A
- Interaction
    - **Slider (Excavation Date Brush)**: Using your right hand, hover the right guide over the slider knob on the controls menu, press the select button on the right controller, and move it left and right to brush (selectively filter) visible artifacts by their excavation dates. The left-most (default) setting is to view all artifacts, but moving the slider knob to the right will brush artifacts on particular dates in ascending order (Figure 9).

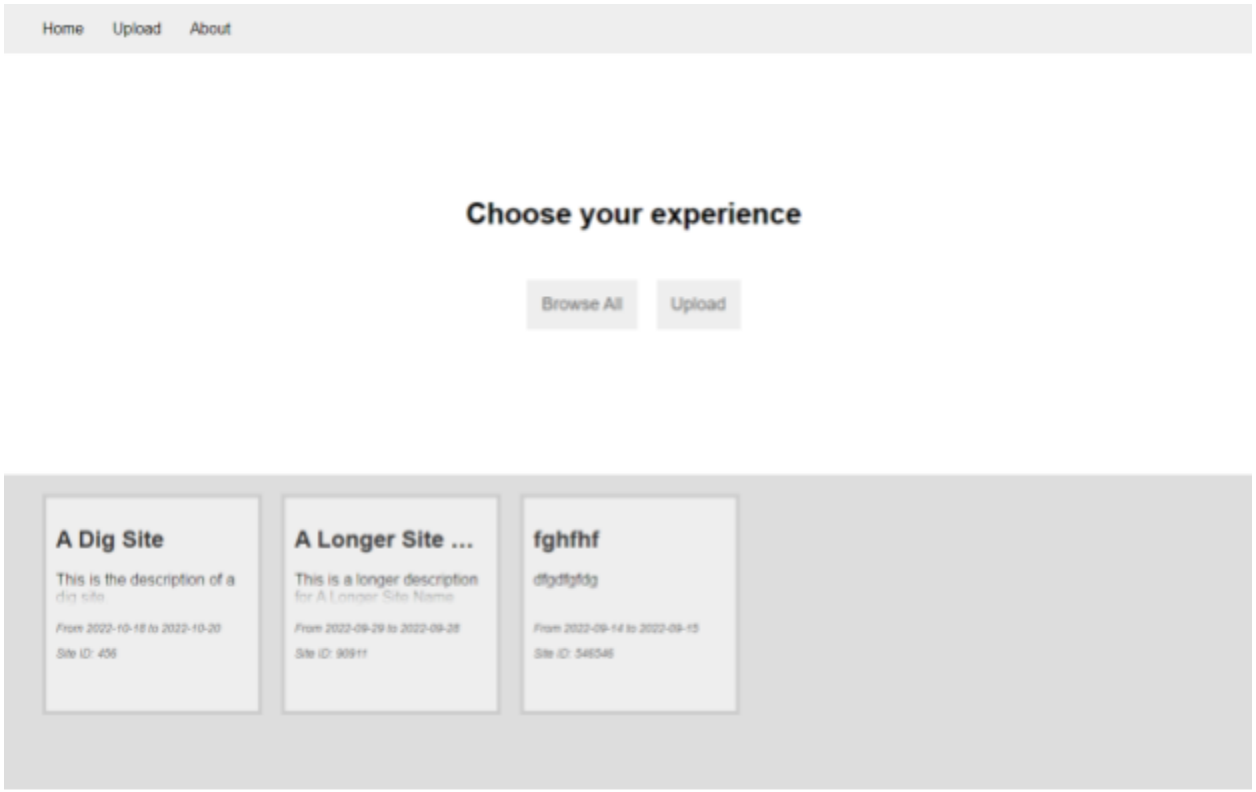

Figure 2: Homepage of Website.

Figure 3: Upload artifact form

Figure 4: Upload dig site form

Figure 5. Browse All page

Figure 6. Artifact View

Figure 7. Dig Site View

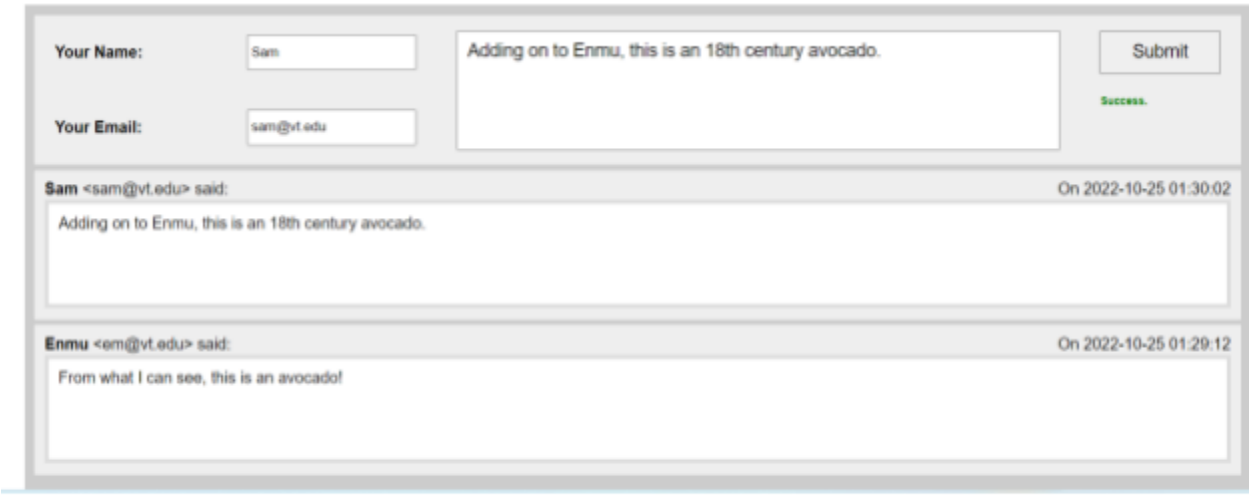

Figure 8: Comment Section

Figure 9: Website 2D Interaction

6 Developer's Manual

6.1 Overview

This project is built on an Apache/MySQL/PHP (AMP) stack. JavaScript (JS) helpers are frequently used for fetching and uploading information from PHP files in a more modularized approach than providing custom PHP/JS for each page. Additionally, the THREE.js library was used to create the XR experiences that supports both a desktop XR experience and Oculus Quest XR experience.

The project is stored on GitHub and the root directory includes a README and a *doc* and *www* directory. The *doc* directory contains documentation for the project, including a broad overview of the project, class information, database interaction, and how XR works in our project. The *www* directory contains the public files accessible on the website.

This project was developed with WAMPServer, an AMP stack that we could run locally on our machines. Publicly, the website is hosted on 000Webhost, a free, limited web hosting service. The URL of our site is http://immersive-archeology.000webhostapp.com/. A benefit of developing with WAMPServer is the use of *VirtualHosts*, that allowed us to simulate a website in particular directories, in both the *doc* and *www* directories. We encourage this approach for future development because it simplifies pathfinding. For example, an *include* statement can be based off of the server root folder defined in the *VirtualHost* folder (e.g., *www*).

6.2 File Organization

Here is a simple overview of the file organization of our project. The *www* directory contains the project code.

6.2.1 *www* Directories

The root directory contains *css, files, img, inc, includes, js,* and *upload* directories. The *css* directory contains multiple cascading style sheet (CSS) files used to style the website. The *files* directory currently contains a single GLB file (*map_pointer.glb*) used as the teardrop marker icon in the XR experience. The *img* directory contains images used in this project. The two currently used are *desktop_nav.png* and *xr_nav.png*, which are the control graphics shown in the respective XR experience (*xr_nav* referring to the Oculus Quest experience).

The *inc* folder contains PHP files used for database interaction, separate from the PHP pages users interact with. An advantage of this approach is both modularity and the ability to block public access to this directory. Further information on the different files and their interactions can be found in later sections.

The *includes* directory contains any files that should be dynamically visually included in other files. The only file we chose to use this for was *header.php*, the header shown across all pages on the website.

The *upload* directory contains the web pages for uploading artifact and dig site metadata.

6.2.2 *www* Files

The files in the root directory are the web pages available on the website. The user will land on *index.php*, the homepage, and see a header and 2 panes. The header is constant and contains some quick navigation links to the *Home*, *Upload* and *About* pages. The top pane prompts the user to either browse all dig sites and artifacts or to upload them. The former links to *www/upload/index.php* and the latter links to *www/about.php*. The bottom pane dynamically queries dig sites and displays them so that users can quickly access them.

The pages *www/artifact.php* and *www/dig_site.php* are the template web pages that query the artifact or digsite with the corresponding ID, respectively, as defined in sections 4.6 and 4.7. *www/xr.php* is the XR

experience page that loads the dig site corresponding to an ID and all of its artifacts, as defined in section 4.8. *www/browse.php* dynamically queries all artifacts and dig sites and displays them in a table. *www/about.php* contains information about the project and a basic user manual for users.

6.2.3 *www/inc*

This directory contains all of the PHP files used in this project. Most of the files provide PHP interfaces for either downloading or uploading (submitting) data.

Filename	**Description**
dbinfo.php	Contains the information needed to connect to the database. This varies between the locally hosted website and that hosted on 000Webhost. In the VTechWorks repository, sample database credentials are provided. They are not the same credentials as the ones that are live on the 000Webhost site, meaning the sample credentials mentioned in the previous sentence will not provide anybody access to the 000Webhost database the project is deployed on, even though the credentials are public on VTechWorks (as they are sample credentials). Note that this is not a secure method of storing database credentials and should be changed in continuation of the project.
download_artifacts.php	Provides a PHP interface for acquiring all artifacts.
download_comments.php	Provides a PHP interface for acquiring all comments, given a thread ID.
download_dig_sites.php	Provides a PHP interface for acquiring all

	dig sites.
submit_artifact_upload.php	Provides a PHP interface for submitting artifact metadata. Does some error checking but should be refined in the future.
submit_comment.php	Provides a PHP interface for submitting a comment. Does some error checking but should be refined in the future.
submit_dig_site_upload.php	Provides a PHP interface for submitting dig site metadata. Does some error checking but should be refined in the future.

Table 4. PHP files in *www/inc*

6.2.4 *www/js*

This directory contains all of the JavaScript files used in this project. This directory also contains a subdirectory, /*xrworld*, which will be examined in the next section. Each file can be examined more in-depth in the project upload space on VTechWorks(cite).

Filename	**Description**
commentwidget.js	Provides the implementation of a *CommentWidget* object, which is a widget for the comment section and functionality found in the artifact and dig site web pages.
createdomelement.js	Provides 2 helper methods for creating DOM elements. The first, *createDOMElemWithText*, allows users to create a DOM element with customly defined HTML. The second, *createDOMElem*, allows users to create a DOM element with its child

	being an existing DOM element.
submitget.js	Provides a JS interface for submitting a GET request by providing a URL.
submitpost.js	Provides a JS interface for submitting a POST request by providing a URL and an HTML form to parse.
tableparser.js	Provides a function to parse a JSON encoded MySQL array. This is used to parse data retrieved from the *www/inc/download_** PHP interfaces defined in the previous section. The result is a JavaScript array.

Table 5. JavaScript in *www/js*

6.2.5 *www/js/xrworld*

This directory contains files relevant to the XR portion of our project. The subdirectory *2DGUI* will be examined in the next subsection. The files in this directory are built on top of THREE.js. The file *xrworld.js* provides a template for every XR experience in this project – including the displays for the artifact and dig site static web pages that provide the interactive functionalities of zoom and rotation.

Filename	**Description**
artifactscene.js	An XR experience that extends the XR World template that downloads the artifact or dig site model from a provided ID, displays it, and allows users to interact with it as defined in sections 4.6.2 and 4.7.2. Not to be confused with *digsitescene.js*, this is used for the artifact web pages as well as the dig site web pages.
createdomelement.js	Provides an implementation of a controller (or player) for the XR experience in both

	the desktop and Oculus Quest experiences. See the User Manual for overall functionality. Also, see the comments in this JS source file provided on VTechWorks for more information.
digsitescene.js	An XR experience that extends the XR World template that downloads the dig site model from a provided ID, downloads all associated artifacts, places teardrop markers on them and makes them interactable, and sets up the player and controls. See the comments in this source JS file provided on VTechWorks for more information.
testworld.js	A dummy XR experience that extends the XR World template used for testing and example. Not navigable from the website besides manually loading the URL.
threehelpers.js	Provides some helper functions for THREE.js, including methods that normalize an object's scale, return a mesh's midpoints, among others used in other files.
xrworld.js	The template XR experience class. Runs on top of THREE.js by performing WebGL Canvas setup, interfaces for adding and removing objects from the scene, and providing an update function called every render frame with a delta-time for extended classes.

Table 6. JavaScript in *www/js/xrworld*

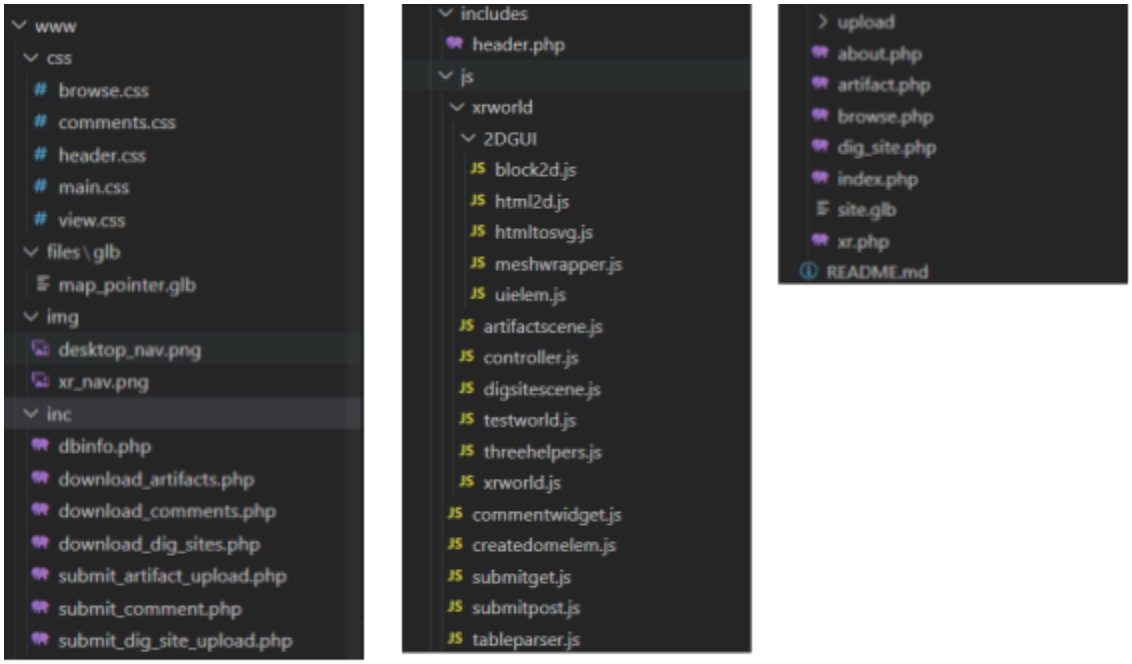

Figure 10. www File Overview

6.2.6 *www/js/xrworld/2DGUI*

This directory contains an interactive 2D GUI library developed in a previous project of Dr. Ogle's. It is built on top of THREE.js because THREE.js does not provide a built-in library for interaction. This library is not very encapsulated and requires contexts (like any derivative of *XRWorld*) to call events like hover and click. A big advantage of this library is its ability to create 3D images from HTML. Since we did not implement this library, only a basic overview is included for each file.

Filename	**Description**
block2d.js	Creates a block UI Element from either a solid color or a texture (image).
html2d.js	Creates a UI Element from a DOM element using *htmltosvg.js*

htmltosvg.js	Converts a DOM element to an SVG image.
meshwrapper.js	Provides a wrapper for any 3D Object by creating an interactable bounding box that has events like hover and clicking.
uielem.js	The base class for UI Elements, implementing an event system.

Table 7. PHP JavaScript in *www/js/xrworld/2DGUI*

6.3 Data Interaction

In this project, we provide data access via connected JavaScript and PHP interfaces.

6.3.1 Database Connection

The database connection occurs with all files in the *www/inc/* directory by using the information provided in *db_info.php*. If the configuration is correct, i.e., the server name, username, password, and database name files all check out, the PHP files will be able to successfully connect to the database. To access specific tables, PHP files manually specify the table name.

6.3.2 User Data to the Database

Generally speaking, the data flow from the user starts from some user data input to an HTML file. Then the data is sent to a JS API call (*www/js/submit_post.js)* that sends a POST request to a PHP submission file determined by an input link (*www/inc/submit_*)*. This PHP submission file processes the data, does some error checking, and then uses MySQLI to connect to the database and perform an INSERT query to a specified table.

6.3.3 Database Data to the User

One method of data flow from the database starts from a web page's JS API call to *www/js/submit_get.js*, which requests data from a provided URL. This function sends a GET request to a PHP download file (*www/inc/download_*)*. This PHP download file processes the query parameters (i.e., the key-value pairs after the *?*), does some error checking, and then uses MySQLI to connect to the database and perform a SELECT query to a specified table. The resulting rows are *echo'*d as an encoded JSON array string, which is returned back to the caller. This approach is used by *www/browse.php*, for example.

Another method of data flow from the database starts from a web page's PHP file, which skips the first step of the previous approach by immediately querying the data before the page loads. The resulting encoded JSON array string is then manually parsed by the web page itself. This approach is used in *www/artifact.php*, for example.

6.4 XR

Without going into too much detail, the THREE.js API is used as the WebXR library used in this project. We recommend future developers spend some time researching THREE.js before beginning work on this project, as well as spending time reading existing code and comments to get a feel for what is happening.

Once developers are comfortable with the basic THREE.js API entities and functions used in this project, they can begin to understand the connection between the desktop and Oculus Quest experiences.

In actuality, there is very little difference code-wise between the two, although the user interaction and immersiveness is obviously quite different. Both of these experiences are still running on the underlying JavaScript API and THREE.js (which itself runs on top of WebXR and WebGL).

The name of the game when it comes to developing for dynamic experiences such as those in this project are callbacks. THREE.js provides

a *WebXRManager* (accessible as a property of the *renderer* element) that allows developers to define callbacks for when a user switches contexts from desktop to XR. In *www/js/xrworld/controller.js*, the *Controller* class adds these callbacks and defines a flag *IN_XR* for later code to determine if the user is in XR or not. Callbacks are also used to determine any updates to controllers. For example, a controller battery dying will trigger an event for developers to handle. This approach is also how users can identify the number of connected controllers and have access to each's gamepads (joysticks, buttons).

The way that a user enters and exits the WebXR environment to and from the desktop experience is determined by the XR device. For example, in the Oculus Quest web browser, the default experience when accessing *www/xr.php?id=XXXX* is the desktop one. However, THREE.js provides a button implementation (*VRButton*) that upon clicking will signal the XR device to enter the WebXR experience. The devices themselves provide interfaces for the user to signal to their web browsers to exit XR experiences. Ultimately, these are signaled to the web browser via the previously mentioned callbacks that bubble to THREE.js's *WebXRManager*.

7 Lessons Learned

7.1 Timeline

When devising our timeline, we considered we would hit some roadblocks, so we added 2 weeks of catch-up time – one in the middle of our timeline and one at the end. This proved to be useful because by the time we approached both of the weeks we were approximately a week behind schedule.

Beyond this, we were fairly on schedule with objectives defined in the timeline. As explained further in the next section, a general challenge regarding our timeline was testing and debugging features on the Oculus Quest experience, which we could primarily only do in Dr. Ogle's office during our weekly meetings.

The general organization of our timeline was effective in that the first quarter focused on project setup, including the basic website format, which was important to get out of the way for subsequent parts of the project. The second quarter focused on building the foundation of our project in the basic implementations. The third quarter focused on refining these implementations and conforming them to Dr. Ogle's standards, and the fourth was finishing touches and working on the user manual and documentation. We wish that we actually planned more time for creating documentation, as this proved to be a very time consuming task.

We were not able to accomplish creating a search system for the artifact and dig site metadata, as the XR portion of the work ended up taking more time than expected. This was not a major issue for our client, Dr. Ogle. We also were not able to refine our website style to the extent we wanted, though it is functional. We also would have liked to do a little user testing, but we did not get around to it. Another important objective we did not achieve was acquiring a VT web hosting solution and domain. According to our client, he was not able to secure a hosting solution in time.

Overall, we are happy with our timeline and the progress we achieved with it.

7.2 Problems and Solutions

One of the most difficult challenges we faced was the workflow of developing and testing the Oculus Quest experience. One way we were able to test some functionality, specifically simulating entering and exiting the XR environment and moving fake controllers, was with the WebXR API emulator chrome extension (15). However, this did not allow us to simulate button and joystick pressing, which were fundamental controls in our Oculus Quest XR experience. The only time we could test this functionality was in Dr. Ogle's office by using his own Oculus Quest.

Another difficult broad challenge we encountered was developing a controller that adapted between desktop XR and Oculus XR experiences. This was a design choice used because of the similarity of code and functionality of code between the experiences, albeit with different modes of input. We were able to take advantage of this by creating modular functions, such as *calculateInputVector*, that would acquire user directional input as a 3D vector from either the keyboard or Oculus Quest controller depending on the experience. This result can then be used in subsequent calculations without any other code modifications.

However, there were some limitations to this approach, specifically from the aspect of user interaction. The way that a user interacts with artifacts in the XR experience on the web browser is based off of a raycast from the mouse position on the screen, while in the Oculus Quest experience it is based off of the right controller's position and orientation. There appears to be an issue with the THREE.js API regarding a method used to retrieve the world matrix of the right controller, which contains its orientation and position. The method, *matrixWorld*, returns an identity matrix instead of what it should. Additionally, the *matrix* method, which returns the local matrix, is working properly. Thus, it appears there is a bug somewhere in the API code. Nonetheless, we performed the calculations necessary to find the world matrix manually and fixed this problem.

8 Future Work

8.1 Project Setup

Teams can set up this project by pulling the repository from the Github link provided in the VTechWorks library. The information provided below has only been tested on Windows. Teams will need to download and install WAMPServer. Before the project will run on their local machines, teams will need to start the WAMPServer executable in the WAMP bin directory. Starting this executable will start processes for Apache, MySQL and PHP.

To set up the database, users should navigate to this link in their web browsers: *localhost/phpmyadmin*. This will take them to a login page where they should simply hit the *Go* button. After this, they should navigate to the *User Accounts* tab in the top navigation panel. Here, they should add a user account with the sample credentials provided in the *www/inc/db_info.php* file, specifically *dbUsername* and *dbPassword*. These should correspond with the *Username* and *Password* fields in the form to add a user. Then, teams should hit the *Go* button at the bottom of the page to create the account.

Next, teams should create the archeology database by clicking on *New* in the leftmost panel. The name of the database should be *archeology*, which will be created when the user hits the *Go* button. Users can then access the database in the same leftmost panel. After clicking on *archeology* in the leftmost panel, teams can create tables by entering the name of a table and hitting the *Go* button. Specific information on the tables to create are provided in this report and the README.md file in the Github repository.

Teams should also set up *Virtual Hosts* with WAMPServer by opening the system tray, left clicking on the WAMPServer icon, hovering over the *Your VirtualHosts* panel, and then clicking *VirtualHost Management*. A web page will open that will prompt teams for a name for the Virtual Host (we used *archeology*) and the absolute path to the directory (the *www* directory in the Github repository). Once teams click on the *Start Creation of the Virtual Hosts* button, they will be able to type in *archeology* in their web browser

and the website should be up and running, though they should restart WAMPServer before attempting to do so.

8.2 Future Work Ideas

Teams in a subsequent semester can work on a few ideas we did not focus on or achieve in our project this semester.

Teams can first implement the functionalities that we did not achieve, as described in the previous section. They can also work further on artifact interaction in VR. For example, they could implement functionality that allows the user to pick up and rotate the artifacts with controllers in the Oculus Quest XR experience. They could also allow clients to upload high-resolution digital photos of the artifact. With that, a neat functionality would allow users to see all artifact information in the XR experience without leaving it to the static versions, which would be especially ideal for the Oculus Quest experience.

Another big area for improvement is to implement user-to-user interaction in the XR experience. We envision future groups expanding on this work to create a collaborative environment for multiple users to interact in dig sites. Some specific areas to work on to achieve this are the development of avatars, a voice-over-IP-based chat system, and a pinpoint system that allows users to communicate more visually.

Another area for future work is site security. This site will need to be protected from hackers and robots. It is important to prevent them from adding bad data and viruses to the site. This may be an issue with our site because the database login information is located in a public PHP file. This should be done differently in the future. Additionally, teams should implement more client-side and server-side error checking, for instance, in both the JS and PHP form submission files.

Team in subsequent semesters should acquire a VT web hosting solution and domain.

9 Acknowledgments

We would like to thank Dr. Todd Ogle for proposing this project and providing guidance along the way. We'd also like to thank Dr. Fox for his advice, feedback, and guidance throughout this project. Their contact information can be found below:

- Client: Dr. Todd Ogle

  jogle@vt.edu

- Instructor: Dr. Edward A. Fox

  fox@vt.edu

10 References

1. Apache Software Foundation. (November 15, 2022). *Welcome to The Apache Software Foundation!* Retrieved November 15, 2022, from https://www.apache.org/

2. WampServer. (December 4, 2022). *WampServer, La Plate-forme de développement web sous windows - apache, MySQL, PHP*. Retrieved December 4, 2022, from https://www.wampserver.com/en/

3. MySQL. (November 15, 2022). *MySQL*. Retrieved November 15, 2022, from https://www.mysql.com/

4. Ogle, T., & Skarbez, R. (November 15, 2022). *Immersive Archaeology.* Retrieved November 15, 2022, from http://hdl.handle.net/10919/93734

5. PHP Network. (November 15, 2022). *PHP: Hypertext Preprocessor.* PHP: Hypertext Preprocessor. Retrieved November 15, 2022, from https://www.php.net/

6. Three.js Community. (November 15, 2022). *Three.js – JavaScript 3D Library.* Retrieved November 15, 2022, from http://threejs.org

7. Wikimedia Foundation. (2022, December 2). *Virtual reality*. Wikipedia. Retrieved December 4, 2022, from https://en.wikipedia.org/wiki/Virtual_reality

8. Wikimedia Foundation. (2022, November 19). *Extended reality*. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Extended_reality#:~:text=Extended%20reality%20is%20a%20catch,to%20interact%20with%20each%20other.

9. The Khronos Group. (2020, December 3). *GITF - runtime 3D Asset delivery*. The Khronos Group. Retrieved December 4, 2022, from https://www.khronos.org/gltf/

10. Iqbal, K. (2020, September 1). *GLB*. Retrieved December 4, 2022, from https://docs.fileformat.com/3d/glb/#:~:text=GLB%20is%20the%20binary%20file,images)%20in%20a%20binary%20blob.

11. US Department of Commerce, N. O. and A. A. (October 1, 2012). *What is Lidar*. NOAA's National Ocean Service. Retrieved December 4, 2022, from https://oceanservice.noaa.gov/facts/lidar.html

12. Meta. (December 4, 2022). *Oculus quest store: VR Games, Apps, & More*. Oculus. Retrieved December 4, 2022, from https://www.oculus.com/experiences/quest/

13. 000Webhost. (December 4, 2022). *Free web hosting - host a website for free with Cpanel, PHP*. Retrieved December 4, 2022, from https://www.000webhost.com/

14. Meta. (December 4, 2022). *Meta Quest browser on Oculus quest*. Meta Quest VR Headsets, Accessories & Equipment. Retrieved December 4, 2022, from https://www.oculus.com/experiences/quest/1916519981771802/

15. Google. (December 4, 2022). *WebXR API emulator.* Google. Retrieved December 4, 2022, from https://chrome.google.com/webstore/detail/webxr-api-emulator/mjddjgeghkdijejnciaefnkjmkafnnje?hl=en

16. Zenodo. (December 4, 2022). *Research shared.* Zenodo. Retrieved December 4, 2022, from https://zenodo.org/

17. Todd Ogle. (2021, May 4). *Veterinary anatomy VR - equine model process.* YouTube. Retrieved December 4, 2022, from https://www.youtube.com/watch?v=6HpZyfvAQ_s&ab_channel=ToddOgle