Maya Archaeology XR (VR) - Final Report -

Archaeological Research meets Modern Technology

1. Abstract

The world is continuously changing and evolving, reshaped by man and nature alike, with many parts of our history constantly being lost. Unfortunately, while we cannot recover lost information, we can work to preserve what is left. In this project, we introduce a new era of technology to archaeological research by digitally scanning and recreating excavation sites using LiDAR (Light Detection and Ranging), digital photography, and eXtended Reality (XR). We can store high resolution and accurate representations of physical locations to be loaded in Virtual Reality (VR) devices. This enables users to visit archaeological sites in a virtual environment. Our project demonstrates that with equipment available today, we can build and experience Maya excavation sites in a VR application.

Project Website: https://sites.google.com/ucsd.edu/xrchaeology/home

2. Introduction

Motivation

The human race has been around for hundreds of thousands of years, with the earliest fossil evidence found dating back to around 300,000 years ago in Africa [1]. While recorded history only spawns roughly the most recent 5,00 years [2], civilization has existed for much longer than that and we are always discovering new things about our past that have been forgotten to time.

Unfortunately, the world is changing more today than it has ever been, both by man and nature alike. This means that the lost parts of our culture as a whole are being destroyed, leaving more questions than answers in their wake. Problematic as this is for researchers, consequences may stretch further as these ancient records may hold answers that we might find useful even today.

Our goal is to use technology to digitally preserve these excavation sites and the relics they contain using modern techniques utilizing LiDAR (Light Detection and Ranging), digital photography, and eXtended Reality (XR). This will enable anyone from researchers to enthusiasts to visit and study excavation sites from the comfort of their own homes.

History and Previous Work

Traditionally, documenting archaeological findings has consisted of drawing and writing down observations. Excavations date back to as early as the 1600s [3] and was significantly improved with the invention of photography. Photography enabled the capture of scenes on film, eliminating the reliance on detailed illustrations and is still used today in the form of digital ground and aerial photography [4].

Unsatisfied with the limited resolution and accuracy of existing methods, newer technologies have been used to develop more advanced methods such as the use of Photogrammetry and LiDAR. This has enabled archaeologists to measure and create 3D models of artifacts and model dig sites, terrain, and structures all while not disturbing the area physically (minus photon and air exposure) [5].

Our Contribution

While there has been no doubt been great advances in the field over just the past few years, a few key problems still stand out that we aim to address:

- Ease of accessibility
    - While photos are easy to view, 3D models and other data often requires specialized software to even open
    - The data is often also hard to access for an average person, usually directed at the researchers working on a site
- Scale of realism
    - Looking at a photo or model is one thing, but being there in person is another
    - Viewing the photo and model on a screen is harder to gauge the scale of the area and the connection between objects
- Disconnect of locations
    - Being in the age of technology, we are able to browse all of humanity's information with the click of a button, this should be on the same level
    - Being able quickly to "travel" to different sites may lead to new discoveries

By utilizing eXtended Reality (XR), we aim to use Virtual Reality (VR) to create an application that can solve these issues. VR in its current state uses a head mounted display (HMD) and handheld controllers to override the senses. This enables the user to be immersed in a virtual world that is created by the application. This technology is actually more accessible than its name would lead on, for example, many modern smartphones support the functionality to act as a VR device. Therefore, we see the accessibility being far greater than those with the required software and knowledge of traditional methods required.

In this virtual world, the user would be placed in accurate recreations of the digitally scanned excavation sites. This enables the user to have a sense of scale and immersion of traveling to the site in person rather than looking through a window at snapshots of someone else's perspective. Furthermore, this enables the user to travel to multiple locations quickly and also enable the uninitiated to be educated about the area.

3. Technical Material

Software

The main piece of software used in this project was Unity Engine version 2019.3.11f1. This is used to build a cross platform application that can run on a variety of hardware combinations and can also easily be ported and modified. Unity Engine uses a combination of a high level editor to create virtual scenes and assets combined with C# scripts to handle the core functionality logic. The XR Interaction Toolkit, developed first party by Unity and with hardware vendor support, is used to handle the interactions from VR hardware and the standard graphics application.

Using Unity allows the project to be easily expanded and changed without much work needing to be done to the underlying framework. Also by not using external packages and relying on third party developers, we have full control over the lifecycle of our project and are only dependent on the first party support provided by Unity. This ensures that the project can be maintained regardless of third party packages and new hardware / features being added.

Software such as Photoshop and Audacity were used to process images and audio files used in the project as well. Assets such as menu textures and other minor details were created with it.

Hardware

The application was developed on a PC running Windows 10 version 1909 with an AMD Ryzen 9 3900X, Nvidia RTX 2080 Ti, and 64GB DDR4 RAM. This ensured we would have rapid deployment times and adequate hardware to learn and try new things using Unity Engine.

The VR hardware used was an Oculus Rift S attached to the same PC. Additionally, the application was ported over to Android to run on the Oculus Quest, using a Qualcomm Snapdragon 835 ARM based SoC.

Assets

The project utilizes meshes and models made of two Maya sites located in Guatemala. Two models of the excavation sites of El Zotz were used in this project. These models include the 3D mesh consisting of data points to recreate the area in a .fbx format as well as the texture data in .mat format. Along with the 3D models of the excavation site, a dancing spider monkey bowl model was also used. For a general purpose selection leve, an asset was downloaded off the Unity Store.

Audio recordings were also provided in English and some in Spanish. These are used for the narration system to teach the uninitiated about the area and points of interest. The menu and interactable audio spheres created were done through Unity and Photoshop to create the textures. All prefabs were also made in Unity, the hand models were from the original Oculus Integration package before we decided to abandon third party packages, though the animations were not also carried over.

Scripts

The scripts created provide most of the custom functionality used in the project. The majority of the work done for the project was in this section.

Additional documentation for functions, members, and other assets can be found at:
https://sites.google.com/ucsd.edu/xrchaeology/vr-development/vr-documentation

- Animator
  - Provides a simple rotation and floating effect controlled by a timer.
  - This script is applied on AudioSpheres to give them a more dynamic float-y effect. It has a few tunable parameters such as the rate of rotation and oscillation as well as having two states (rates of change) which can be modified by a timer.
  - It handles this by executing a calculation based on time every frame to update the position and rotation of the object.
- AudioSphere
  - Provides functionality to interact with NarrationSystem.
  - This script provides an identifier to objects to classify them as an interactable object that interacts with the Narration System. It allows the storage of a track name, used to find the corresponding audio track file.
  - It has two functions that can trigger an audio event from the Narration System, either by collision (user touching the sphere) or by a manual function call (utilized by the point system).
- CharacterControllerManager
  - Provides advanced control over CharacterController on the main XR Rig.
  - This script has references to the user's CharacterController and the Camera Offset in the user's XR Rig, as well as the XR Rig itself. This allows the script to have precise control and management over the locomotion and the movement of the user via the joystick.
  - It contains the functionality to move the entire user space (with camera offset accounted for) to be and face in a certain position and rotation. It also allows the user to virtually crouch at half their current height to assist with getting into small areas. It also allows the user to turn the camera.

- ConstantRotation
    - Provides a simple functionality to rotate constantly by specified rate.
    - This script provides the functionality of animating the Teleport Destination when the user is using the Teleportation System. It simply rotates an object by a specified amount using the time elapsed and updates once per frame. This allows the object to have an obvious presence in the scene.
- Fader
    - Using a Quad, provides functionality to fade the scene in and out.
    - This script provides functionality of fading the entire scene in and out with a quad placed in front of the user's camera. By specifying the time to fade to opaque (black) and clear (transparent) colours, Lerp() is used to linearly interpolate the two colours using the current time.
    - It can take a time to transition, or just use the default time specified using the public variable. It handles the update once per frame.
- GrabbableObject
    - Allows an object to be interacted with a Grabber
    - This script provides an identifier to let the Grabber System know that the object is interactable.
    - It has one variable to return if it is currently being interacted with already.
- Grabber
    - Script enables any trigger Collider to have grab interaction with GrabbableObject.
    - This script should be placed on the hand object of the user. Upon collision with an object, it checks if it is a GrabbableObject to verify interaction, then it will register it as a temporary object if so. Upon user input, it will "grab" it by setting the hand to be the object's parent. Upon user input again, it will release and set the object's parent to null.
    - Update currently handles tracking the velocity of the object. It updates a 2-depth queue of positions and when the user lets go of the object, it will calculate the velocity based on the last known position and current position and divide by the time elapsed.
- GuidedLevel
    - Contains fields necessary for use in GuidedSystem.
    - This script should be placed anywhere in a Excavation Site Prefab. It enables the Guided System's points of interest for the user to visit.
    - It has two fields, the Site ID, which identifies the site the level is currently based on and the list of points.
    - The list is of GuidedPoint, which simply contains a position, rotation, and AudioSphere to play when the point is visited.

- GuidedSystem
    - Provides a passive mode where the user can be brought to points of interest.
    - This script provides the core functionality for the passive mode. It has a reference to all the levels in initialization so there isn't a problem loading in the points dynamically (as seen with the Quest).
    - After the level has been loaded, the user's input will increment or decrement the counter that will move the user around to the points of interest. Each point will automatically place and face the user as well as play the corresponding AudioSphere that's related to the point of interest.
- Hand
    - Provides an enum and object type to identify a hand.
    - This script provides a type to identify which hand is which. It is used in a variety of scripts to allow the script to know which hand initiated the interaction.
    - Also, when placed on an object, it can be specified which hand it is in the Inspector.
- InputHandler
    - Handles the input for the XR system and currently is also the game system.
    - This script provides all the user input's functionality. It tries to acquire the two controllers in each of the player's hands and will try to find them if it ever loses it. It also supports haptic feedback on the controllers.
    - Every frame, the controller inputs are polled and stored in a variable for each button or joystick. Then in the same frame, it will execute the command bound to each input if possible.
    - User position updating with the joystick is handled in LateUpdate() so that it doesn't conflict with the other scripts that also modify the user's location.
- LinearLine
    - Renders a linear line and also provides collision functionality.
    - This script simply uses a LineRenderer to display a line originating from a position (the user's hand in the Point System) and tries to draw until it collides with something or runs into the max length. It is updated every frame based on the position and direction of the origin's Transform.
    - It is currently filtered to only collide with "UI" layer elements as it is used for the Point System for user interaction with objects.
    - Also, it provides the functionality to detect what it has collided with so the system can execute the correct command accordingly.
    - The colour and maximum distance can be configured using the Inspector.

- MenuInteractable
  - Enables a GameObject to have functionality with MenuSystem.
  - This script provides an object with an identifier so it can interact with the Menu System. It stores the option that it corresponds to and also contains functions to execute the command upon interaction via various methods such as collision and the Point System.
- MenuSystem
  - Provides functionality to change options or locations with a toggleable menu.
  - This script controls the main menu in the application. It contains a reference to the menu, as well as a reference to the buttons it contains. Finally, each site's description panel is also referenced so they can be dynamically displayed to the user.
  - It supports interactable buttons that can change colour based on their selection and tightly integrates with the rest of the systems.
  - When displayed, it stops rendering the excavation site so that the walls and other objects don't interfere with visibility and interaction of the menu.
  - It also has two panels on either side of the main menu to display the controls to the user whenever the menu is shown.
- NarrationSystem
  - Provides functionality with an AudioSource for per-recorded audio tracks as a narrator.
  - This script provides functionality with AudioSpheres and an AudioSource (presumably placed on the user) with multilingual support and error checking in case a file isn't found.
  - The list of supported AudioTracks is loaded with references upon start up and can be controlled to play any track or stop playing with functions.
  - Audio tracks in different languages will be specified by "_<lang>" appended to their names. Where <lang> is the language code. For example, a Spanish version of the track "test" would be "test_sp".
- ParabolicCurve
  - Script to generate and draw a parabolic curve that is affected by gravity originating from a given position.
  - This script uses a LineRenderer to display a parabolic curve originating at the specified origin and stopping when either it collides with something or has rendered the specified maximum number of segments.
  - Every frame, the LineRenderer is updated with positions calculated by the specified time interval to simulate a project in motion affected by gravity.
  - It is used in Teleportation to display the target destination location.

- PlayerSpawn
    - Provides functionality to return the position and direction for the player spawn in a level.
    - Used in an Excavation Site Prefab, it allows a position and rotation to be defined as the user's spawn location. This allows the user to be placed in a predetermined location and rotation when entering a level.
    - It provides two functions to simply return the position and rotation.
- PointSystem
    - Provides functionality to interact with objects using a liner line pointer.
    - Using the LinearLine and hand objects as origins, upon user input it will display a line that the user can interact with objects using. Objects that are currently supported are AudioSpheres and the Menu System.
    - It uses the LinearLine to detect objects and if a collision occurs, it will display a small pointer on the destination to indicate that the object is interactable.
- SiteManager
    - Script to provide level management.
    - This script provides the functionality to change levels for the user. It currently supports three sites, selection, Diablo, and M7-1. Each site has been referenced and more can be added via prefabs.
    - If a site does not currently exist, it will create one. And every time a site is created, it will move and rotate the user to the PlayerSpawn's parameters.
- SubtitleSystem
    - Provides functionality to update text that's placed in front of the user.
    - Using a custom shader and text that is placed in the front of the user. It will render text that can always be visible and not occluded. It also supports pre-defined transcripts from the Narration System to display.
    - Text management is handled by a queue system, each text is displayed for a number of seconds and override text can also be specified to display warnings and statuses to the user.
- Teleportation
    - Provides functionality to teleport the player around a scene.
    - Using the hand as an origin and the ParabolicCurve, TeleportDestination prefab, and input handler, it allows the user to preview a location to teleport to and execute the movement. If the location is invalid (no valid ground below) then it will not do anything.
    - This script was one of the first written and does not follow the conventions used by the newer scripts for movement.

4. Milestones

Development Process

The project started out with high hopes to modify an existing project and fix the bugs, but that quickly turned out to be a poor decision. The original project was built using Unity as well, but also used OculusIntegration, which is a third party package.

Unfortunately, the package was broken in many areas and was impossible to fix the bugs without learning the entire package and fixing many of the bugs manually. There also wasn't any documentation to start with so it was difficult to identify where problems were actually occurring. This led to development from scratch using only first party packages.

The first few weeks started out as expected, learning a new platform and language was challenging at first but substantial progress was seen. A basic application was created with movement that responds to objects and collisions as well as a teleportation system to move around. Then interactable objects were implemented and the Narration System was also added. Finally, the Menu System and Site Manager were implemented. Due to this substantial progress, the original roadmap was revamped.

In the following weeks, the core feature set of a Pointer System, Subtitle System, Guided System, and other minor features were completed. The remainder of the time was spent on debugging such as lighting or issues with movement and turning. Then finally, quality of life features such as the fade system and other minor details were completed.

Finally, documentation and demos were made for the first finalized version of the project.

Challenges and Solutions

I. This project began as trying to improve the current iteration of the existing project. But as we quickly found out, it was impossible to fix the bugs without rewriting the entire package that was used.
This led us to just start over from scratch so we know exactly how everything is implemented.

II. The movement in particular was difficult to implement. Due to the nature of this project and the environments it puts the user in, collisions require extremely delicate attention as they greatly impact the experience.
This led to us utilizing a modified version of the CharacterController built into Unity for the majority of the application, but it still required some fine tuning to get the correct functionality.

III. Not having access to a lot of assets also made things difficult, for example the hand asset is very basic and was taken from the original OculusIntegration. This caused some compromises to be made and sticking with more basic features without the complexity of models and animations.

IV. The overall inexperience with Unity and constantly changing implementations was difficult to grasp to create the desired functionality. For example, custom shaders were written to implement certain functionality. But without the knowledge of Unity's rendering pipeline, OpenGL knowledge can only take one so far.
These resulted in compromises made, again, to simply the design and overall features set.

V. Due to solo development and lack of predefined documentation, again due to the learning process, the code style and solutions constantly changed and were experimented with.
In the end, after functionality was implemented and bugs resolved, the code was attempted to be standardized but some aspects remain too tightly integrated to change easily. This would require a more time consuming redesign and isn't desirable for a development cycle.

5. Conclusion

In this project, we used Unity Engine and an Oculus Quest / Rift S to design and create a VR application. This application allows the user to travel to archeological sites virutall and is targeted at researchers and interested people alike. It's designed to be easily accessible compared to traditional methods and also show that we have technology today to start on the path of digital preservation of archaeological sites.

Built using C# and the XR Interaction Toolkit, the application is designed to be easily expanded upon and features easy to implement. The intent was to make the system as modular as possible so assets can be added and removed without very much trouble. Currently, in the initial version's state, there is much to be added still but a proof of concept that the system works is the primary motivation of the project.

6. References

1. Scerri, Eleanor M. L.; Thomas, Mark G.; Manica, Andrea; Gunz, Philipp; Stock, Jay T.; Stringer, Chris; Grove, Matt; Groucutt, Huw S.; Timmermann, Axel; Rightmire, G. Philip; d'Errico, Francesco (1 August 2018). "Did Our Species Evolve in Subdivided Populations across Africa, and Why Does It Matter?". *Trends in Ecology & Evolution*. **33** (8): 582–594. doi:10.1016/j.tree.2018.05.005. ISSN 0169-5347. PMC 6092560. PMID 30007846.
2. *The Origin and Development of the Cuneiform System of Writing*, Samuel Noah Kramer, *Thirty Nine Firsts In Recorded History*, pp. 381–383.
3. Hunter, Michael (1975), *John Aubrey and the Realm of Learning*, London: Duckworth, pp. 156–57, 162–66, 181, ISBN 978-0-7156-0818-0
4. Bourgeois, J. and Meganck, M. (eds.) (2005). *Aerial Photography and Archaeology 2003. A Century of Information.* Archaeological Reports Ghent University 4. Ghent: Academia Press. ISBN 90-382-0782-4
5. "Archaeology Applications with Photogrammetry." *PhotoModeler*, 11 Oct. 2018, www.photomodeler.com/pm-applications/photogrammetry-academics-research/archaeology/.