St. Augustine Forts in VR

Team A34

Samuel Hussain

Thuc Luong

Jameel Mohamed

James Stricker

Jack Vo

Shane McDermott

Julian Collazo

Team Sponsors

Amy Giroux

Brook Miller

Table of Contents

Minimum Viable Product

Our minimum viable product will consist of

- A VR program that runs at a minimum target framerate of 90 fps on the Quest 2.

- A reconstruction of fort Matanzas to the specifications of the provided documents
    - A reconstructing of the surrounding geoplane as a walkable area for players.
- An approximate and shortened reconstruction of the Matanzas River between fort Marion and fort Matanzas depicted as it was before 1865 that will obscure the view between the forts.
    - LOD or seamless scene loading functionality between each fort that will be implemented while traveling between forts.
- A boat simulation to transport the user down the Matanzas River from the previously constructed fort Marion to fort Matanzas.
    - A small boat model to facilitate this.
- Fixing camera height bug present in the previous project
- Removal of any unused assets.

Timeline/Roadmap

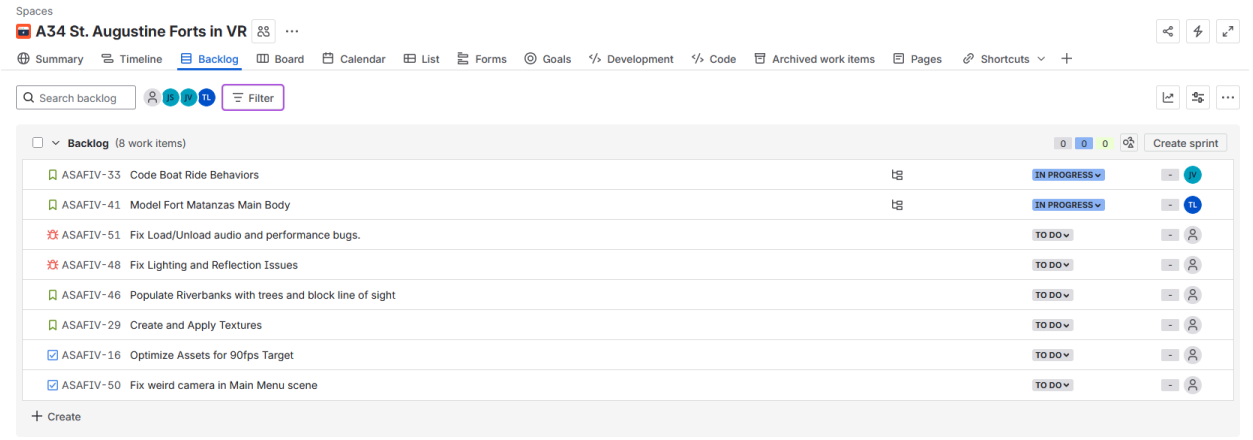
- Sprint 1 (10/13/25 - 10/27/25)
    - Restore VR Functionality
        - Fix initial height mismatch
    - Model Tower
    - Model Fort Matanzas Geoplane
        - Create Heightmap
        - Texture and Scale Terrain

    - Extend Riverbank
        - Create Heightmap
        - Texture and Scale Terrain
    - Model The Boat
        - Model Main Geometry
        - Apply Textures
    - Remove Unnecessary Assets
        - Catalogue Unreferenced Assets
        - Remove Unreferenced Assets
        - Test After Removal
- Sprint 2 (11/4/25 - 11/18/25)
    - Populate Riverbank and Matanzas with trees
    - Fix height issues when teleporting
    - Model Fort Marion First Floor
        - Model Exterior
        - Model Interior
    - Model Stairs and Ladders
    - Create Boat Ride Behaviors
        - Pathing Behaviors
        - Seating Behaviors
    - Fix Water Reflection Lighting Issues
- Remaining time in SD1 will be dedicated to researching more specific topics, fixing small bugs, and completing document assignments
- Sprint 3 (Early Spring SD2)
    - Model Fort Marion Second Floor Exterior
    - Add Details to First Floor Interior
    - Fix Minor Visual Bugs

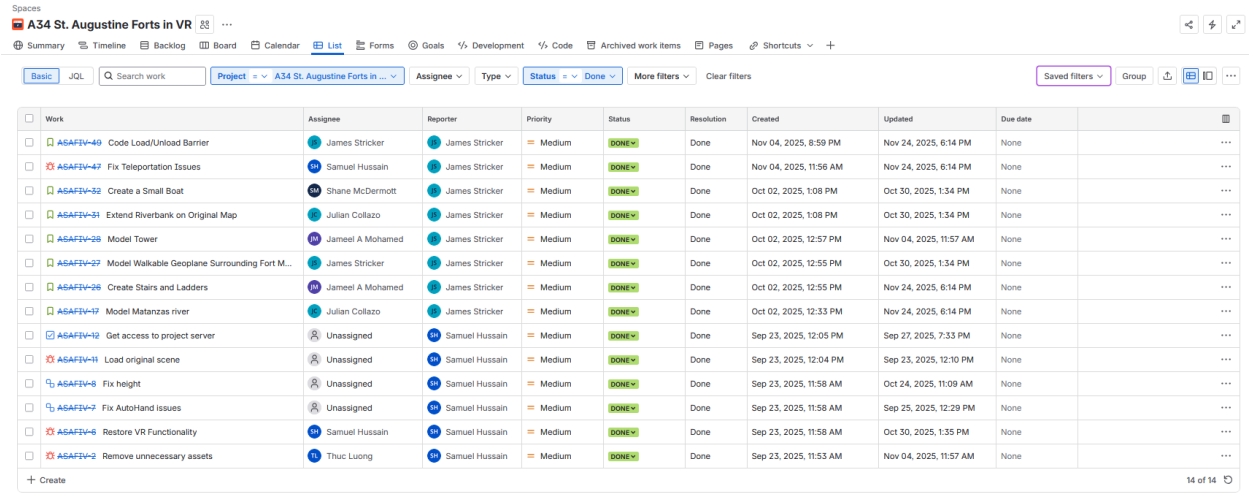
    - Create Fort Matanzas Textures
    - Stress Test Running on Quest 2 APK
    - [Additional Space Reserved For Emergent Bugs or Unfinished Tasks]
- Sprint 4 (Mid Spring SD2)
    - Optimize Assets for 90FPS and Future Scalability
    - Diagnose Portions of the Project That Need Polish
    - [Additional Space Reserved For Emergent Bugs or Unfinished Tasks]
- Sprint 5 (Mid-Late Spring SD2)
    - Improve Behaviors or Visuals Diagnosed in Sprint 4
    - [Additional Space Reserved For Emergent Bugs or Unfinished Tasks]
- Get Sponsor Feedback on All Complete Features
- Sprint 6 (Late Spring SD2)
    - Apply Any Final Changes Suggested in Sponsor Feedback
    - Attempt Additional Features if MVP is Satisfied Already
- Time In-Between and After Sprints Will Be Dedicated to Compiling Information, Completing Documentation, And Communicating Progress.

Backlog and Accomplishments

Our Jira Backlog and Completed Tasks are shown below. Our next sprint has not been organized yet, so there are no current sprints to show.



[Figure 1 Jira Backlog]



[Figure 2 Finished Jira Tasks]

Initial Research and Historical Information

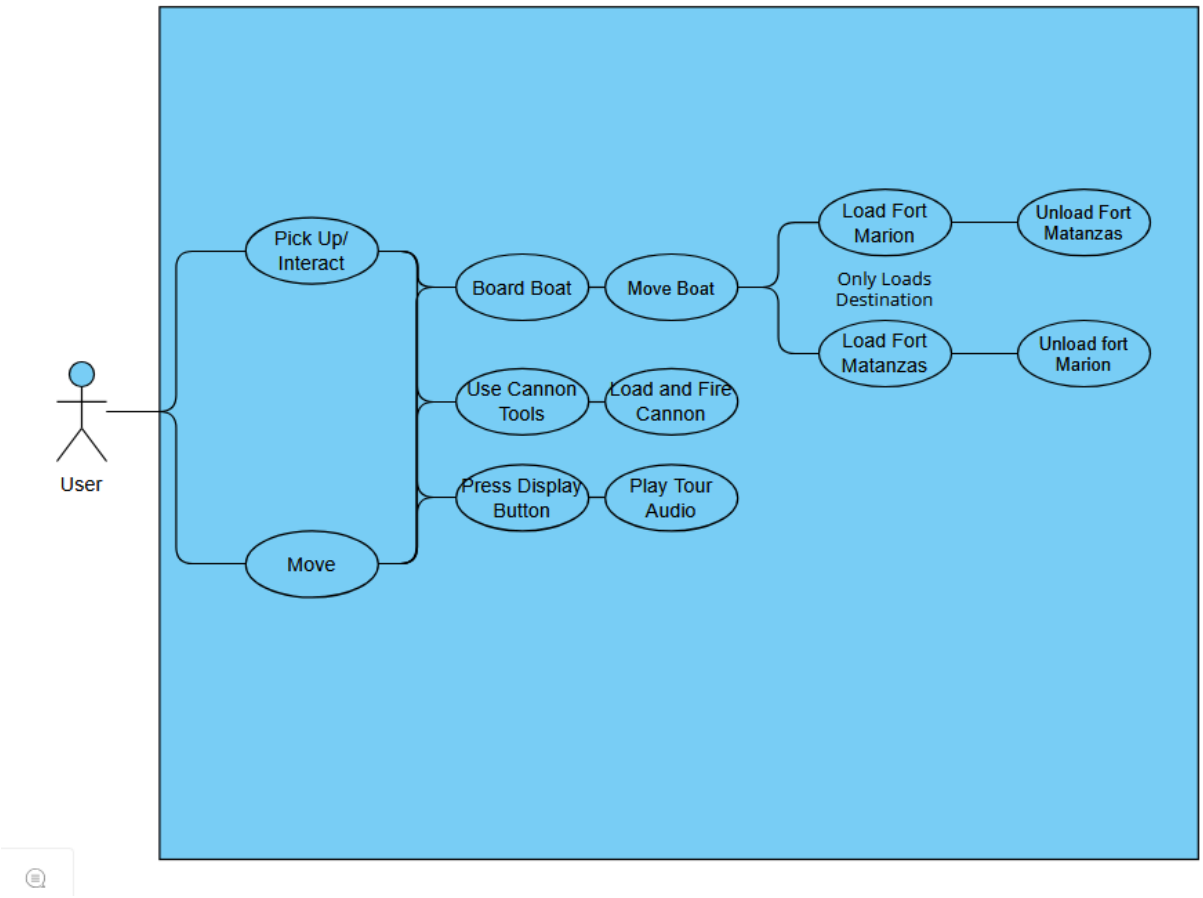
Since we were late to establish a sponsor meeting, we had plenty of time to do research and pool documents to make sure we have a good historical and technological understanding of this project. The historic period we are aiming for geographically is around the civil war era. This is before the Bridge of Lions or the toll bridge that preceded it were constructed, and also before the Army Corps of Engineers dug out artificial canals in the area. Ironically, during this period Fort Matanzas would have been in disrepair before reconstructive efforts were made with the creation of the National Park Service. For the Fort itself we are reconstructing it from architectural documents made in the 1930's, but the goal is to model it as it stood during its Spanish occupation. While Fort Marion was in use by the US military after its years as a Spanish Fort, Fort Matanzas was not seen as strategically useful by the time the US had Florida. We were also able to find multiple maps of the areas around Fort Matanzas from different eras, with keys designating types of foliage and residential areas. We compared these maps to see how the area developed and used them to model and scale our terrain.

On the technological side, we researched how to use Blender, Unity, and Meta Quest development tools. For Blender we learned the basics of 3D modeling and setting up reference images, this is covered more in depth in the later modeling sections. Learning Unity had us researching development tools for VR, the existing tools used in this project, and how to navigate

issues that were already present in the project. As for Meta Quest development tools, we learned how to operate the VR headset, and run the project in it. We were able to run the project while directly connected, through wireless link, and as an APK directly on the Quest. This also led us to create a Developer Hub group so we can coordinate and keep track of exported APK versions of our project.

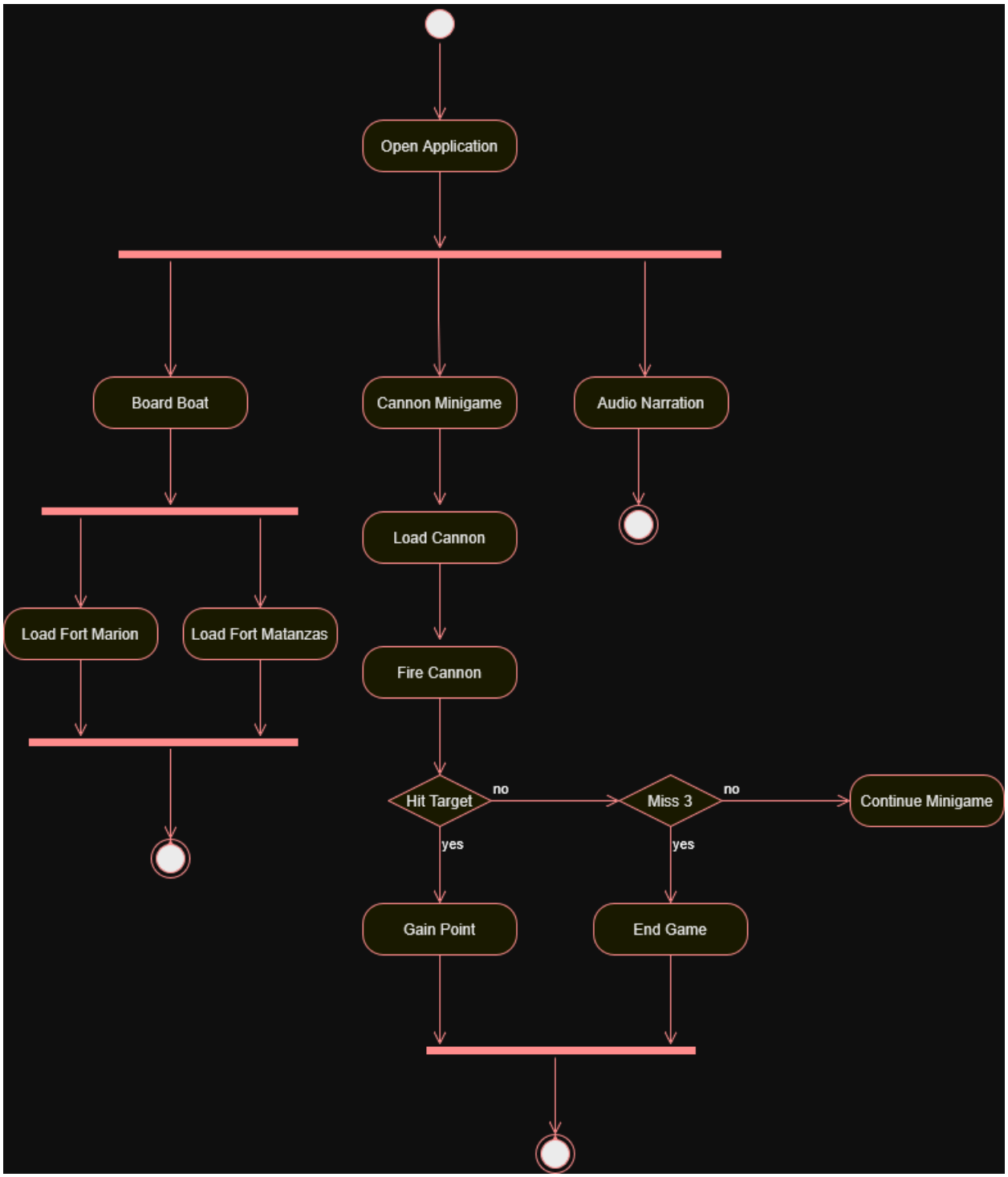
Diagrams

Use Case Diagram



[Figure 3 Use Case Diagram]

Activity Diagram



[Figure 4 Activity Diagram]

Individual Motivations

Samuel Hussain

Ever since I began learning about history many years ago it has been a captivating topic for me, especially the history of the state of Florida. So when I saw this project pitched to the class it was something that grabbed my attention almost instantly. Not only is the topic interesting to me, but combining two very different areas of study, technology and the humanities, was very intriguing to me. I was curious how we could develop a game that would not only educate its users, but keep them entertained and immersed in a digital world from another time. Another motivator for me is that I have never worked with virtual reality development before, and I believe that mastering the skills associated with that would make me a better and more versatile software engineer. Aside from VR development, I don't have much experience with the platform of virtual reality as a whole, but it has always been something that intrigued me. Diving headfirst into this new digital realm seemed very exciting for me and was an opportunity to learn more about new and rapidly developing technologies. I am motivated to complete this project to the best of my ability and provide an enthralling and informative digital experience to anyone who steps into our simulation of the forts of St. Augustine.

Julian Collazo

This project caught my attention right away because it gave me a chance to work with virtual reality, something I have always been interested in but never had the chance to do myself. I already own a VR headset and spend a lot of time exploring immersive worlds, so the idea of creating one was really exciting. I also like that the environment has history, it gives the space more meaning and makes designing interactions more interesting.

Mostly, what drives me is the technical and creative challenge of VR. I enjoy figuring out how to make a world feel alive and responsive, and this project gives me the freedom to experiment with that while still working within a historical setting.

James Stricker

My motivation for this project is a desire to learn VR development and an interest in history and archaeology. I am also driven by previous experience and enjoyment of game development, computer graphics, and modeling. I'm interested in creating this project with attention to historical accuracy and understanding of not just the forts at the time, but the surrounding area as well. I also want to help preserve a national park digitally as it once stood so that those who may not be able to visit it, can still see it accurately scaled as if they were there. I am looking forward to contributing to a genuinely exciting project that I feel I have the experience to guide my teammates and efficiently learn new skills.

Thuc Luong

When I was younger, I became interested in history from playing video games that explored the genre. My intrigue led me to read into historical events beyond the curriculum. I was surprised to see a project that combined my passions and hobbies during the presentations. I had assumed that I would not get the opportunity to engage in anything related to history after enrolling as an undergraduate Computer Science student. Additionally, the virtual reality aspect of this project was a major contributor to my motivation. I have always been interested in VR, but I never had the opportunity to try or obtain a headset. I believe VR is still a largely unexplored platform for games or educational experiences. As for my experience working with technologies associated with the project, I am very inexperienced. I hope to learn much about VR, Unity, and 3D modelling.

Jack Vo

This pitch stood out to me from the rest as soon as I heard it presented. The idea of recreating the St. Augustine forts in virtual reality immediately stood out as something meaningful and useful in the real world. The forts are such a distinctive part of Florida’s history, and the opportunity to help bring that experience to people who may never get the chance to visit in person felt both exciting and worthwhile. Allowing someone to step into a virtual boat ride through the forts adds a level of immersion that traditional photos or videos can’t offer, and I wanted to be part of building something that could make history feel alive, accessible, and engaging.

On top of the historical aspect, this project aligned perfectly with the technical skills I've been wanting to grow. I've always been interested in VR, but never had a structured reason to dive into it, especially at the scale of an explorable environment. Working on this gave me the chance to strengthen my Unity experience, experiment with 3D environments, and contribute to a project that could realistically be expanded or maintained in the future. It felt like the ideal combination of creativity, technical challenge, and historical storytelling. Ultimately, I chose this project because it wasn't just another assignment—it was something I genuinely wanted to see come to life.

Shane McDermott

Throughout high school, I was part of the modeling and simulation program that was offered. I learned how to develop high-quality 3D models using professional software like Autodesk Maya. The St. Augustine Forts in Virtual Reality pitch immediately piqued my interest when Dr. Giroux explained how she needed a team to model the remaining fort in St. Augustine and add the functionality to travel between the two forts. She also mentioned that the project would be done in Unity, which I have used briefly before for a small simulation project.

This project ended up being my top choice because it sounded like the most interesting and unique option. With AI on the rise, it is becoming much easier for the average person to build a website or app in just a few hours, so I wanted to take on something that could not simply be generated through a prompt. A simulation project requires creativity, real modeling

work, and a deeper level of detail that AI still cannot fully replicate. I liked that it involved more than just coding and that I would be able to model real objects and help recreate the environment of St. Augustine in a way that feels realistic inside a headset. It felt like the perfect combination of technical skills and creativity and a chance to build something meaningful that stands out.

Jameel Mohamed

I am excited to contribute to this project because it combines creative digital design with historically accurate storytelling in a VR environment. Having visited Fort Marion and spent time in St. Augustine, I am motivated by the opportunity to help recreate that experience for users who may never visit in person. This work allows me to build on my skills in Blender by modeling accurate reconstructions of real structures, while also gaining valuable experience in Unity for VR and game development.

My primary motivation is to improve the visual fidelity and environmental detail of the existing project while applying those same standards to our newly developed sections. During my visit to the Castillo de San Marcos, I was struck by how its textures, lighting, and atmosphere communicated a strong sense of history. I want Fort Matanzas in VR to evoke that same immersive presence.

My goals focus on two main areas. First, I aim to enhance the fort models with realistic surfaces and refined architectural features to make the

environment feel lifelike. Second, I plan to support the development of the boat ride experience by incorporating regionally accurate details such as wildlife, vegetation, natural water movement, and ambient sound to make travel along the Matanzas River more engaging and believable.

Overall, I hope to help create an atmospheric VR experience that feels authentic, educational, and memorable, allowing users to connect more deeply with the history of St. Augustine's forts.

Societal Impacts

By creating a realistic and informative simulation of two different forts in St. Augustine, we open the doors to a new, unorthodox way of learning about and experiencing significant locations and events in history. History is usually taught through reading textbooks or listening to lengthy lectures and while this works for some people, many others struggle to learn about new concepts this way. This project makes learning about Florida's history more accessible to those who struggle with traditional learning, while still being an exciting simulation for anyone to experience. We believe that creating an interactive way to teach history can help a broader range of people become more educated and interested in the history of their state. This increased interest may also result in more visitors to the real life sites of Fort Marion and Fort Matanzas which are represented in our project. More visitors to these historical sites means the state receives more money that can go towards the preservation and improvement of the forts in St. Augustine.

These funds can also go towards conservation of the coastal and wetland ecosystems that surround the various forts of St Augustine.

Legal, Ethical, and Privacy Concerns

Legal issues may have risen when working on the project if our reference material was copyrighted. We did not have to worry about that because we used firsthand reference material of the fort blueprints, and photos that our sponsor provided us. Furthermore, we made sure that all assets such as textures or models were either created by our team or sourced from public locations that allowed for reuse. Due to the forts being public historical sites, there were no restrictions on their design specifically but nonetheless we remained cautious to not use any copyrighted reference material.

There are some ethical concerns that we had to consider while working on the project, that forced us to consider the balance between immersion and responsible design choices. For example, the cannons were an integral part of the forts' history and so we had to represent them in a way that was non-violent and instead educational. If we changed them so that they were more graphic and combative, then we are purposefully taking away from the main goal of the game which is to be educational and a learning experience. Similarly, we had to avoid creating historically-specific NPCs in an effort to prevent inaccurate portrayals of real individuals or unintentional reinforcing of stereotypes. Instead, NPCs are more basic and generalized in order to not imply a certain viewpoint.

Privacy concerns were not an issue that we ran in to during the development of the project. If we had implemented a sort of sign in system that stored emails and passwords, then this would not be the case. The overarching idea and execution of the game doesn't require a sort of authentication feature and so we figured it would save much more hassle for little disadvantage if we didn't include a sign in.

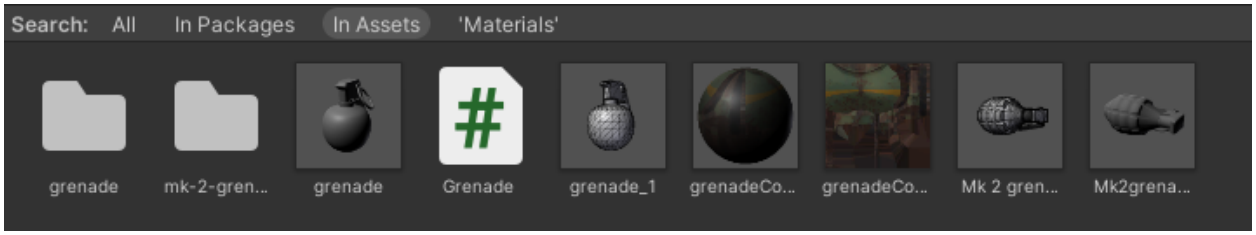
Removal of Unnecessary Assets

This task involved the identification and deletion of assets deemed unnecessary to the project. Our sponsor specifically mentioned that the size of the project had ballooned to over 10 gigabytes after the previous team had finished their work. Dealing with this technical debt became our team's priority before we would add new assets such as the small boat and the riverbank extension. Therefore, the version of the project that has been optimized would become the baseline to which everyone pulled from. This was to prevent people from pushing back the deleted assets into the Github repository. After the completion of this task, we plan on pushing it back into the backlog. The reason for this is that as our team creates and utilizes new assets, we will inevitably leave unused and possibly large files. This would require another evaluation of the project files for cleanup.

Defining the Unnecessary

After discussing with our sponsor, we were instructed to recreate both Fort Marion and Fort Matanzas as they were during the American Civil War.

Therefore, the criteria of assets to delete came to include those that are ahistorical or inappropriate for the American Civil War setting.



[Figure 5 Mk. 2 Grenade]

[Figure 6 Halloween themed prefabs]

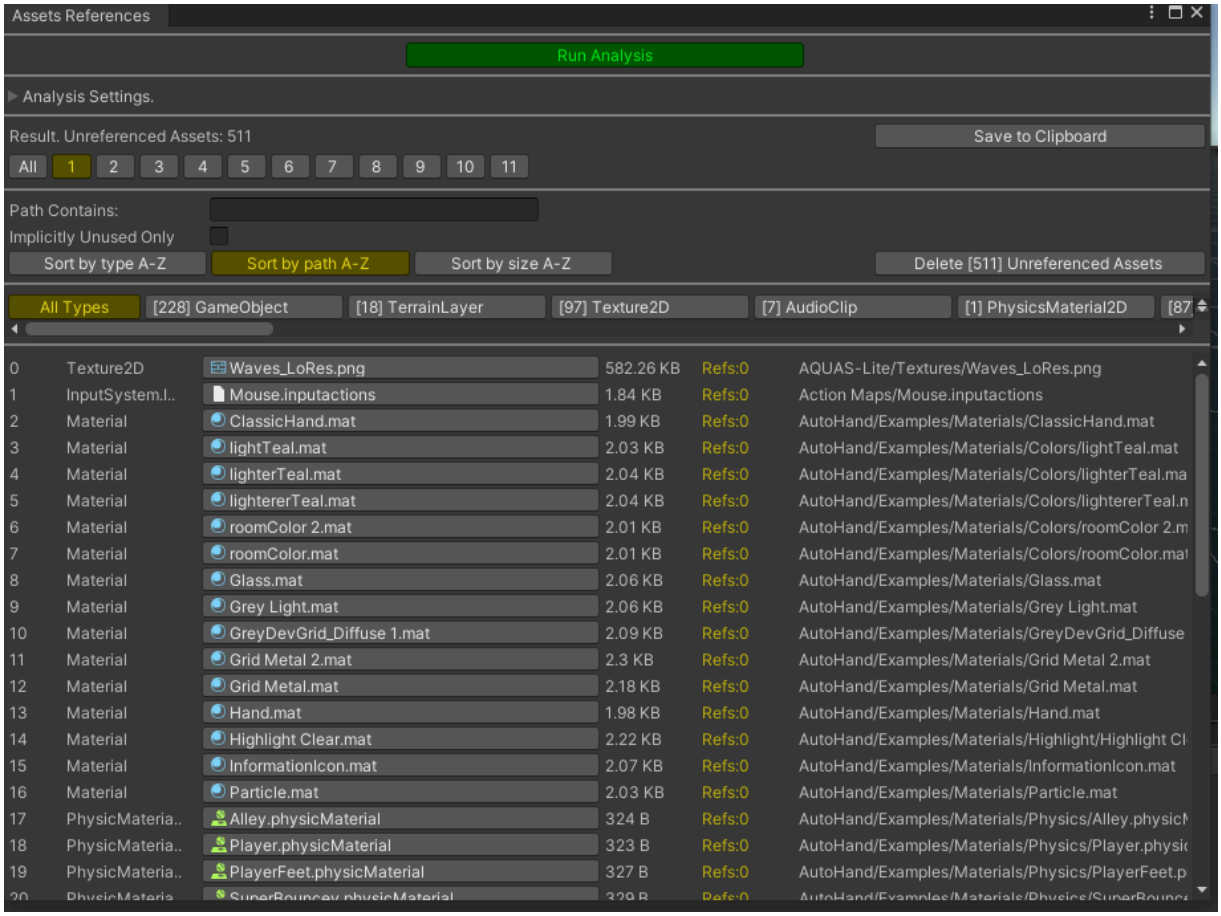
Besides assets that did not fit within the Civil War period, we focused on assets that had a large impact on the total file size of the project. Some examples of large assets included:

- 4K Textures
- Audio Files
- Complex 3D Models

Besides assets that did not fit with the theme of the project, we determined that assets that were not directly used by the main scene were to be manually reviewed. An example of this is a terrain pack that contains many types of ground textures. We would have to determine the necessity of each ground type after assessing the soil composition around Fort Marion and Fort Matanzas. Some free asset packs that the previous team used included demos. These demos were determined to not be essential to the final project. Finally, we identified some duplicate files which were obvious candidates for deletion. There were multiple ground and tree packs that fit this criterion.

Identification Methods

As for our methods to identify the unnecessary assets defined by the previous section, we utilized a tool called the Dependencies Hunter Unity3D Tool. This tool was created to search for and delete unreferenced assets in a Unity Project.



[Figure 7 Default Dependencies Hunter UI Analysis]

However, we found through our research that the Dependencies Hunter had deleted essential files to the projects of others that had used it before. They had used the tool's ability to automatically delete all the unreferenced assets. Because of this, we chose to not use the deletion feature of the Dependencies Hunter and instead opted to delete unreferenced assets after extensive manual review. Initially, we tried to review and delete asset files in the descending order of their file size. This

method proved to be confusing and time consuming for the team member assigned to the task. The teammate had to flip through many different file directories as the largest files were seldom located in the same folder. We improved our methodology by going through each folder and then using the Dependencies Hunter to list the unreferenced files by size in that folder.

Deleted Files/Folders

- Assets/Models/CANDLESTICKS
    - Candle models that weren’t in the main scene.
- Assets/Terrain/Tom's Terrain Tools/Unity Terrain Assets/Trees Ambient-Occlusion
    - This contained a variety of tree types such as the Alder, Mimosa, and Sycamore that weren’t used in the main scene.
- Assets/Terrain/4K PBR Realistic Ground Textures
    - Although 4K realistic ground textures would be visually beautiful, the Meta Quest 2’s hardware is too weak to justify having individual 32MB textures.
- Assets/TerrainSampleAssets
    - This was a duplicate folder of one in the Assets/Terrain folder.
- Assets/_TerrainAutoUpgrade
    - This folder was empty.
- Assets/AQUAS-Lite/Demo/DEMO.unity
- Assets/AQUAS-Lite/Demo
- Assets/AQUAS-Lite/Prefabs

    - The AQUAS-Lite folder contained a demo and textures made specifically for it. Additionally, there was an unused waterplane prefab.
- Assets/Audio/ChapelDisplay.mp3
- Assets/Audio/ContestOfNationsDisplay.mp3
- Assets/Audio/CoquinaCanvasSmall.mp3
- Assets/Audio/ExampleAudioPrompt.mp3
- Assets/Audio/MoeyauhayistHeapOfBirds.mp3
- Assets/Audio/NaturalCoquinaConstruction.mp3
- Assets/Audio/PedroMenendezDeAviles.mp3
- Assets/Audio/SirFrederickHaldimand.mp3
- Assets/Audio/SoldierCarving4.mp3
- Assets/Audio/SoldierDisplayBoard.mp3
- Assets/Audio/SpanishColonizationDisplay.mp3
- Assets/Audio/WilliamCarr.mp3
    - Unused large audio files.
- Assets/Books/Scenes
    - This folder contained test scenes that showed how the book models looked.
- Assets/CharacterPack Lowpoly (FREE)
    - Although these were used as NPCs in the main scene, we decided to remove them for the time being.
- Assets/Darth_Artisan/Free_Trees/Scenes
    - Demo scenes.
- Assets/FotoCarsFree
- Assets/Prefabs/Cars.prefab

    - Cars did not fit in with the setting, nor were these prefabs used in the main scene.
- Assets/Free forest pack/Black tree/Materials
- Assets/Free forest pack/Black tree/dead tree model01.png
- Assets/Free forest pack/Black tree/dead tree model02.png
- Assets/Free forest pack/Black tree/dead tree model03.png
- Assets/Free forest pack/Black tree/dead tree model04.png
- Assets/Free forest pack/Blur tree
- Assets/Free forest pack/Textured tree
    - Unused dead and blurred tree models.
- Assets/Free Pixel Art Forest/Demo
- Assets/Free Pixel Art Forest/Preview
    - Example demo and preview.
- Assets/Free Pixel Art Forest/PSD
    - Photoshop layers for the Free Pixel Art Forest.
- Assets/Free Pixel Art Forest/You may also like
    - This folder advertised some other packs made by the creator.
- Assets/Free Pixel Art Forest/PNG/Background layers/Materials
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0000_9.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0001_8.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0002_7.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0003_6.png

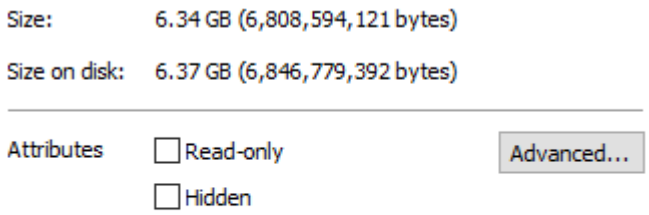
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0004_Lights.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0006_4.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0007_Lights.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0008_3.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0009_2.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0010_1.png
- Assets/Free Pixel Art Forest/PNG/Background layers/Layer_0011_0.png
    - Unused background layers except the fifth one.
- Assets/Free_Trees/Scenes
    - Example scenes for Free_Trees.
- Assets/JaneArtWork
    - The JaneArtWork folder contained the Halloween Free Pack which was not appropriate for the Civil War era representations of the forts.
- Assets/Medieval props/ShowRoom.unity
- Assets/Medieval props/ShowRoomSettings.lighting
- Assets/Medieval Props Asset Pack/DemoScene.unity
    - ShowRoom and DemoScene showcasing the props and prefabs.
- Assets/MedievalTavernPack/Scenes

    - Demo Unity Scenes for the Medieval Tavern Pack.
- Assets/Rock Package/Scene
    - This contained a scene for the rock map and its lighting settings.
- Assets/Simple Foods/FoodDemoScene.unity
    - Another demo scene showcasing the food prefabs. The Simple Foods folder may be deleted after further review.
- Assets/Terrain/_TerrainAutoUpgrade
    - This folder was not empty, unlike its duplicate in the Assets folder. However, the terrain layer files were unused, leading to this folder’s deletion.
- Assets/Terrain/Brushes
    - Brushes were already being used elsewhere.
- Assets/Terrain/Ground textures pack (Duplicate)
    - This Ground Textures pack folder was a duplicate of the Assets/Ground textures pack folder, so we chose to delete the one that was not being used by the main scene.
- Assets/Terrain/TerrainSampleAssets/TerrainBrushes
    - More unused terrain brushes.
- Assets/Terrain/TerrainSampleAssets/Textures/Heightmaps
    - These heightmaps included terrain layers such as snow and black sand.
- Assets/Terrain/Tom's Terrain Tools/2DTools/Scenes
    - Demo scenes showcasing the 2D Tools from Tom’s Terrain Tools.
- Assets/Terrain/Tom's Terrain Tools/Example AutoMagic
    - Unused example raw heightmap files.

- Assets/Terrain/Tom's Terrain Tools/Unity Terrain Assets
    - Terrain textures and materials that weren’t used in the main scene.
- Assets/AutoHand/Examples/Mesh/mk-2-grenade
    - The Mk 2 grenade was not appropriate for the time period.
- Assets/AutoHand/Examples/Scenes/OpenXR/Demo.unity
    - Demo for OpenXR that isn’t necessary.
- Assets/AutoHand/Examples/Prefabs/Simple/PistolCartoon Variant.prefab
    - Although firearms were invented by the time of the Civil War, the cartoon pistol did not fit the realistic style. It was also modeled after modern pistol designs.
- Assets/AutoHand/Examples/Prefabs/Kitchen/Fridge.prefab
    - Ahistorical refrigerator prefab.
- Assets/AutoHand/Examples/Scenes/XR/Demo.unity
- Assets/AutoHand/Examples/Mesh/diamond.fbx
- Assets/AutoHand/Examples/Prefabs/Kitchen/BigSoda.prefab
    - Some more objects that did not fit in with the theme of the project.
- Assets/Books/Prefabs/HDRP
- Assets/Books/Materials/HDRP
- Assets/Books/Prefabs/Standard/BooksAll.prefab
- Assets/Books/Prefabs/URP
- Assets/Books/Materials/URP
- Assets/Books/Prefabs/Standard/Scrolls
- Assets/Books/Materials/Standard/ScrollPaper.mat
- Assets/Books/Textures/URP/ScrollsPaper_Normal.tif

- Assets/Books/Textures/URP/ScrollsPaper_MetallicSmoothness.tif
- Assets/Books/Textures/URP/ScrollsPaper_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Tube1_Normal.tif
- Assets/Books/Textures/URP/Table1_Normal.tif
- Assets/Books/Textures/HDRP
- Assets/Books/Textures/URP/Table1_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Table1_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Tube2_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Tube2_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Candles_Normal.tif
- Assets/Books/Textures/URP/Candles_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Candles_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Inkwell1_Normal.tif
- Assets/Books/Textures/URP/Inkwell2_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Tube1_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Inkwell1_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Inkwell1_AlbedoTransparency.tif
- Assets/Books/Textures/URP/Inkwell2_MetallicSmoothness.tif
- Assets/Books/Textures/URP/Inkwell2_Normal.tif
- Assets/Books/Textures/HDRP
    - The Books folder contained book props unreferenced in the main scene.
- Assets/Laterns and candles/textures/CarvedLampPost2 [Specular].tif
- Assets/Laterns and candles/Meshes/CarvedPost2.fbx
- Assets/Laterns and candles/Meshes/CarvedPost2_double.fbx
- Assets/Laterns and candles/Meshes/LampPost.fbx

- Assets/Laterns and candles/Meshes/LampPost_double.fbx
- Assets/Laterns and candles/Prefabs/LampPost_double.prefab
- Assets/Laterns and candles/textures/LampPost [Specular].tif
- Assets/Laterns and candles/Laterns demo.unity
- Assets/Laterns and candles/Laterns demoSettings.lighting
- Assets/Laterns and candles/Prefabs/CarvedPost2_double.prefab
- Assets/Laterns and candles/Meshes/Latern5.fbx
- Assets/Laterns and candles/Prefabs/Latern5.prefab
- Assets/Laterns and candles/Materials/Latern5.mat
- Assets/Laterns and candles/textures/Latern5 [Specular].tif
    - The “Laterns and candles” folder was notable for its textures having large file sizes. The prefabs had references to separate materials which also had to be deleted.
    -



[Figure 8 Size of Project after Deletions]

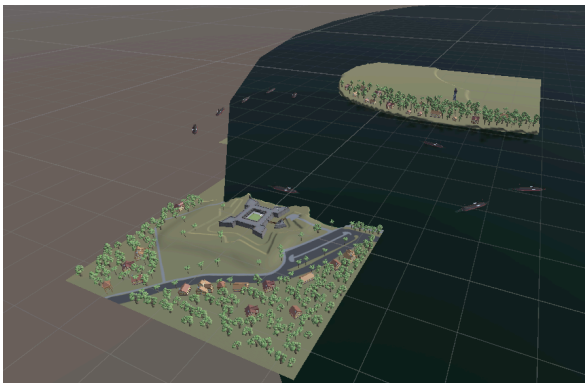
Extending the Riverbank

Objective

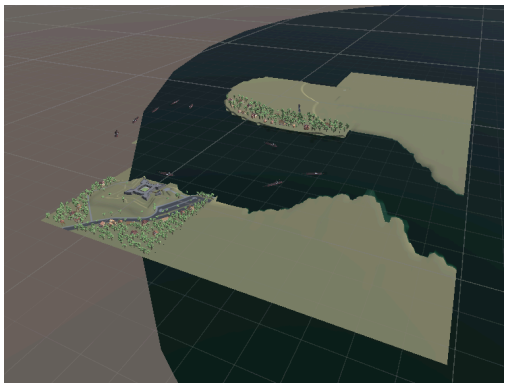
As part of our main requirements for this project, we need to create a river that connects Fort Marion to Fort Matanzas, allowing the user to experience a riverboat ride between the two forts. The Matanzas River is quite lengthy, and if scaled accurately, the boat ride would take far too long. To address this, we decided to scale the river down while maintaining its recognizable geography, ensuring that the experience remained both realistic and engaging.

Research and Preparation

We focused on extending the existing riverbank created by the previous senior design team and generating a new height map of the Matanzas River area. Before starting, we researched different techniques for creating and shaping terrain in Unity. We reviewed tutorials on manual terrain sculpting and height map generation to understand how to form natural landscapes. To practice, we set up a separate Unity project as a sandbox environment where we could freely experiment and learn.



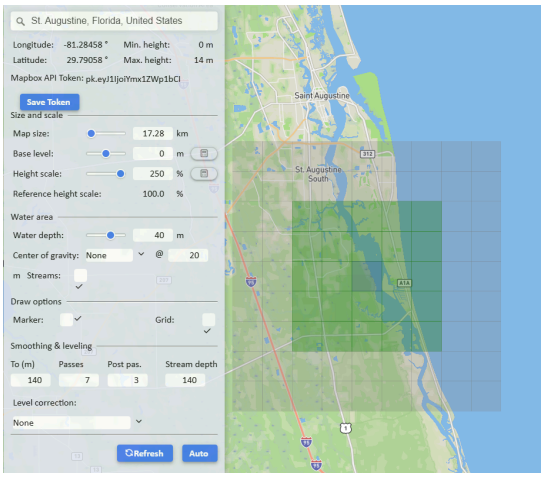
[Figure 9 Original Riverbank]



[Figure 10 Extended Riverbank]

Height Map Creation and Terrain Generation

For the height map, we used the website heightmap.slydark.pl, which allowed us to select a specific real-world area and export it as a height map file. We selected the Matanzas River region and generated a PNG file that could be imported into Unity. Using Unity’s Terrain Tools, we imported the height map and adjusted the length, width, and maximum height parameters to generate the terrain accurately. It was important that these settings matched the source image to ensure proper scaling and appearance.



[Figure 11 Heightmap Tool]

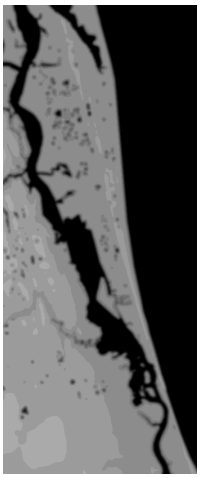
Once imported, we noticed that the generated terrain was extremely large. Since players would only travel along the river, most of the surrounding land would never be explored and would only consume unnecessary computing resources. To optimize performance, we used Unity's terrain tools to split the terrain into smaller tiles and removed the sections that were not needed, keeping the focus on the playable area.

Troubleshooting and Refinement

During the process, we encountered an issue where the generated terrain appeared too flat. The exported height map was very dark because Florida's natural elevation changes are minimal. To resolve this, we learned how Unity interprets grayscale values. Each pixel's brightness corresponds to elevation, and these values are multiplied by the maximum height setting. Using GIMP,

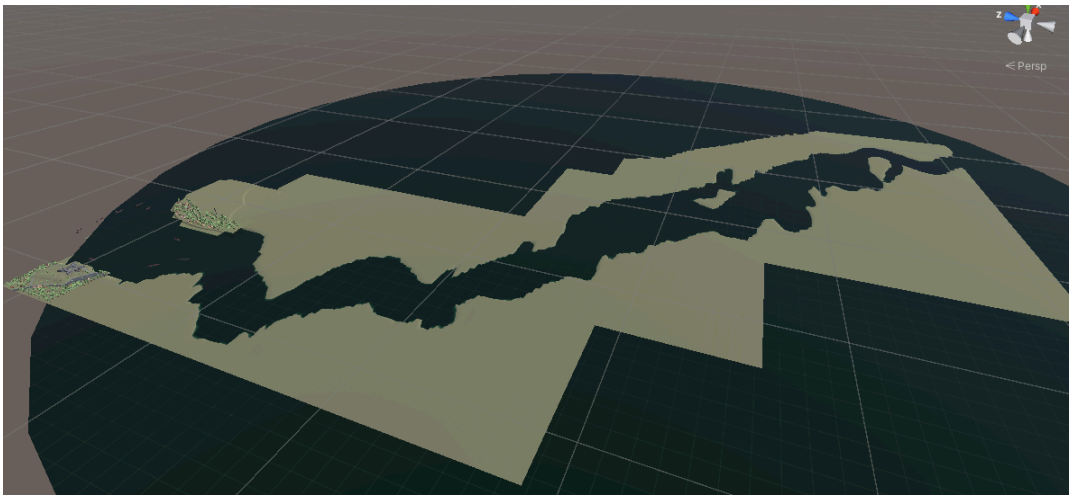
we brightened the higher regions of the image so that the tallest points appeared white (value of 1), allowing Unity to represent them correctly.

After adjusting the color values and rescaling the terrain, we regenerated the height map within Unity to finalize the corrected riverbank. This ensured that the Matanzas River appeared at the correct scale, flowed smoothly into the surrounding terrain, and provided an immersive foundation for the future riverboat ride.



[Figure 12 Generated Heightmap]

[Figure 13 Scaled Heightmap]



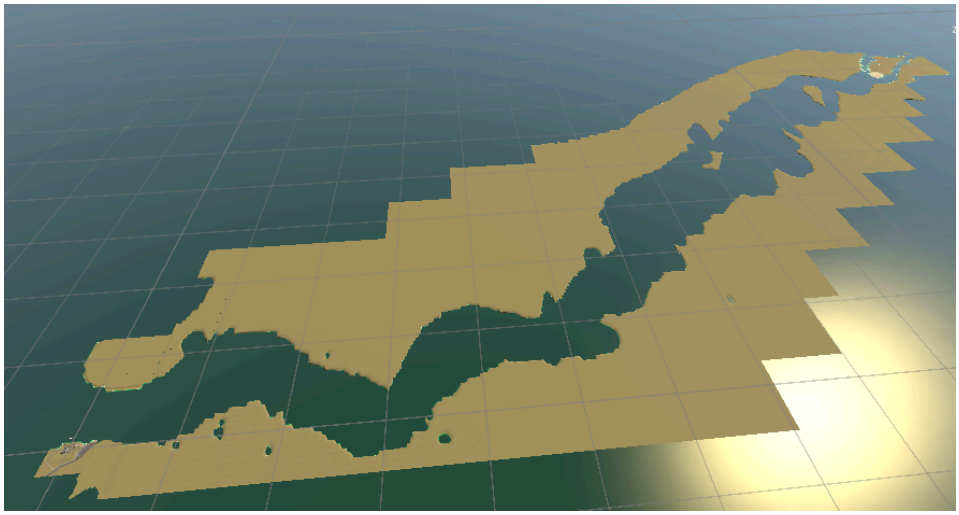
[Figure 14 River generated by height map]

Completing the River From Fort Marion to Fort Matanzas

Since the previous submission, we have successfully completed modeling the entire river path from Fort Marion to Fort Matanzas. This involved importing the Matanzas Terrain asset and aligning it with the terrain generated from our custom river height map.

When importing the height map, we initially struggled to get the two terrain pieces to line up correctly. The scales were slightly off, and the borders between the terrains didn’t match precisely. After experimenting with various solutions, we found that the most effective approach was to divide the river terrain and increase the resolution of the terrain tile being manually edited. The higher resolution allowed for more precise adjustments, and after refining the height values along the boundaries, the pieces blended smoothly.

We also made several adjustments to smooth out rough edges, level mismatched areas, and add extra terrain detail around the Fort Matanzas region to create a natural transition.



[Figure 15: Completed River generated by height map]

Next Steps: Boat Ride Length and Adjustments

Now that the full river has been modeled, our next major task is to evaluate whether the scaled river is an appropriate length for the planned boat ride experience. While we made efforts to shorten the river while keeping its shape recognizable, it is still possible that the current length may result in a ride that feels too long for users.

In the upcoming sprint, we will:

- Test a prototype version of the boat ride along the full river path.
- Measure the travel time at different boat speeds.
- Assess whether the pacing feels engaging or slow.

- Determine if any sections of the river should be shortened or removed.

If the river seems too long, we will make the necessary adjustments to maintain the overall natural feel while keeping the experience enjoyable and efficient.

Modeling the Fort Matanzas Geoplane

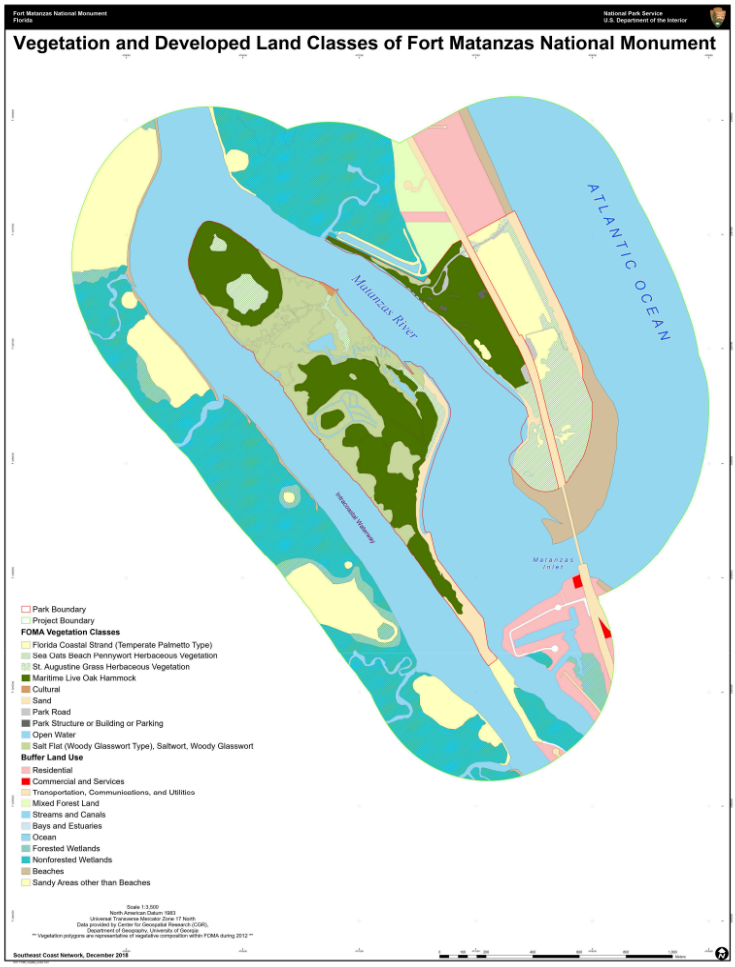
Objective

Create a walkable geoplane of the area surrounding Fort Matanzas, with ability to connect to the river between the forts. This Geoplane will be textured to match the existing assets in the project and will be close to its accurate real world scale.

What we have completed

For the terrain around fort Matanzas, we wanted to make sure we made an accurate geoplane to represent the surrounding area and river. To do this we sourced some more maps of the surrounding area, including a useful vegetation and developed land classification map from the National Park service in figure 16[2](National Parks Service). This map has been useful for understanding the general geography around fort matanzas and the distribution of trees and other plant life to reference later when we populate the island with trees. Initially we tried to use the image as a reference and sculpt the geoplane by hand, but this proved difficult for keeping scale and creating an accurate coastline.

Sculpting and all other portions of this part of the project were done with Unity's terrain tools, and GIMP. Initially, sculpting the



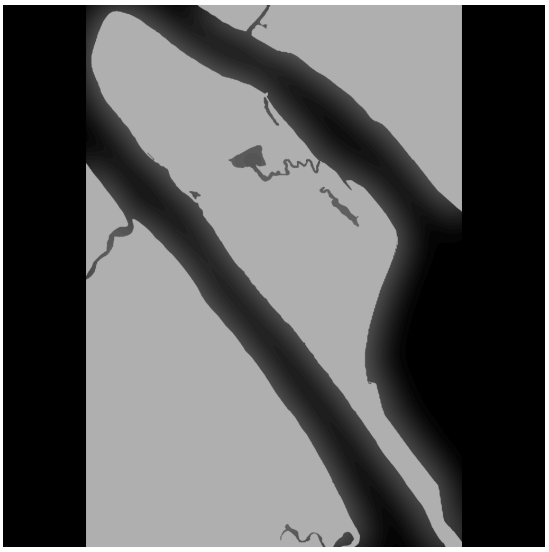
terrain involved creating a plane textured with the vegetation map, and using the set terrain height sculpt setting in terrain tools to set the height of a terrain just below this plane to just above it in certain areas, allowing us to effectively trace the map. After making a few terrain pieces, this was found to be a lot of effort to keep accurate. Instead of wasting time doing this by hand, we chose to seek out heightmaps of the region to apply to one large terrain piece that we can scale.

[Figure 16: Vegetation and Developed Terrain Map]

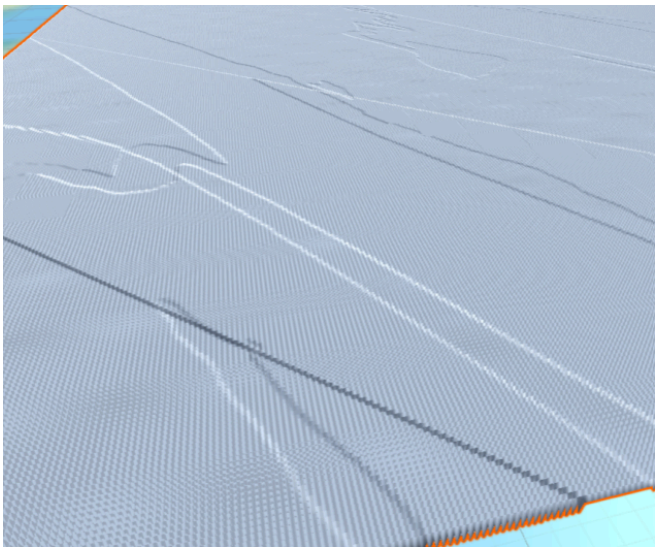
First we looked for real geographic heightmaps from actual mapping services, but it turns out that heightmaps of coastal Florida are not very useful for unity, due to minimal differences in height between sea level and the land. The heightmaps output by any program we used were also too large for our use cases, with minimum ranges of thousands of miles.

Creating a Heightmap

Our use case requires a smaller region at higher resolution with a more clear difference in height layers. In order to accomplish this, we created a heightmap in GIMP using the previously shown map as a reference. We had to research exactly how to create .raw heightmaps, export them from GIMP, and then correctly import the .raw to Unity. First we created a 1025x1025 canvas (.raw files must be in some even aspect ratio that is a power of 2 plus 1), then imported the map to use as a base for the heightmap, then we split up the image into black portions(water) and gray portions(land) and applied a blur to create a smoother slope between the layers. The resulting heightmap is shown on the left below.



[Figure 17: Heightmap]



[Figure 18: Import Error]

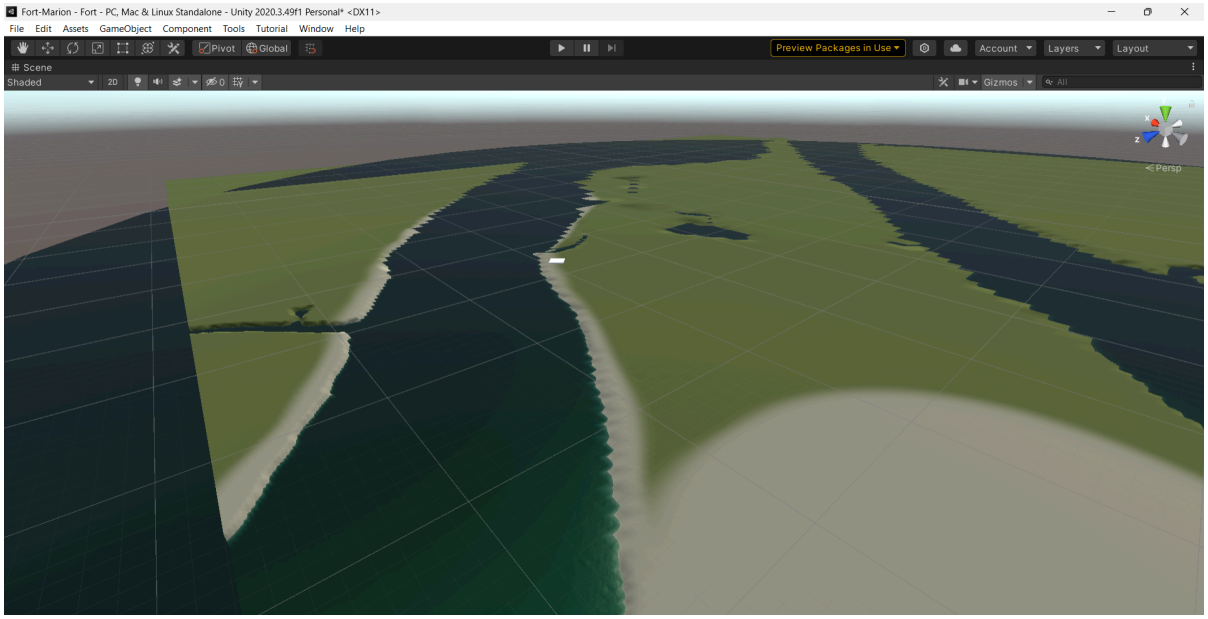
When we tested this map, we forgot to switch the color mode from RGB to grayscale, and also did not set up the correct bit depth, which resulted in a bug, where the heightmap, when imported, created jagged terrain along each pixel, that only barely represented the original image. After some confused troubleshooting and searching for related issues on unity forums,

we figured out the issue and exported the map to .raw in grayscale, with 16 bit depth, and made sure our import sizes in Unity were set to 1025 x 1025.

The resulting terrain was not to scale, but was accurate to the shape of the island. In order to scale the terrain, we used google maps to measure the distance of the northernmost point on our heightmap to the southernmost point and scaled the terrain plane to fit those dimensions in Unity's editing space. We learned that in unity, if you are making projects to any kind of scale, it is understood that 1 square on the smallest grid is 1 meter, 10 of these make up a square on a larger grid when zooming out, and so on, in magnitudes of 10 each time. Using this we scaled the island close to its accurate size, but just a little smaller for performance and user experience. It is large enough to give the user a sense of scale, but small enough to not add too much time to the boat ride down the river or cause significant performance problems.

Applying the Heightmap and Editing the terrain

After scaling the map, we then used the terrain smoothing brush to smooth out any rough features near areas the player would see up close, and applied textures for grass and sand using the texture brush. We plan to add foliage in the future to make the island feel less flat, and to block line of sight to areas the user is not meant to see. Lastly we added a short box to the scene, scaled roughly to the length and width of fort Matanzas, and placed where it would be on the island. The resulting terrain from all the previous steps is shown in the screenshot below.



[Figure 19: Finished Terrain]

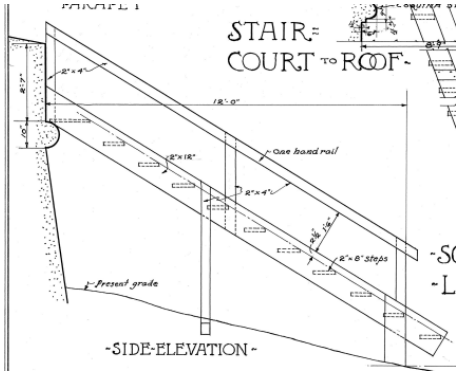
Enough has been done on this portion for it to qualify as done for this sprint, but new tasks will be created to make it look better and make it easier to implement other parts of the project. A task for populating the island with trees, and 2D background planes to block line of sight to some portions of the scene will be created, and a small subtask will be added, which will be sinking some space in the terrain to put the lower levels of the fort in without visible geometry clipping issues.

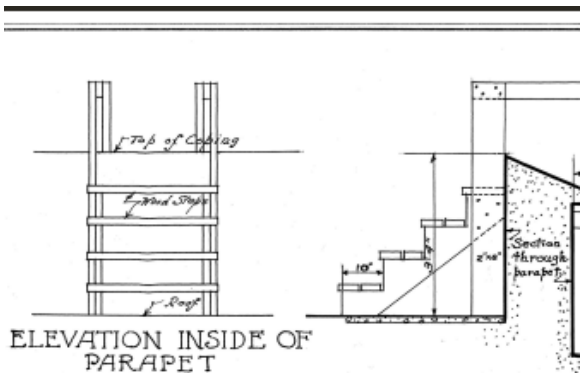
Stairs and Ladders

Overview

Fort Matanzas contains multiple sets of stairs and ladders that play an important role in both navigation and historical accuracy. These structures include the entrance stairs, parapet stairs, court-to-roof ladder, and ladder

leading to the court. Each of these elements must be carefully modeled to ensure that they accurately represent the real-life architecture of the fort. The sponsor provided us with detailed architectural plans and measurements, which serve as our primary reference for scaling and proportion. Each model must align precisely with these dimensions to



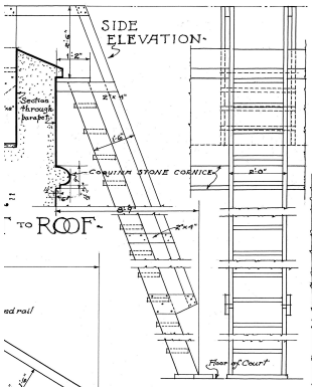


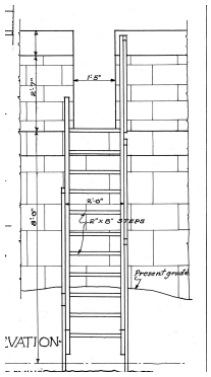
maintain the authenticity of the environment.

[Figure 20 Entrance Stairs] [Figure 21 Parapet Stairs]

Modeling Process

We will model each stairway and ladder according to the provided blueprints. This process involves studying the sponsor’s plans to understand the relative





[Figure 22 Court to Roof Ladder] [Figure 23 Ladder leading to the court]

height, width, and angle of each structure. Using these references, we will create the models in Blender.

To maintain consistency, we will use a uniform scale across all models and verify measurements by comparing the models to reference objects within the Unity environment. This ensures that when the user navigates through the fort, each structure feels proportionally correct and realistic relative to the surrounding environment.

Integration

Once the stair and ladder models are completed, we will integrate them into the Fort Matanzas environment and test their functionality. The primary goal is to make sure that the stairs are walkable and the ladders are climbable. For stairs, we will confirm that the player's character can smoothly ascend and descend each step without clipping or sliding. We may adjust the colliders or slope angles to create natural movement that feels similar to walking on real stairs.

For the stairs, we will test how the player's movement system responds when walking over steps and make adjustments as needed to achieve natural transitions between flat ground and elevated areas.

For the ladders, we plan to explore several approaches for user interaction. One possibility is to allow players to physically mimic a climbing motion

using their VR controllers, while another option could involve a simple interaction that automatically moves the player up or down the ladder.

Modeling the Fort - Tower

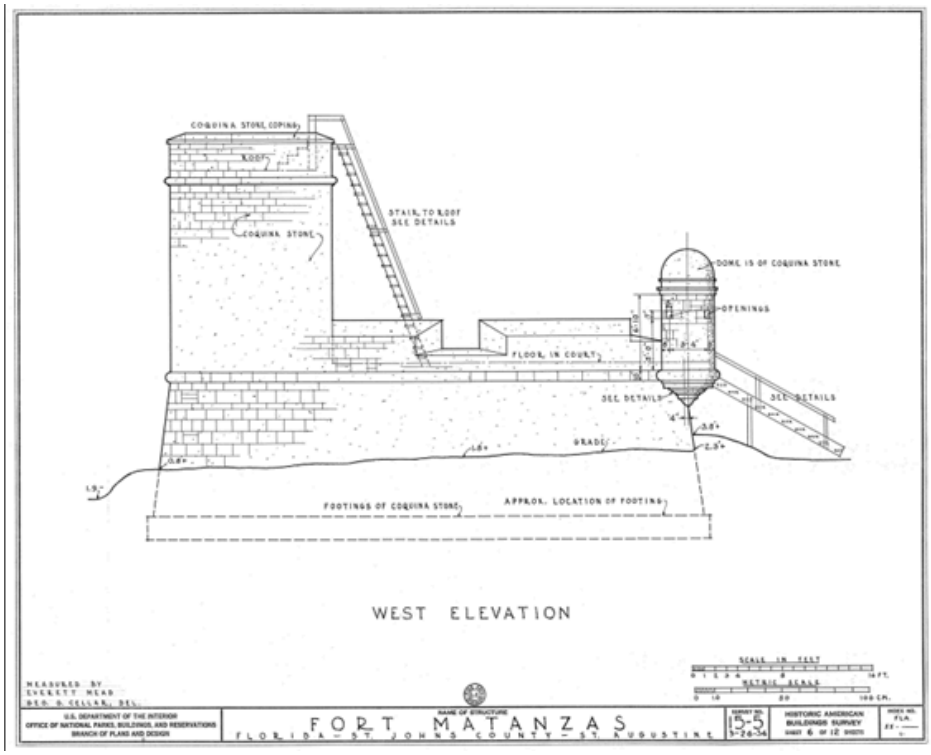
Objective

The task of modeling the fort was split into multiple parts so we could have enough people resources to ensure the details were accurate and to scale. The tower on Fort Matanzas is a standout feature, so it received its own task. Its geometry was noticeably different from the main fort body as it was cylindrical and featured a dome shape at its top and bottom.

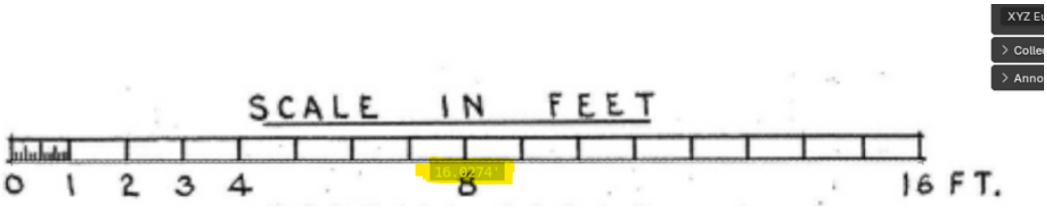
[Figure 24 Exterior view of the tower]

Research

We used Blender to model the fort's tower as it was free, open source and tutorials were widely available. There are many detailed photos of the tower specifically on the internet as its unique nature means visitors pay special attention to it. Historical architectural drawings were also provided to the team which aided in getting the historical details right. By using the **Measure** tool built into Blender, we were able to match the scale on the drawings to Blender's internal scale. These plans along with photographs on the internet were extremely useful to the modeling process.



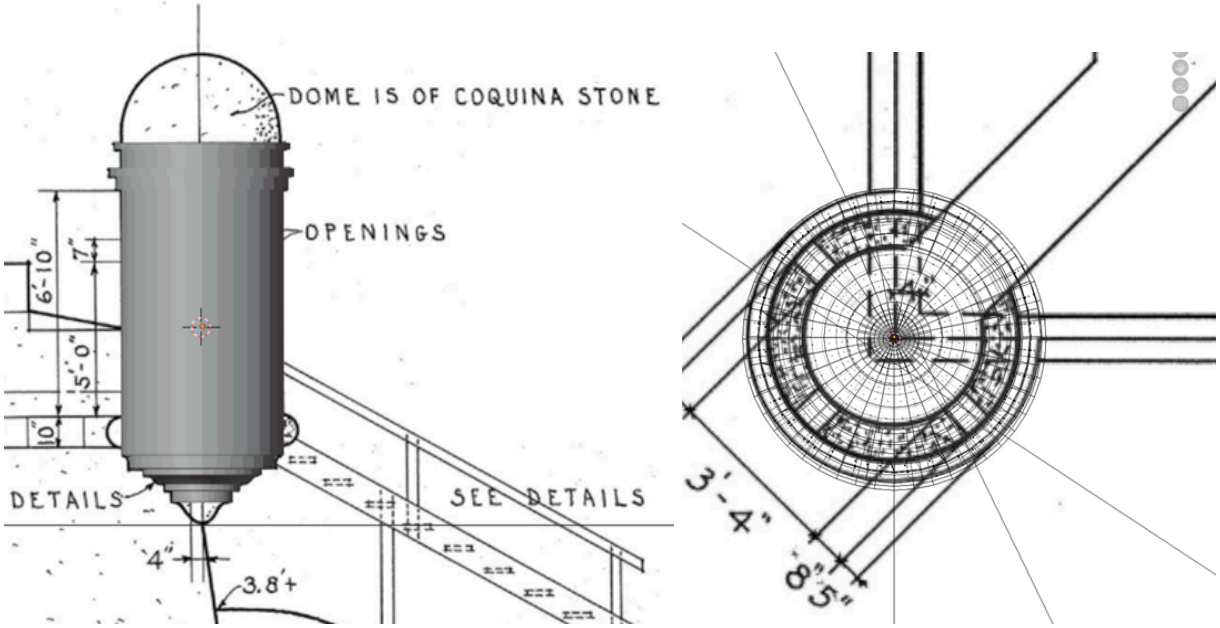
[Figure 25 Side view of the fort from historic plans.]



[Figure 26 Blender’s Measure tool aligned with the plan’s scale with the tool’s measurement highlighted.]

Modeling the Tower

The modeling process began with a cylinder mesh in Blender to establish the main body of the tower. The height and radius of the cylinder were scaled precisely according to the architectural reference plans. Once the base proportions were correct, edge loops were inserted along the vertical axis to allow for smooth curvature adjustments. The upper and lower domes of the tower were then modeled using sphere primitives, which were scaled and positioned to blend seamlessly with the top and bottom of the cylindrical body. The lower dome was slightly flattened to better match the fort’s original construction, while the top dome maintained a more rounded appearance consistent with historical photographs.



[Figure 27 Partial model of tower with architectural plan as reference(side and top views respectively).]

To replicate the stone ledge at the base of the tower, a torus mesh was added and positioned around the lower portion of the structure. The torus was scaled non-uniformly in the Z-axis to give it a flatter profile, accurately representing the decorative and structural ledge visible in reference images. With the overall shape completed, Boolean modifiers were used to create openings for doors and windows. Each opening began as a separate cube mesh that was positioned and scaled according to the tower's reference drawings. Boolean difference operations were then applied to subtract these shapes cleanly from the tower's mesh. After the Boolean cuts, excess geometry was cleaned up by merging vertices and removing non-manifold

edges. Finally, edge connections and smoothing operations were performed to ensure consistent shading and curvature across the model's surface.

[Figure 28 Render of finished but untextured model of the tower with a 5'7" human for scale.]

[Figure 29 Render of finished but untextured model showing the doorway and one of the window cutouts.]

Next Steps

The next stage in the process involves UV mapping and texturing. Once the geometry is finalized, UV seams will be strategically placed along natural boundaries—such as the base of the ledge, dome transitions, and window frames—to reduce stretching and visible seams. The model will then be unwrapped to create a clean, organized UV layout suitable for detailed texturing. We plan to apply PBR (Physically Based Rendering) materials that accurately simulate coquina stone, the primary building material of Fort Matanzas, giving the tower a sense of realism and time accurate appearance.

After texturing, the tower model will be integrated with the main fort structure. Once imported into Unity, the material properties and lightmaps will be adjusted to optimize performance for the VR environment, ensuring the target framerate of 90 fps is maintained. The completed tower will serve as a central visual and interactive element within the digital reconstruction, showcasing both historical accuracy and immersion in the final VR experience.

Modeling the Fort - Second Floor

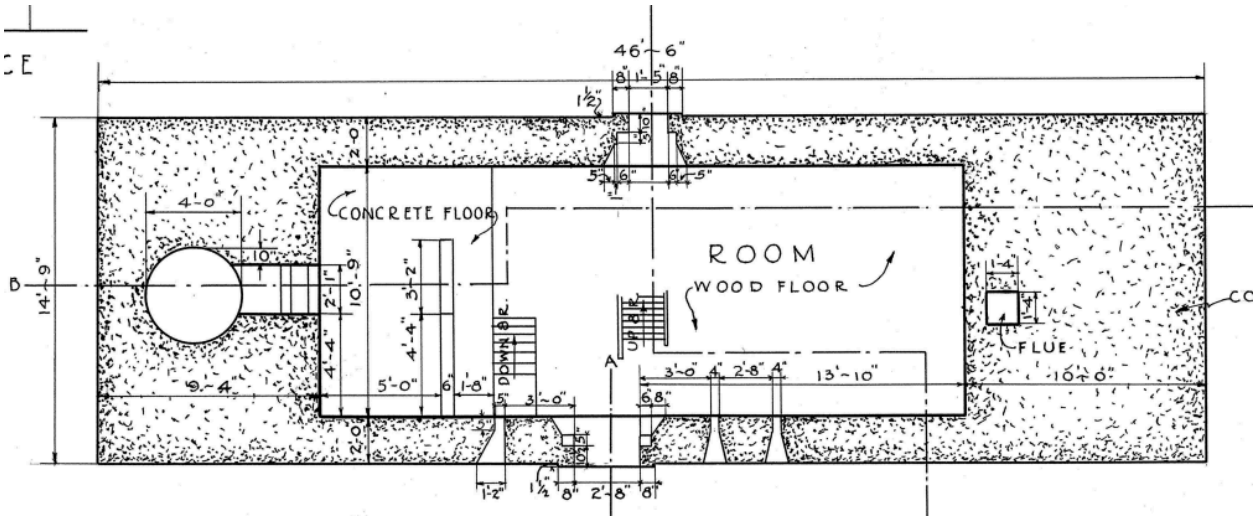
Objective

The process of modeling the main body of Fort Matanzas was divided into two focus points: the first floor and the second floor. The two floors have their own subtasks of modeling the interiors and exteriors. In our second sprint, one of our members focused on modeling the second floor of the fort. This involved creating a 3D model of the second floor and roof along with detailing the doors, windows, and other intricacies.

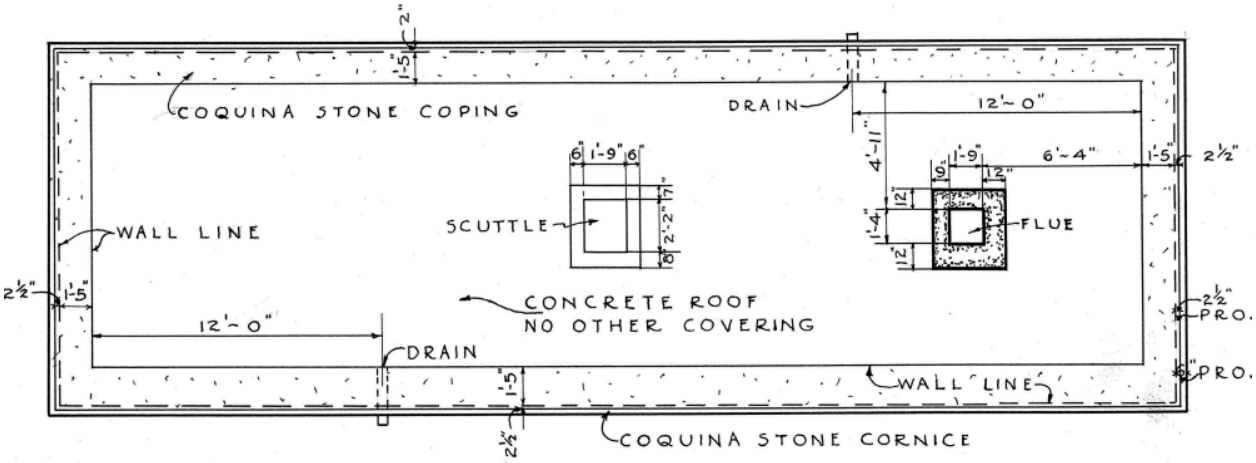
[Figure 30 Reference Image of Second Floor Interior]

Research

As with the tower of Fort Matanzas, we opted to use Blender for the same reasons. We were also provided with the full plans and measurements of the fort from a survey in the 1930s. Having the exact measurements and angles from the architectural drawings made possible modeling the fort in a precise manner. After careful analysis of the architectural drawings in Blender, we found that we were unable to model the fort by superimposing meshes onto a screenshot of the diagrams. This was because the architectural drawings did not always have straight lines or were slightly inaccurate to their measurement scales. Instead, we decided to use the drawings only as references as we used the numerical measurements to maintain precision. Our sponsor was also helpful for modeling the Fort in regards to her knowledge about the Fort and her active correspondence in Discord.



[Figure 31 Floor Plan of the Second Floor]



[Figure 32 Roof Floor Plan]

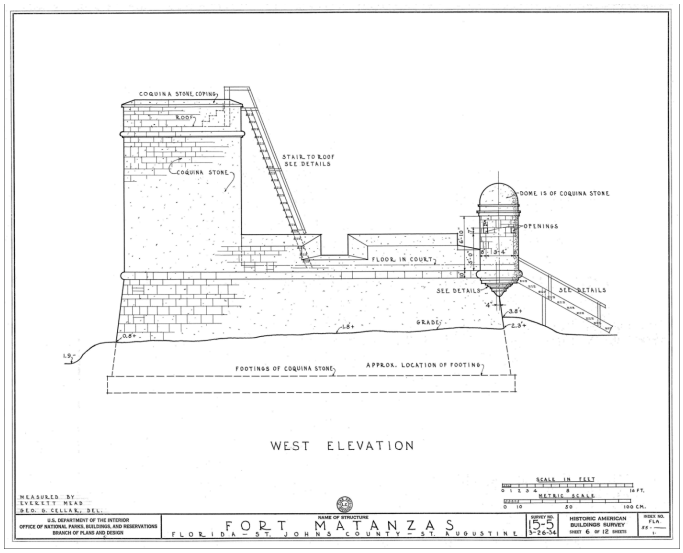
Beyond the provided architectural drawings, we explored the internet for further research. An important point of reference came from the National Park Service's website on Fort Matanzas. The website's Virtual Tour provided

a 360 degree view of certain points around the Fort. This gave us valuable references to parts of the second floor and roof that weren’t commonly photographed. The National Park Service’s website also had a three-dimensional scan of Fort Matanzas made by the University of South Florida’s Libraries. This 3D model was helpful in verifying the measurements and proportions of our modeling in Blender.

Modeling

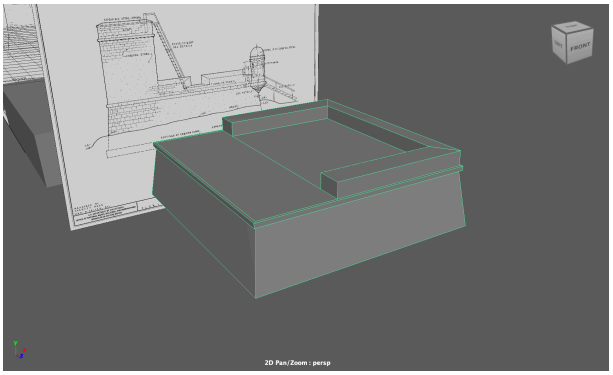
Modeling the Fort - First Floor

To model the first floor of Fort Matanzas, we used a side view reference from the west elevation of the fort. This allowed us to get the basic shape of the model down. Starting with a cube, the shape was modeled into a trapezoidal figure to resemble the base of the fort.



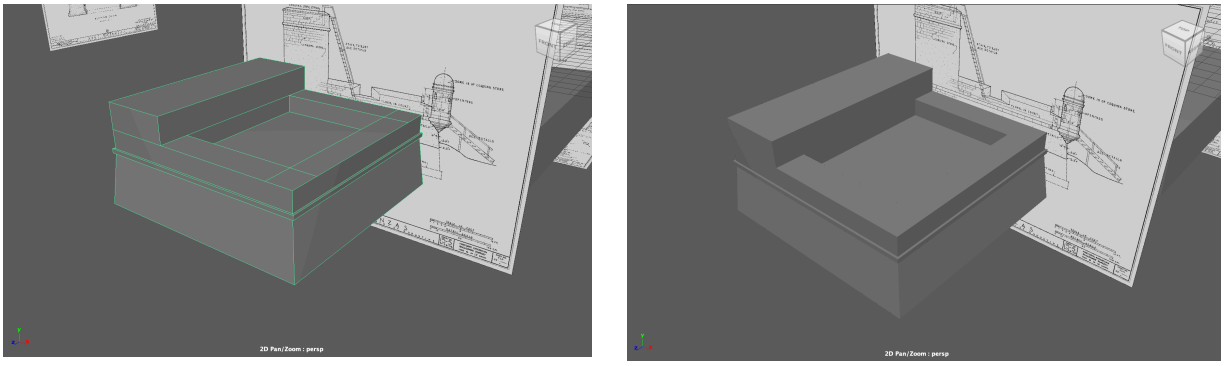
[Figure 33 The primary view of the fort used to model the first floor]

As seen from the reference image, there is a thin segment in between the base of the fort and the top deck. In order to replicate that, the top of the trapezoid was extruded slightly upwards. The top was extruded once again in order to start the deck.



[Figure 34 The initial appearance of the first floor]

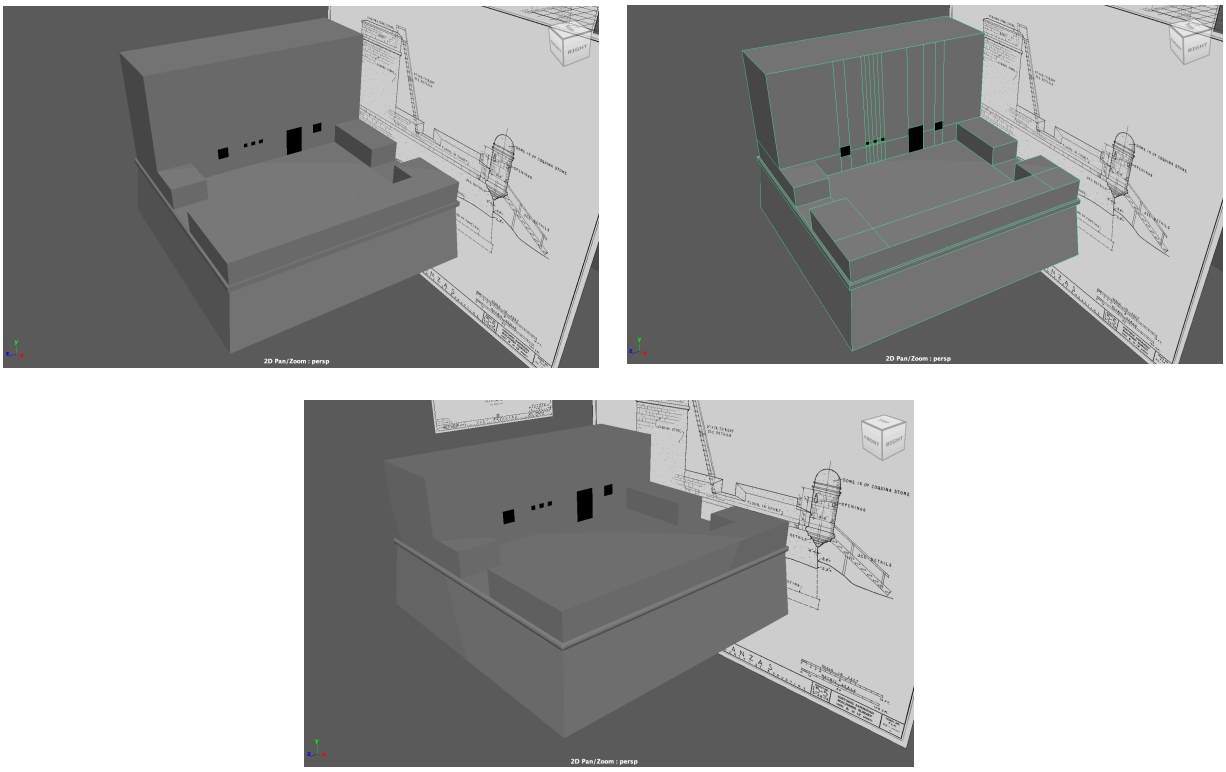
Recreating the deck in Maya caused some difficulties. The extrusions caused some faces to overlap others, which resulted in bad geometry. As a result, the model was redone a few times until a solid start to the model was reached.



[Figure 35 Redoing model]

After some trial and error, the model reached a point of satisfaction for the developers to continue.

Using the multi-cut tool in Maya, we created edges on the faces of the model and extruded the newly-created faces outwards. As seen in Figure 35, there is one side that is higher than the others. This side will contain the room of the fort, while the other sides are the balcony of the first floor.



[Figure 36 Finishing touches]

The next step was to carve out the various perforations in the fort. These include parts of the balcony, the doorframe of the body, and the windowframes. In order to maintain good geometry, the multi-cut tool was used once again, and edges were drawn across the body of the fort, as seen in Figure 36. Then, the faces were deleted.

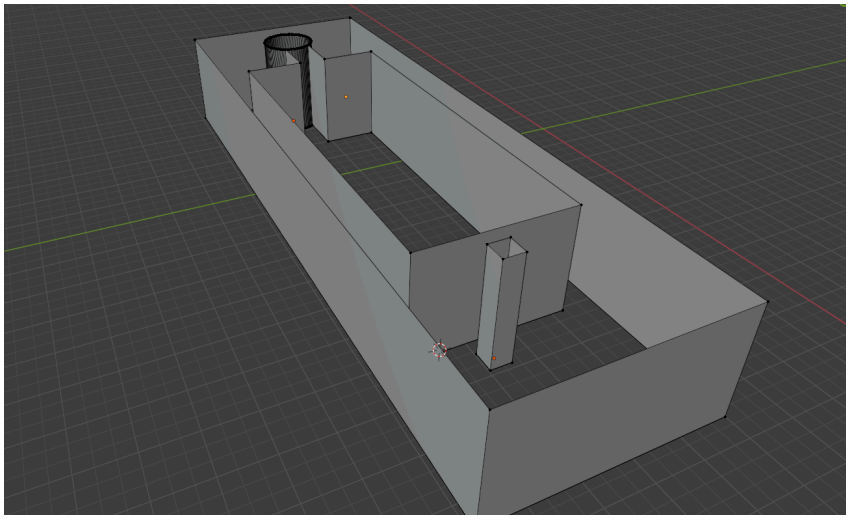
The segment in between the first floor and the trapezoidal section was also rounded out using the bevel tool in order to resemble what was provided with the reference images.

Next Steps

The next step for the first floor is to model the inside of the fort, where the door leads to. In order to do this, faces on the inside walls will be extruded inwards and reversed, and further touches will be made.

Afterwards, we will have to connect the second floor to the top of the first floor, add the ladders and dome, and bevel the edges in order to mitigate the sharp transitions that the edges currently have.

Modeling the Fort - Second Floor

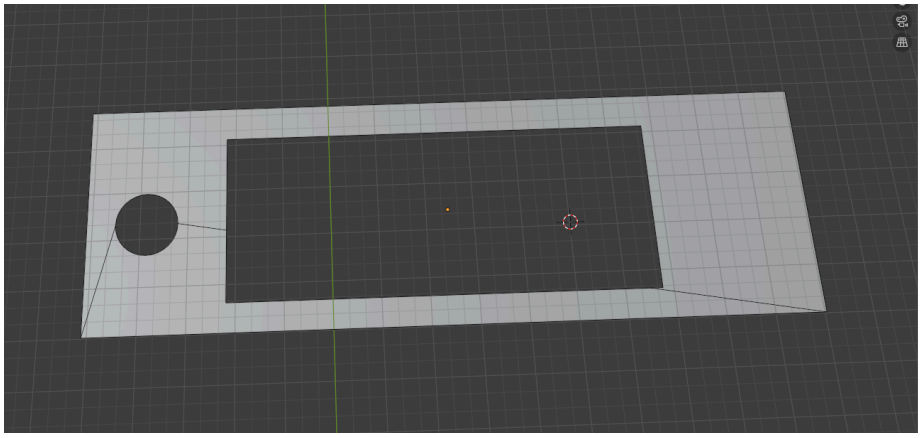


[Figure 37 First attempt at the Second Floor]

In our first attempt at modeling the second floor in Blender, we outlined the interior walls, exterior walls, and flue using vertices and edges

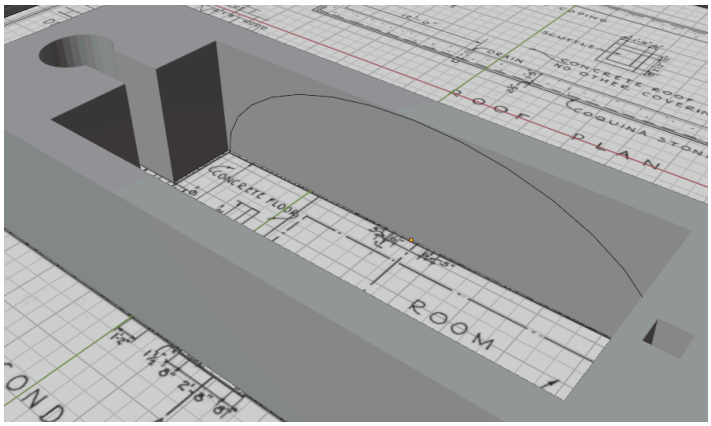
following an online tutorial. We started by taking a screenshot of the floor plan and importing the image into a Blender project. Then, we added a 2D plane mesh and scaled it to the dimensions of the exterior walls. We deleted the face of the plane to obtain its vertices and edges. The same was done for the interior walls and flue. However, the floor plan of the interior wall had a slightly more complex shape than a rectangle or a square. We had subdivided one of the edges of the interior wall and connected it to a circle mesh. Finally, we took all the edges and extruded them to the height that was bound by the ceiling which is shown in Figure TL.8. The issue that arose from the first iteration of the second floor was filling the space between the interior and exterior walls. Due to the complex shape of the interior walls and the inclusion of the flue, it was impossible to use Blender's Solidify modifier to add thickness to the walls. We were unable to come up with a solution to this issue, so we were forced to abandon the online tutorial in favor of an alternative method.

In our next attempt at modeling the second floor, we decided to keep the face of the 2D plane to have the space between the interior and exterior walls already filled.



[Figure 38 Second Attempt using Boolean Differences]

We used the circle and 2D planes provided by Blender to cut out the interior space and flue. This was made possible by generating a Boolean modifier on the main 2D plane and setting a Boolean Difference between it and the desired shape to cut out. After extruding out the walls of the second floor to the measured height, we modeled the ceiling. The second floor's ceiling was irregular as it was not a simple reflection of the floor and flue. We noticed that the ceiling was in the shape of an arch that spanned the rectangular portion of the interior. We accomplished this part by first importing a circle and deleting the lower half of its vertices. Then, we stretched it on the X-axis to cover both sides of the interior walls. Finally, we extruded it on the Y-axis to cover the perpendicular walls.



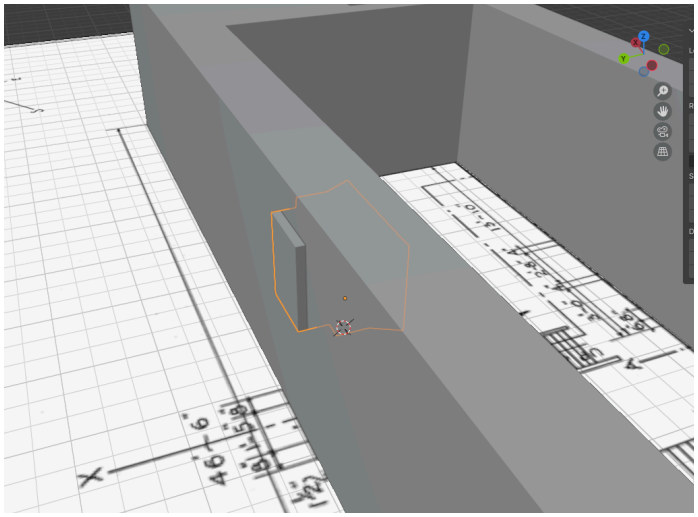
[Figure 39 Arch ceiling before extrusion]

Because the ceiling was a one-dimensional half-circle extruded into a two-dimensional arch, it led to an issue with using Boolean Difference later

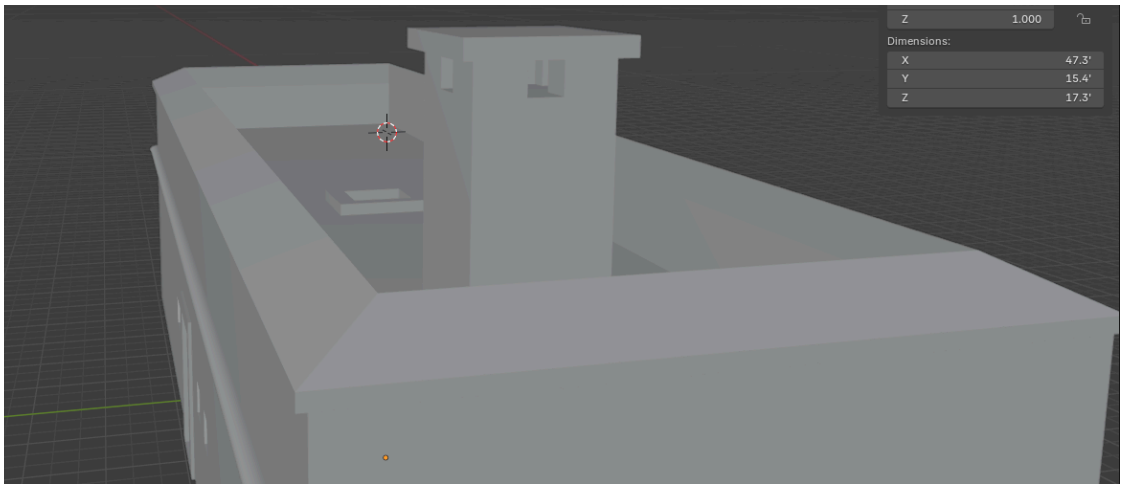
on. When we tried to create spaces for the ladder going to the roof and the hatch for the circular area on the side using cube meshes, the Boolean Difference modifier did not act as expected. It seemed as if the modifier was taking the difference of one side and the union of the other. After trying to recalculate the normals, we found that the inside and outside faces were the same because the ceiling was a two-dimensional object. The solution to this problem was to use the Solidify modifier to add depth to the ceiling. This allowed us to set the normals properly on the mesh and use the Boolean Difference modifier.

For the windows and door of the second floor, we first created 3D meshes that represented the windows and door as if they were filled in completely. This contrasted with our previous method of using simple meshes to carve out simple holes in the floorplan. With our complex windows and door meshes made with the measurements from the floor plan, we again utilized the Boolean Difference modifier.

[Figure 40 Rear Window Mesh before Boolean Difference]



Once the windows and door were completed, we moved on to model the roof and its details. We decided to model the floor and walls of the roof separately because of the relatively large space between the ceiling of the second floor and the floor of the roof. Another reason for this was due to the roof having a slanted top with a slight overhang. This contrasted with the straight wall of the second floor. For the floor of the roof, we took a cube and expanded it to match the measurements detailed in the architectural plans for the side of the fort. We took the same rectangular prism as the one used to cut out a hole for the ladder in the ceiling as used in a Boolean Difference with the roof floor. The rectangular prism was used again to create the slight extension for the ladder as seen in the figure below. Finally, we used a Boolean Union to add the flue's chimney onto the roof's floor. The chimney is composed of a rectangular prism along with a cube resized into the chimney top. The holes for the chimney were created using Boolean Differences using a custom object similar to one used in the windows.



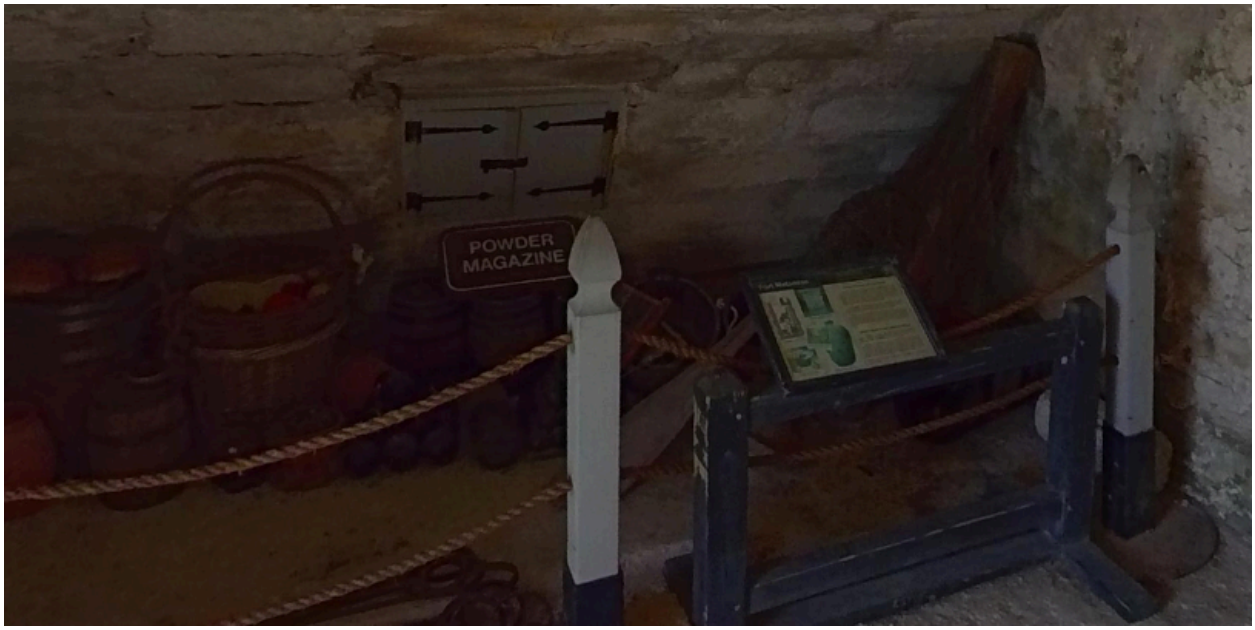
[Figure 41 Completed Roof]

As for the roof walls, we extruded up to the specified height in the architectural documents, stopping before the sloped portion. When we modeled the sloped portion of the roof wall, we started with the slight overhang. Then, we selected the edges of the overhang and extruded them up above the interior of the roof wall. After finishing the roof, we assembled the roof floor, roof walls, second floor, and ceiling using Boolean Unions.

Next Steps

After everything that has been completed so far, the next step for this part of the fort is to create the UV mappings and textures. The UV mappings and textures will allow for us to add details to the plain three-dimensional mesh such as coquina rock and other materials found in Fort Matanzas. We also need to assemble the models of the first floor, second floor, and tower. As one of our members created the first floor in Maya, we opted to export our files in the FBX format. The FBX format allows for exchanging three-dimensional data between Blender and Maya.

To create an accurate representation of the second floor, we need to add furnishings to the interior. This includes a bed, a desk, and some other miscellaneous objects. The Powder Magazine also needs a hatch as shown in the figure.



[Figure 42 Furnishings to be added]

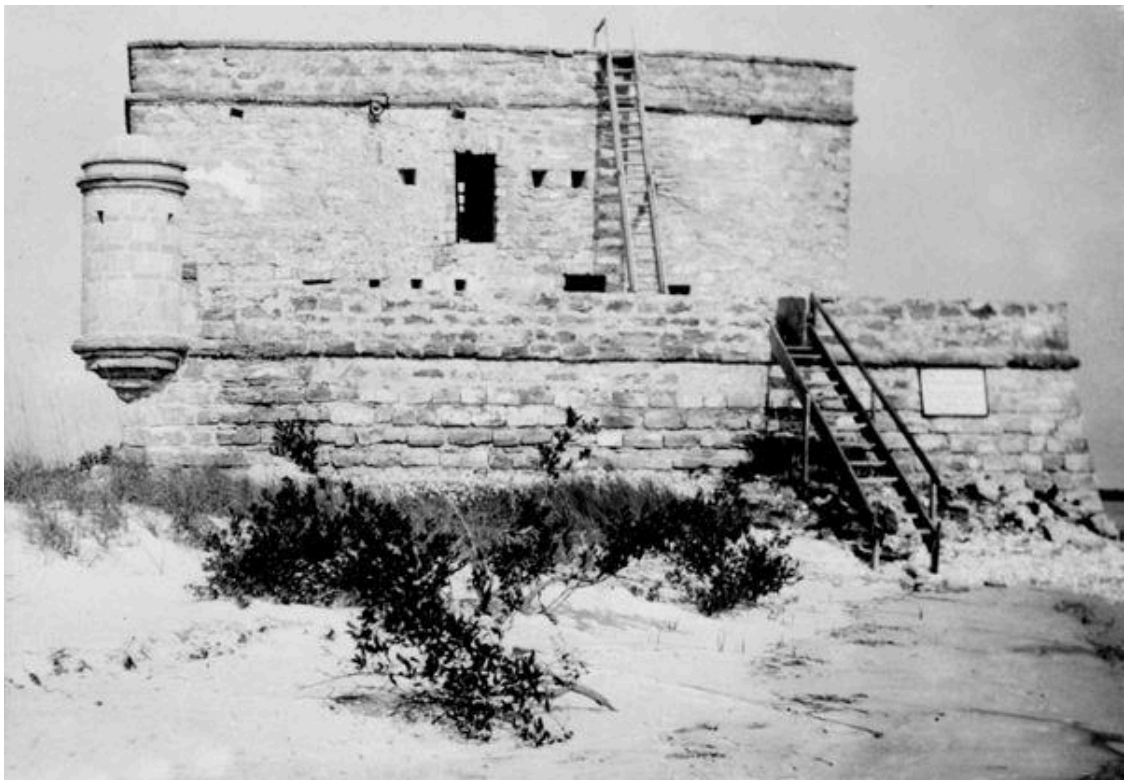
Modeling the Fort - Ladders and Stairs

Objective

This task involved modeling the three ladders and two staircases of the fort. We relied on historical architectural plans and archival photographs provided by our sponsor, supplemented by a reference image found during our research. These resources enabled us to reconstruct the structures as they existed historically, prior to modifications made to satisfy modern visitor safety standards. The resulting models provide an accurate recreation for use in our game.

Research

We modeled the ladders and stairs in Blender. It made sense to use it since it’s free, open-source, and already works well with the rest of our project tools. It also helped that there are plenty of online tutorials to reference while working. To keep the scale correct, we brought in the architectural plans and used Blender’s measuring tools to match the real-world dimensions, just like we did for previous parts of the fort. The archival images were used for reference while modeling.

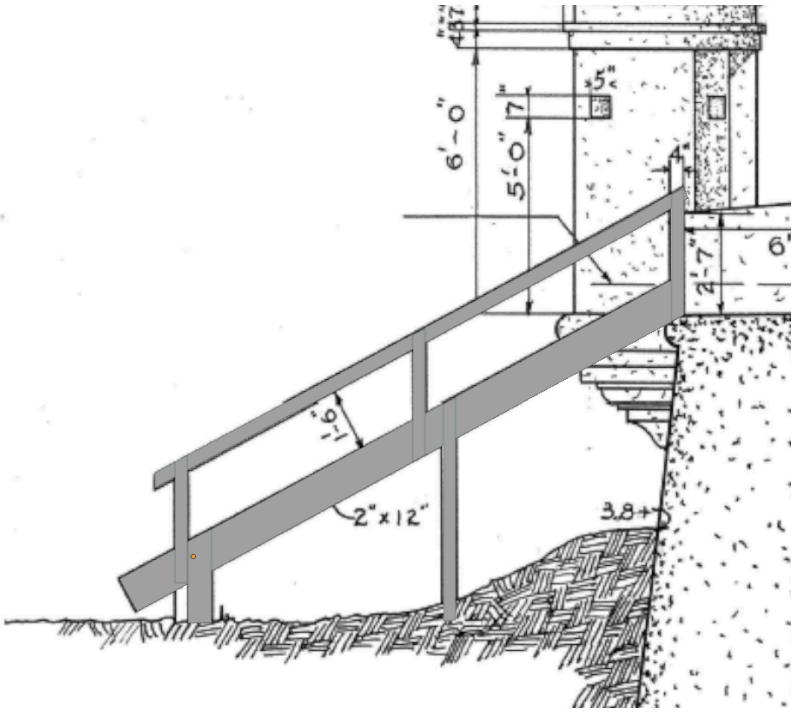


[Figure 43 Historical photograph showing the original outer ladder and stairs of Fort Matanzas]

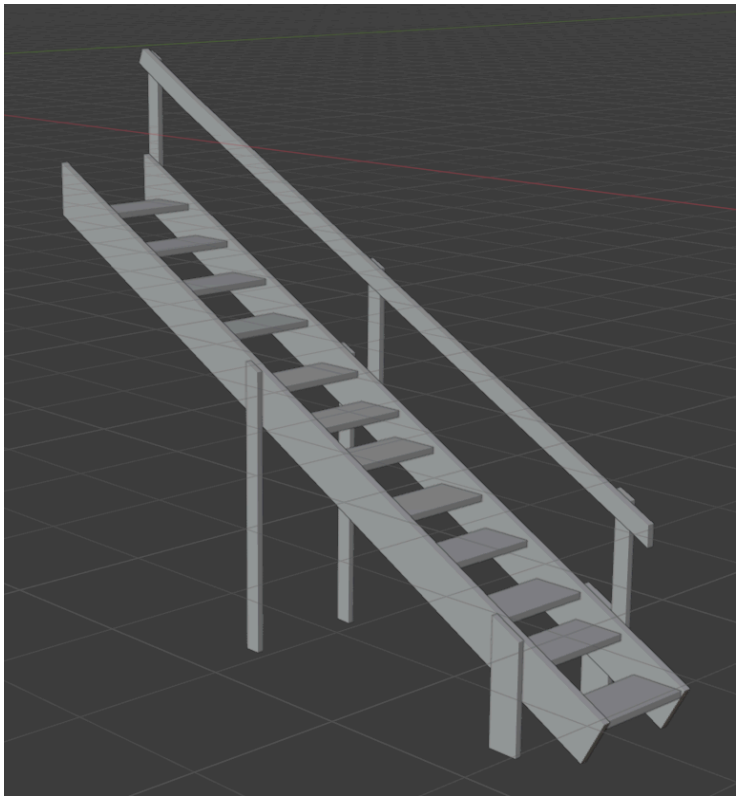
[Figure 44 Sponsor provided photos showing internal ladder placement(left) and ladder details (right)]

Modeling

Compared to the other parts of the fort, the ladders and stairs were much more straightforward to model. We followed the architectural plans closely and used simple geometry in Blender. Each piece started from a basic cube that we scaled to the correct dimensions. For the stairs leading to the courtyard, the first stringer was made by extending a rectangular block at an angle, then duplicating it to form the opposite side. The handrail and support beams were made in a similar manner. A single step was modeled, duplicated, and positioned along the stringers to match the spacing shown in the drawings.

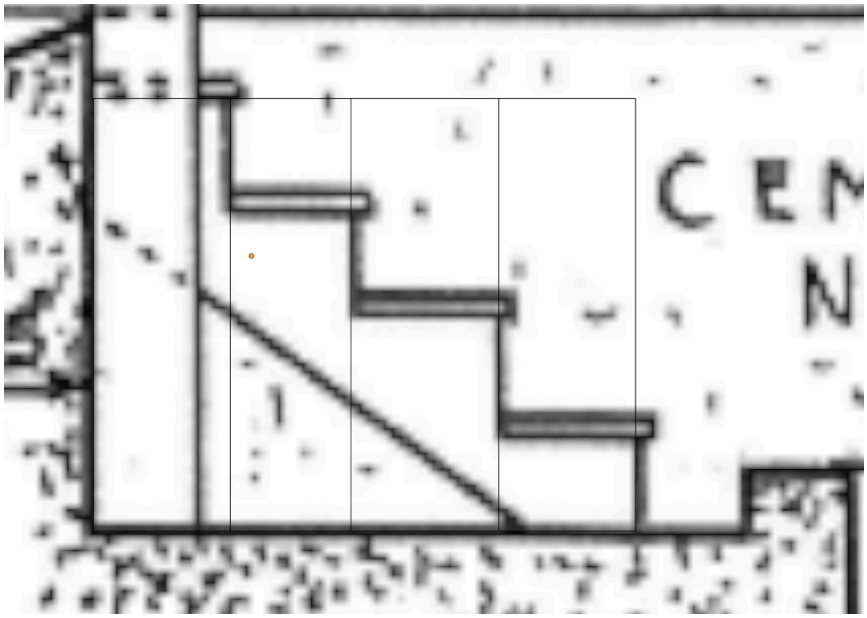


[Figure 45 Stairs leading to courtyard modeled according to architectural plans]

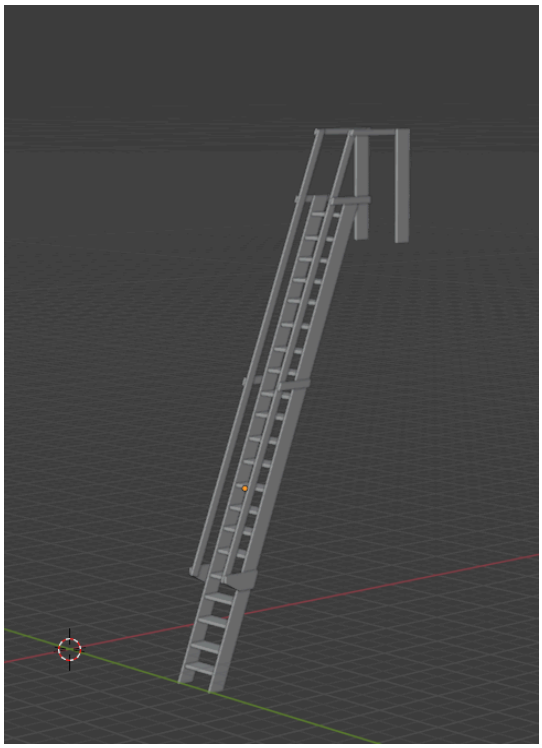


[Figure 46 Alternate view of stairs leading to courtyard modeled according to architectural plans]

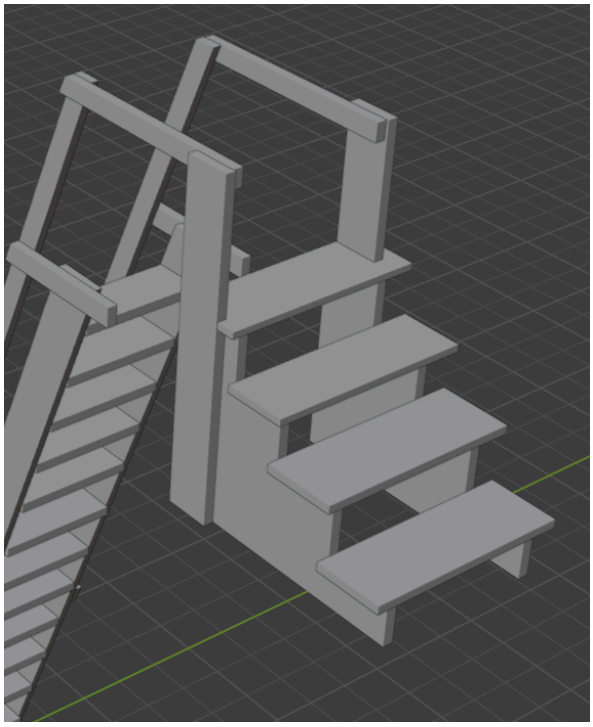
Next, we modeled the ladder that leads up to the parapet. This section is a bit unusual because it switches from a ladder to stairs once it reaches the top of the fort. We started by creating the side rails, duplicating and spacing them based on the measurements in the plans. The rungs were built from small extruded rectangles and evenly placed along the rails to match the layout in the reference images. After the ladder was finished, we moved on to the short stair section leading onto the parapet. The stringer was made from a rectangular mesh with loop cuts added so it could follow the drawing, and the steps were then duplicated and placed just like the main staircase.



[Figure 47 Cube Mesh with loop cuts matching the stringer of the stairs leading to parapet]



[Figure 48 Outside ladder leading to roof of the fort]



[Figure 49 Parapet stairs that are connected to the outside ladder that leads to roof]

Next Steps

Before these models can be fully integrated into the game, they will need to be UV unwrapped and exported to Unity for texturing. The ladders and staircases will also be combined with the existing tower and floor models to complete the entire structure of the fort. In addition, two interior ladders still need to be modeled, which will follow the same workflow once the remaining reference measurements are finalized.

Constructing the Rowboat Model

This phase of the project focused on 3D modeling using Autodesk Maya. After setting up the software, the modeling environment was reviewed, and the interface, tools, and controls were re-familiarized. While navigation and shortcut commands were quickly recalled, modeling techniques such as maintaining clean geometry, correct edge flow, and balanced proportions required additional practice.

A rowboat model was selected as one of the initial development tasks. This object served both as an essential asset for the project and as an opportunity to reestablish modeling proficiency. The model provided a controlled yet meaningful introduction to the workflow, serving as preparation for more complex structures planned for later stages.

A core component of the project involves implementing a boat ride feature to allow travel between two in-game locations. A rowboat asset was therefore required. Rather than relying on a premade online model, the decision was made to design and construct a custom version within Maya.

Creating the model in-house provided several advantages:

- Ensured stylistic consistency with other project assets.
- Allowed full control over polygon density, proportions, and UV layout.
- Offered valuable technical practice for larger assets planned for future sprints, such as the fort structure.

Despite this, there were some drawbacks:

- Technical difficulties, such as software crashes, were experienced multiple times and caused some progress to be lost
- Posed challenges finding a balance between designing a rowboat from online reference images while also preserving originality

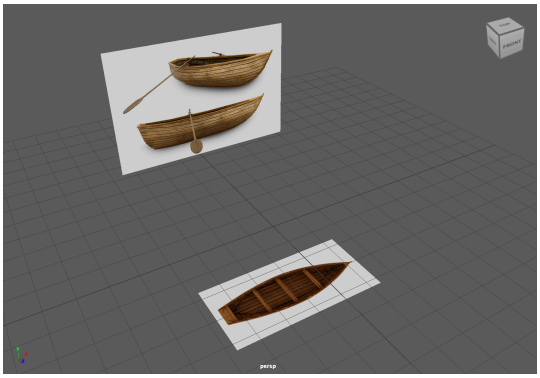
The rowboat modeling task served as both a training exercise and an initial production deliverable that contributed directly to the overall gameplay environment.

Modeling Process

Multiple reference images of real rowboats were collected from top, side, and front perspectives. These images were imported into Maya as image planes aligned to the XY and XZ axes. The use of orthographic reference planes allowed accurate control of proportions and curvature during the modeling process.

[Figure 50 Reference images for the rowboat]

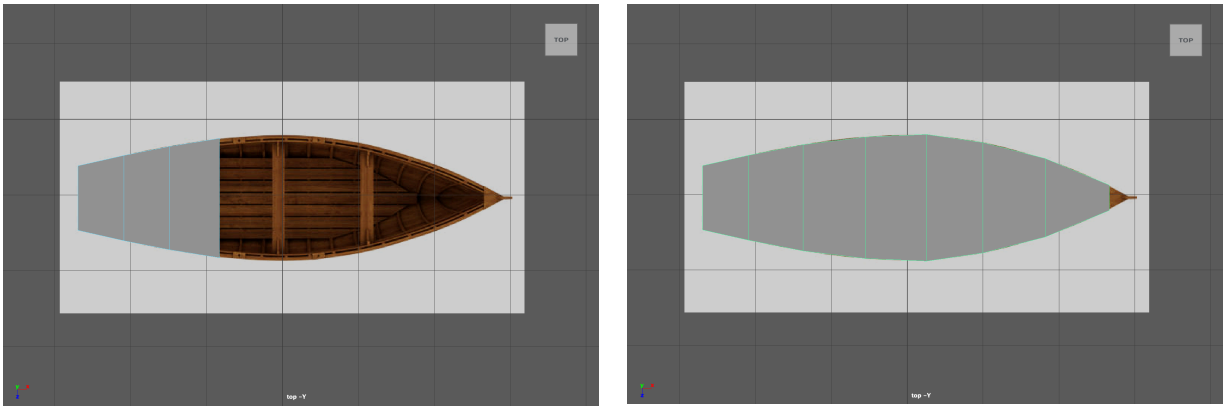
This setup established a foundation for the geometric accuracy and realism of the model while maintaining compatibility with the project’s stylized visual direction.



[Figure 51 The image planes set up in the 3D modeling environment]

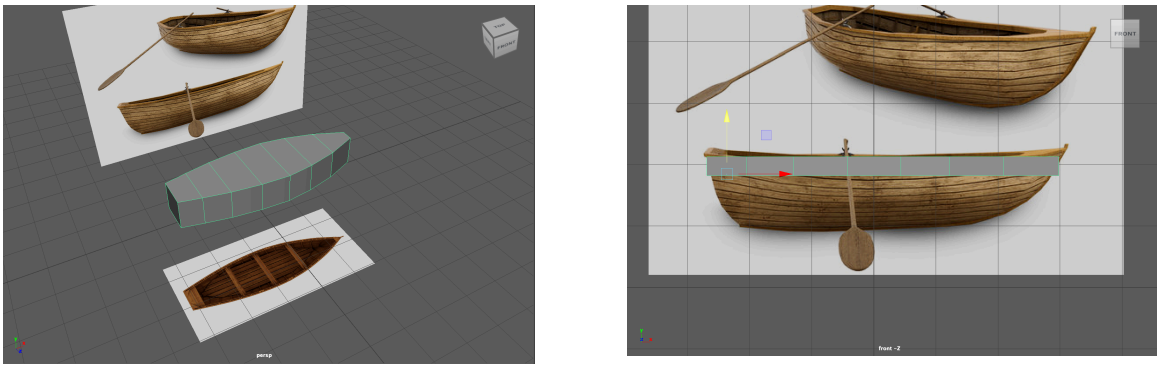
Base Shape Construction

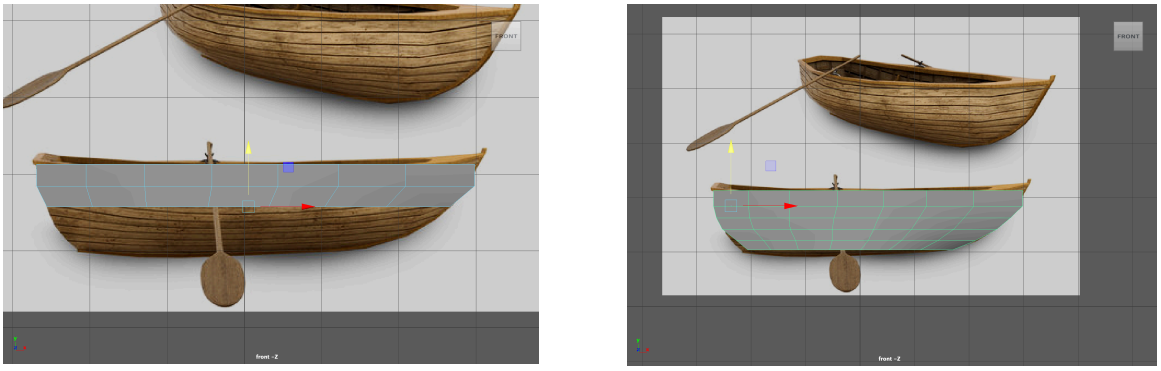
The modeling process began with a cube primitive as the base object. Using the top-view reference image, one of the cube’s faces was extruded repeatedly to form the general outline of the boat’s hull. This extrusion-based technique maintained clean edge flow and allowed controlled refinement of the shape without introducing unnecessary complexity.



[Figure 52 The process of starting the rowboat model]

Once the top silhouette was formed, the hull was extended downward from the side view to define the curvature and depth of the body. The combination of these steps established the fundamental proportions of the model.





[Figure 53 The general shape of the rowboat was beginning to come together.]

Iteration and Refinement

Clean geometry and efficient topology were prioritized throughout development. Early versions of the model revealed issues with uneven edge distribution and undesirable topology patterns. To address these problems, the rowboat was restarted twice, resulting in significant improvements to mesh organization and smoothness.

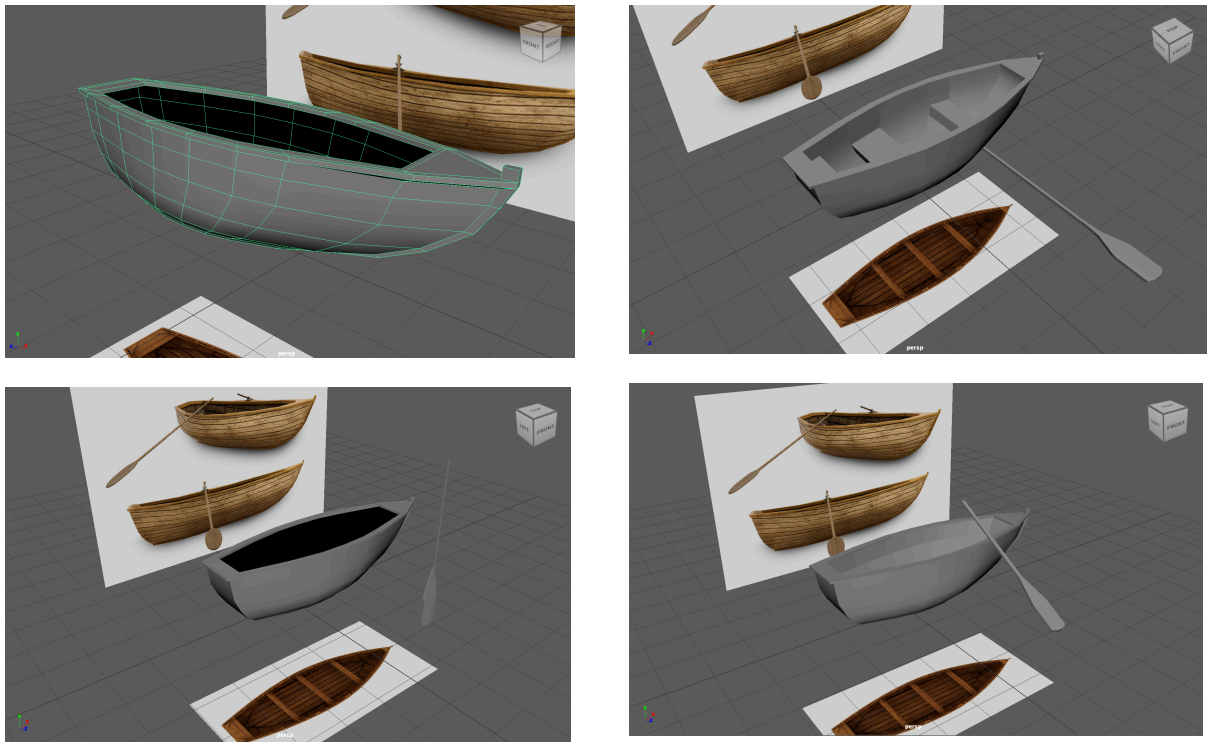
The final iteration demonstrated:

- Quad-based topology with consistent edge loops.
- Symmetry maintained through the use of mirror and alignment tools.
- Appropriate curvature verified using smooth preview mode.

This process emphasized structured iteration and problem-solving in the context of technical modeling.

Interior Detailing

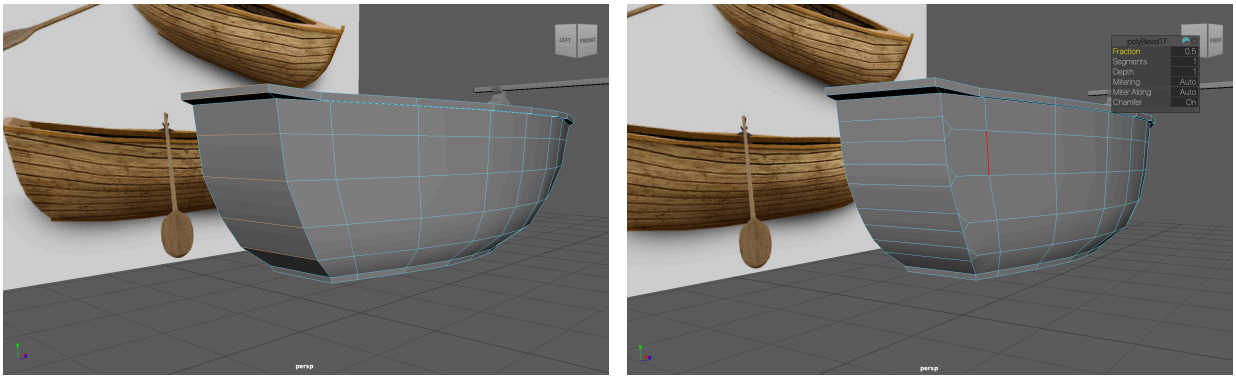
Following completion of the exterior hull, work progressed to the interior components of the rowboat. This stage involved modeling inward extrusions to define wall thickness and internal curvature, along with additional details such as seating planks and structural supports.



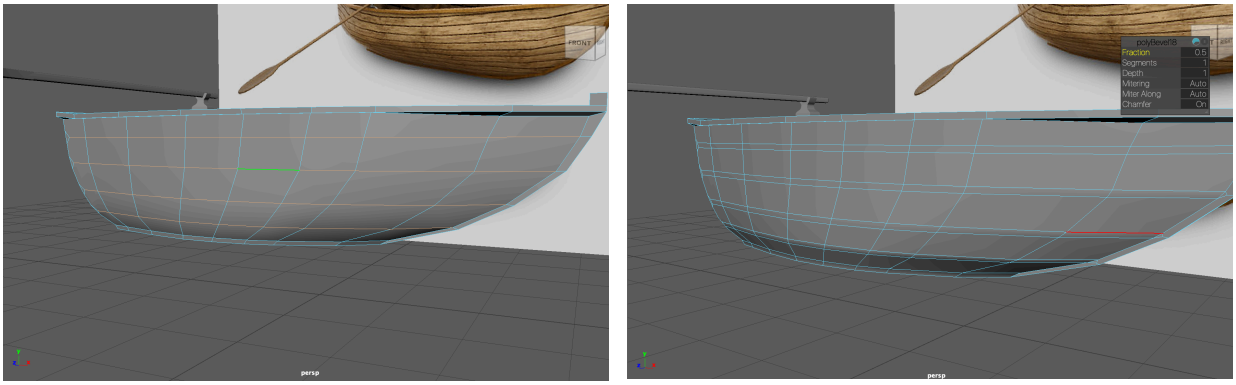
[Figure 54 Inward extrusions]

It was important to maintain even wall thickness across the model in order to have a realistic approach to the rowboat.

At this step, it also became important to apply bevels to edges in order to make more natural surface transitions, as opposed to sharp edges. This adds to the realism of the boat



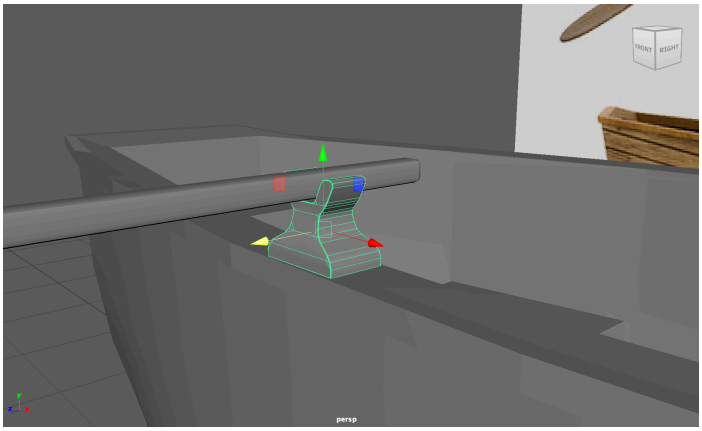
Here, the sharper edges of the back of the boat were mitigated with the bevel tool, which increased the number of faces and made the overall surface smoother.



[Figure 55 Smoothing out sharp edges]

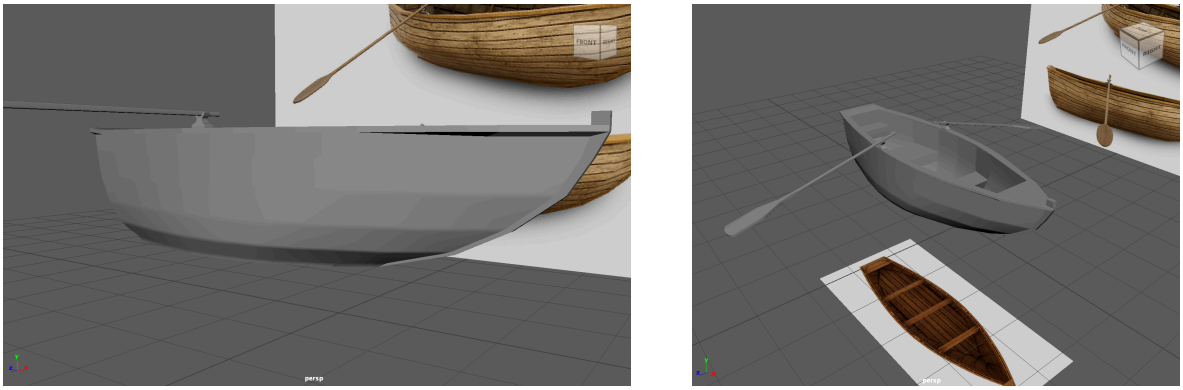
The completed model reflects an iterative design process focused on geometric accuracy, clean topology, and visual realism. Repeated refinement of the hull improved both technical quality and modeling efficiency. The project reinforced key principles of 3D asset creation, including controlled edge flow, proportional design, and iterative correction.

It is worth noting the process of achieving the oar and the piece that connects the oar to the boat. Both of these were separate objects from the boat and were achieved using a similar manner, starting with primitive cubes and achieving a rough model, and finally smoothing things out with the bevel tool.



[Figure 56 The connecting piece’s geometry]

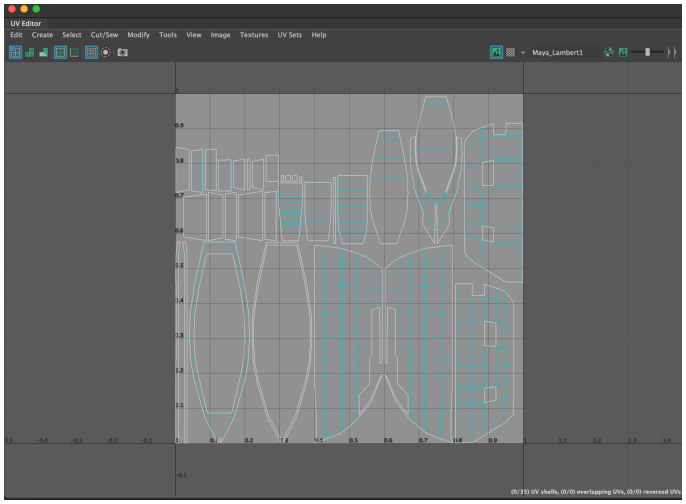
The boat was finished in roughly three weeks, which took significantly more time than initially thought, since the original plan was to get it done in one week (with the next week dedicated to textures). However, due to the relearning and repracticing of Maya tools, the product was delayed a few weeks.



[Figure 57 Finished model]

Textures

The last part that needs to be achieved to complete this sprint is the textures, which starts with completing the UVs, which defines how the textures will wrap around the model. So far, the UVs have just begun, but it will take about a day or two to complete. The hardest part of this task is lining up all of the beveled faces to their UV shell, since the UVs automatically had some beveled faces attached in irregular ways (some beveled faces were attached to other parts that did not make sense geometrically). It is necessary to cut the UV shells and reattach them manually in order to have clean UVs, which will result in a clean-looking texture.



[Figure 58 UV Shells]

Once the UVs are completed, we will apply the textures in Unity and have our completed rowboat.

Restoring VR Functionality

Objective

For this project we are continuing off of a previous team's senior design project and adding new components to it. This primarily consists of creating a 1:1 realistic replica of Fort Matanzas based on provided blueprints and creating a boat ride experience that transports the player from Fort Marion to Fort Matanzas down the Matanzas river. As this is a continuation from a previous project, there were leftover issues in the project that we had to deal with before moving forward with new content and features. The objective of this task is to fix all previous technical issues in order to restore

proper VR functionality and allow the user to comfortably enjoy and interact with this digital experience in virtual reality.

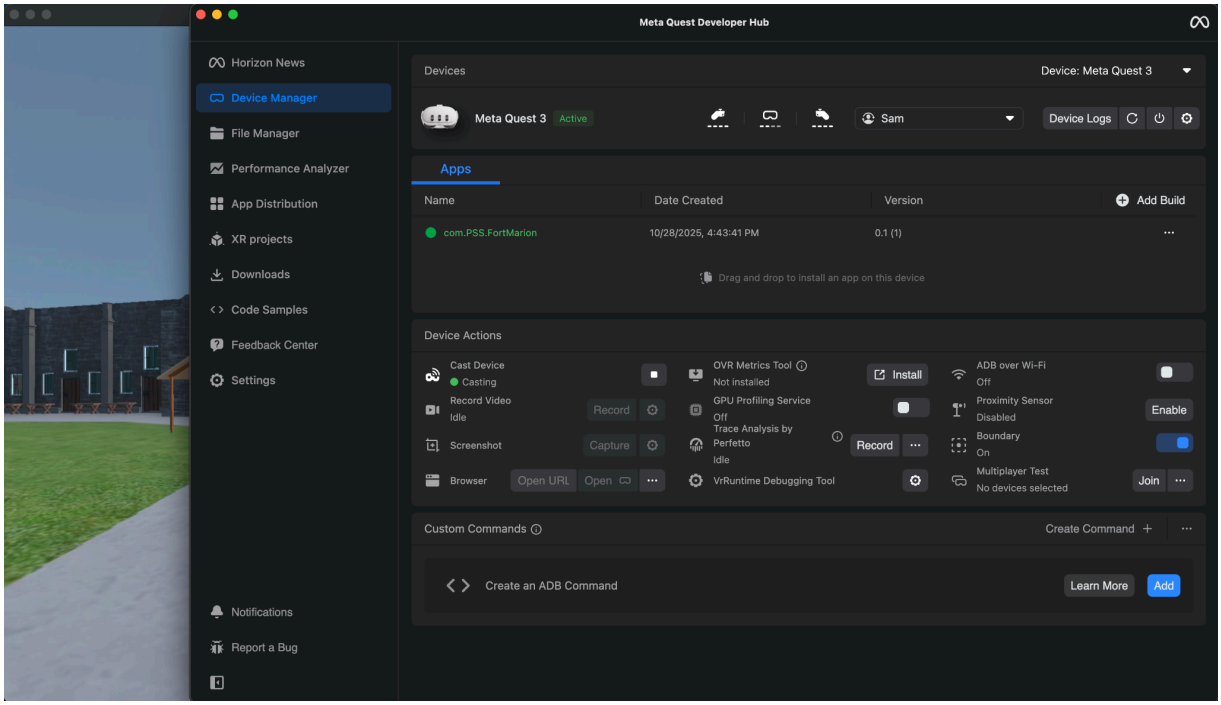
Preparation and Roadblocks

Virtual reality and Unity development were foreign concepts to much of the team at the beginning of this project, so a significant amount of research and preparation was necessary prior to completing this task. We began by looking at Unity tutorials online to familiarize ourselves with this complicated software that we will be working with for the next few months, and over time working with the editor became more of a streamlined process. With the main focus of this task being virtual reality, we also did research on the Meta Quest headsets and how to develop for them, as that is the platform we are targeting for this project. Unfortunately, some of us do not have the hardware necessary to run the game at home, so the Meta Quest 3 headset at the senior design lab in HEC 102 has truly come in handy. Although we have discovered that Unity does have a way to simulate VR controls, it is not nearly close enough to how the game will actually be interacted with by our target audience using a VR headset. Using the Meta Quest 3 in the lab has allowed everyone on the team to debug and test the game how it should be played. Working with unfamiliar hardware does come with its own challenges though. The first one being setting up the headset, which sounds simple but provided us with more difficulty than expected. The headset has not been used in a long period of time so it was necessary for

multiple updates to be installed for the headset's software. However, this requires an internet connection, and the headset was having problems with connecting to both the UCF WPA2 and UCF Guest WiFi networks. We ended up getting around this issue by connecting the headset to a mobile hotspot WiFi network, and although the download speed was very slow, this ended up working in the end. Now that the Meta Quest 3 was set up, the next step was to prepare the headset for development purposes.

As the Meta Quest 3 is a consumer product, by default it does not allow software that is not published to be installed or ran on it. So we had to do research on how to install the project onto the headset and run it. Fortunately this was a relatively simple and straightforward process. The first step was creating a Meta developer account. Once the account was made we had to create a new developer organization and invite each of the team members to it. Then we had to enable developer mode in the device's settings menu. Lastly, to open the newly installed application we had to navigate to the Unknown Sources section of the application library menu and open the latest development build of the project. One piece of software that has been especially useful is the Meta Quest Developer Hub. This application allowed us to connect the Meta Quest 3 to a computer and interact and monitor it from there. The primary function we have been utilizing this application for is installing developer builds of the project onto the headset. Fortunately, this is very easy to do. We just need to drag the .apk file, which contains everything needed for the game, onto the device manager screen and it will install onto the device when connected to the computer with a USB-C cable. We are also able to take screen captures and video recordings

of what is displayed on the VR headset's screen during gameplay, which has been a very useful feature as well for sharing progress with the team and our sponsor. The display of the VR headset can also be mirrored to the Meta Quest developer hub when the headset is connected with a USB-C cable, which is useful so other team members can view what is happening in the game while another team member is playing it. It has also provided several of the screenshots that are present in this design document.



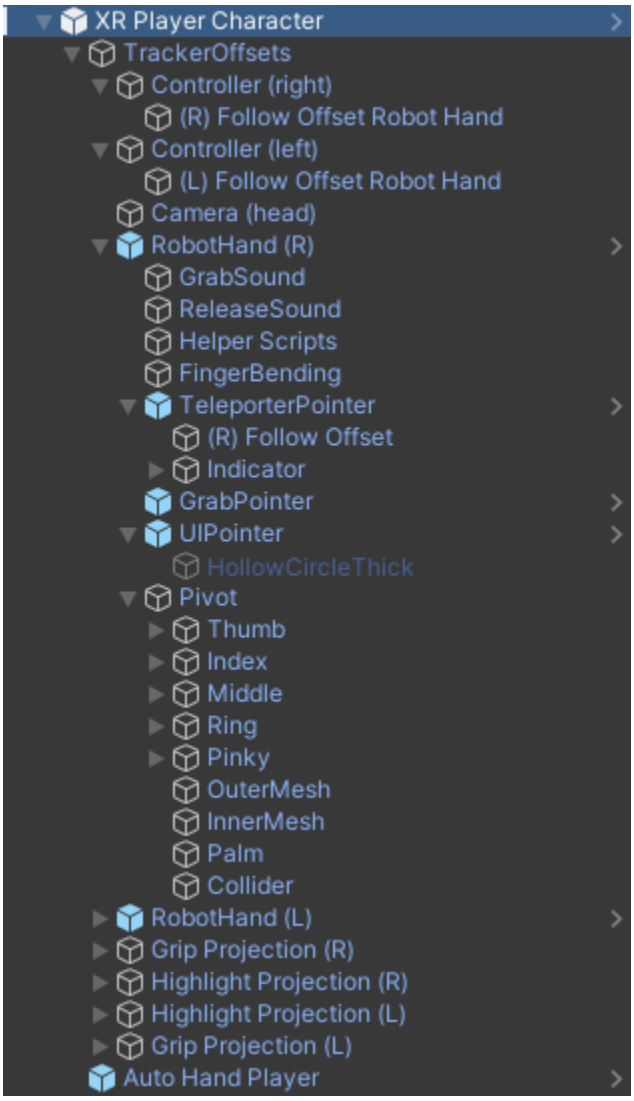
[Figure 59 Meta Quest Developer Hub]

Back to the .apk file itself, the process of compiling and building an executable of the project was a difficult but mainly time consuming process at first. The first thing that needed to be done was switching the platform from Windows/Mac/Linux to Android. This needed to be done because the software used on the Meta Quest 3 is Android based, which is something else we learned while completing this task. This process took several minutes because all of the assets had to be reimported and shaders had to be recompiled to be compatible with the Android platform. This project contains a significant number of assets, which include things such as 3d models, textures, sound effects, and C# scripts. This took especially long because I began working on this task before all of the unused assets were removed from this project by one of our team members. We ran into another issue too before having a viable APK of the project though. The color space setting had to be changed from gamma to linear, because the Meta Quest rendering pipeline uses a linear color space. This was a simple fix as just one setting had to be changed in the Unity editor, but recompiling everything to be compatible with a linear color space took a large amount of time. After all of these obstacles were overcome and we did our research on developing for this platform, we were able to begin the task of restoring the VR functionality of this project.

XR Player Properties

In the Unity game engine, each gameplay environment that the player interacts with is called a scene, and each scene contains several game

objects that fulfil a variety of purposes. A game object is something that exists in our scene, such as an NPC, a cannon, or a door in the fort. In our project, the most important game object is the XR Player Character. This object is how the user interacts with the fort environment, and handles several tasks. There are multiple child objects that are part of the XR Player object, first let's look at the Auto Hand Player. This object handles player movement as well as the movement and interactions of the user's VR controllers. This object does not have any child objects of its own but it has several components that affect the behavior of the player's movement.



[Figure 60 XR Player Character structure and child objects]

There are a couple colliders, which give the object physical space in the environment and collision properties that affect how it and other objects behave upon contact. There is also a rigid body, which applies physics to the game object, such as gravity, acceleration, and friction. Both of these help the Auto Hands Player object behave as if an actual person would, but

to a simplified extent of course. There is also a complicated C# script that defines how the game object behaves and how input from the VR controllers affects the player's movement and actions.

There is another child object called TrackerOffsets which contains several child objects of its own, such as a camera which is how the user sees the scene and what they are doing, two 3d models of robotic hands (RobotHand (R) and RobotHand(L)) that are how the user interacts with the fort, and more. The hands track the positions of the 2 VR controllers, which mimics the user's actual hand positioning. Pressing the trigger buttons on the back of the controllers lets the hands interact with items in the fort. There are also objects under TrackerOffsets that assist in grabbing and interacting with objects in the fort and help provide additional functionality to the RobotHand objects. These are the HighlightProjection and GripProjection objects. Lastly, there are the Controller (Right) and Controller (Left) objects, these keep track of the real world position of the two VR controllers and map their positions to the positions of the two RobotHand models. The XR player object is a great example of how something as complex as a player character in a VR environment can be broken down into several components that work together in the Unity game engine. Although it looked complicated at first, after breaking down the object's structure it is clear how each child object plays its own important role in providing a smooth and interactive experience to the user.

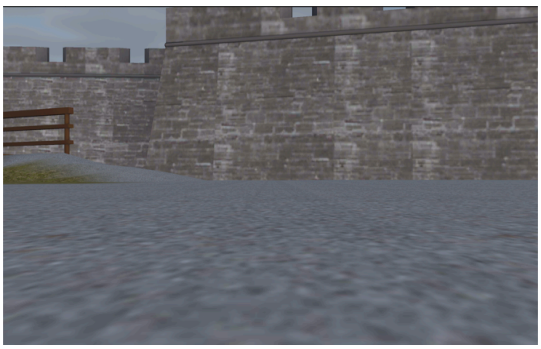
Identifying and Fixing Issues with First-Person View

This was an intimidating task to take on, as this project and its structure are massive and seems confusing at first. It took hours to understand how each game object interacts and what its purpose is. We were able to understand things better by examining the tree of game objects in the Unity editor's sidebar and looking at each of their components. Another useful feature that helped with understanding how each game object functions is disabling certain ones and seeing how it affects gameplay when running the game in the Unity editor. When doing this sometimes models disappeared or the behaviors of other game objects changed. The purpose of fully understanding the project's structure and what each game object does is figuring out how to fix the problems with the VR experience that remained in the previous version of the project. The main issue was the camera attached to the XR player kept falling to the ground and staying there when the game is run. From a first-person perspective, the user could only see the ground and not much of anything else. This is certainly not how the fort is meant to be experienced, so this task became a priority when we began working on the project. Actually figuring out what was causing this problem resulted in being quite the tedious task, as there were many culprits as to what was causing this issue to occur.

The first thing that was considered was the camera, as this seemed to be the most obvious source of the issue. We tried editing the y-axis position of the camera, to see if raising the camera up will prevent it from being too

low. While this worked, the hands were barely in view and the position didn't match the location of the actual player's hands, which ruins the immersion of this effect. It also made interacting with objects in the fort nearly impossible. After reverting the camera to its original position, we started examining other child objects of the XR player character object. We tried disabling gravity on the Auto Hands player object but the camera still fell and behaved the same as before. After much trial and error though, we noticed that the object called TrackerOffsets had a modified x and z position, as well as several child objects including the 3d models for the hands and the camera attached to the player's head. Modifying the position of an object also applies those changes to all of its child objects, so we attempted to solve this problem by increasing the y coordinate of TrackerOffsets. Not only did this fix the camera height, but it also moved all of its child objects up with it so the hands were in view and behaved as expected. After much testing, the y-position was increased from 0.00 to 2.75 in order to position the player's view at a realistic first person height to explore and interact with the fort. We did some experimenting and playtesting on the actual hardware to find a good, realistic height to set the player to. Something that helped with this is comparing the height of the first-person camera to the height of the NPCs walking around the fort, so the player would be at about eye level with them, simulating the height of a real person. With the camera and all other necessary components set to a proper height, the user is now able to properly play and enjoy this digital experience, and our team was ready to move on to new tasks with this blocker out of the way. Displayed below is the difference between the old camera height and the new camera height after implementing this fix.

[Figure 61 Old camera height (left) vs the fixed height (right)]

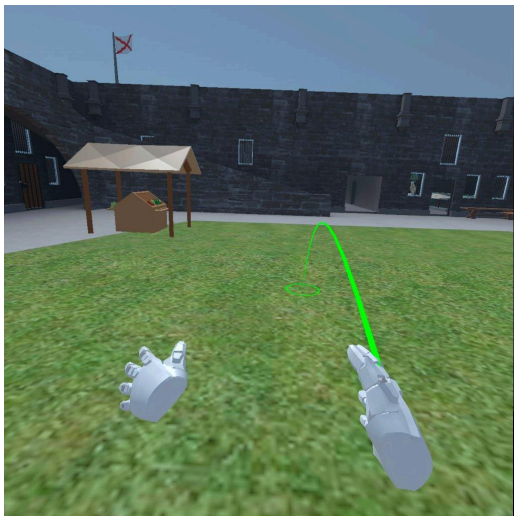


Teleportation Feature Issue

Overview and Problem

In our project, there are two different ways to navigate the environment. The primary one is moving around on foot which is done by moving the left control stick, a tried and true method for movement in a 3d game. The second method involves teleporting to a location, which is useful when the player wants to enter a room in the fort or go up the stairs to the second floor. This is done by pushing and holding the right control stick forward to reveal a circular cursor. The user can aim this cursor by moving the right controller while holding the right control stick forward, and then releasing the right stick once a valid location is decided on. The user is able to teleport within a fixed circular radius, and the cursor will be highlighted red if the user is attempting to teleport to a location that is out of bounds. This is a feature left over from the previous version of the project, and it is

not immediately intuitive to the player. For example, when they try to enter a room but can't they may not know right away that they are required to teleport into it. To remedy this, we can add a brief tutorial before playing the game for the first time, but this can be skipped for players who do not want to go through it before getting to the main experience.

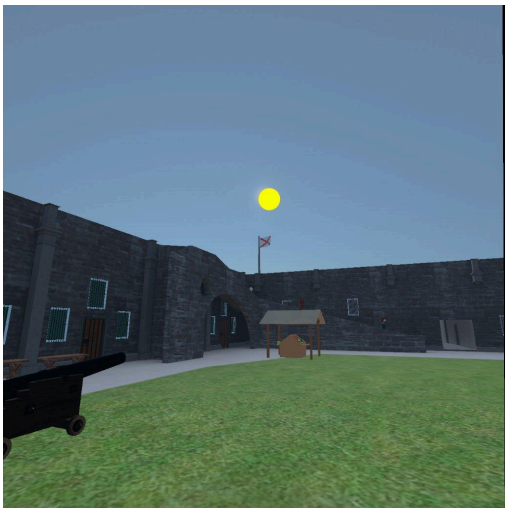


[Figure 62 Teleportation mechanic in action]

Although this feature is helpful and unique, there is a significant issue with it that once again involves the height of the player. After the user teleports to a new location, they are permanently stuck at a new height that is a few units higher than they were before. This throws off the perspective of the game, as the user now appears to be giant, towering over the fort and interactable items. This makes the fort much more difficult to navigate and some objects such as the cannons on the second floor impossible to interact with. There are also several informative plaques around the fort that cannot

be read anymore because the camera is too high up for the user to read their contents.

[Figure 63 Player view before (left) and after (right) a teleport)



Finding the Cause

Initially, we did not fully understand what was causing this issue yet or what was required to fix it. It became our priority to fix this issue in the next sprint as it significantly impacted the quality of the experience. Also, once it gets fixed, the game will be much more playable and we should be done with

dealing with technical debt from the previous project. This issue is more complex than the previous one that dealt with the camera height. This is because the last issue simply required making changes in the Unity editor to fix the problem. But because this issue is something that occurs when a specific action is performed during runtime, we had to do some searching to determine which script controls teleportation behavior and what needed to be changed in that script in order to fix this issue.

In the Unity game engine, anything that causes the behavior of a game object to change is done through the creation and implementation of scripts. These are created in the C# programming language, which fortunately was fairly straightforward for our team to pick up. This is due to how similar the syntax of the language is to Java, as well as them both being object oriented programming languages. Some members of the team have also taken the programming languages class that is offered at UCF, where we were taught the key principles of C# and created programs in that language. Fortunately, there are several extensive and useful libraries for the Unity game engine that one can import into a C# script. These libraries can be used to reference and change the behavior of game objects based on input, which is the task at hand for fixing the teleportation issue. An example of this is “UnityEngine.InputSystem”, which can be used to read various forms of user input. Based on the input that is received, different blocks of code in the script can be executed. This is what bridges the gap between the player pressing a button or a key and something happening in a scene. There are also libraries which manage how the user interacts with the scene using the vr headset and controllers. For example,

“UnityEngine.XR.Interaction.Toolkit”, which is used extensively in various C# scripts in this project.

As previously stated, the first challenge at hand for fixing this issue is locating the root cause of the problem. There are several scripts to look through to locate the source of this issue with many different folders with scripts in them too. First, we looked at the simply named “Scripts” folder and found a file named “TeleportationManager.cs”. This seemed to be the obvious culprit for where we could find the code that is causing this issue to occur. However, this name was slightly misleading because the code that determines the actual teleportation behavior was not found here. This made things more difficult, so we had to do some more digging to determine where we could modify the teleportation behavior to prevent this issue from occurring. This proved to be more difficult than expected, but eventually we located the script we were looking for, which was “Teleporter.cs”.

The reason why it was not straightforward is because this script is located inside a folder for one of the packages that was imported into the project, Auto Hand. The Auto Hand package tracks the movement and button inputs of the controllers and reflects these in two 3d models of hands that are used to interact with the environment. In this case, pushing the right stick forward shoots a laser out of the right hand of the player and the movement of their real right hand determines the location that the user will be teleported to. Therefore, it makes sense that the teleportation script is located in this folder, it was just a bit hard to find at first. But by doing this

search, we understood the file structure of the project better and how various C# scripts are connected to the game objects that they act on.

Determining a Solution

At first, we were unsure of how to modify the code to fix this problem and where in the Teleporter script we should edit. We started our approach to fixing this issue by looking through the file to understand the different variables and what each method does. Thankfully, the last team who worked on this project did a good job naming each method so determining what each of them does was not too difficult of a task. The method Teleport() does exactly what its name says it does, and its logic is fairly simple to follow and understand. It gets the location to teleport the player to from the position of the circular teleport cursor, then checks if it is a valid place to teleport to. If it is, the player's position is updated to the position that the teleport cursor was at and the teleportation is complete.

There is a variable called targetPosition that is a three-dimensional vector which stores the coordinates of where the player should be teleported to. In order to fix the issue of the player being too high up after teleporting, we came to the conclusion that modifying this variable may be the key to fixing the teleportation bug. Earlier when fixing the camera height issue, we modified the TrackerOffsets game object by changing the y-coordinate from 0.00 to 2.75. Our hypothesis was that while increasing the height offset

fixed the height of the player walking around, it had a detrimental effect on how teleportation was carried out. This resulted in the player's height being stuck at too high of a value after teleporting. To offset the new player height after teleporting, we created a new variable in the Teleporter script called teleportOffset. This was also a three-dimensional vector, the data type being defined as Vector3 in the UnityEngine library. We defined this variable with a value of (0, -2.0, 0), so this vector is simply an arrow pointing down in a three dimensional vector space with a magnitude of 2.

Now that this vector is defined, our plan to fix the teleportation issue was to undo the positive height offset in the trackerOffsets object by adding a negative offset to the player's position when teleporting. Although there is a positive offset of 2.75 in trackerOffsets, using a negative offset of -2.75 resulted in the player being teleported into the ground. So after some experimenting with different y values, we decided on a height offset of -2. Subtracting one Vector3 from another Vector3, in this case subtracting teleportOffset from targetPosition, conveniently works just like how subtracting vectors works in mathematics. Therefore, we could simply update the code for where targetPosition is defined by adding the negative offset of teleportationOffset to the position of the teleportation cursor before teleporting the player object to that coordinate. With this fix in place, teleportation now works as expected. The player's height does not get stuck at a certain y value like it was before and the player is positioned at the proper height for where they are teleported to. Now that this fix has been implemented, there is no longer any leftover technical debt from the previous version of the project. The user can now explore the fort easily and

movement behaves as expected, greatly improving the overall gameplay of our virtual reality experience.

[Figure 64 Player view before and after teleporting into a room]

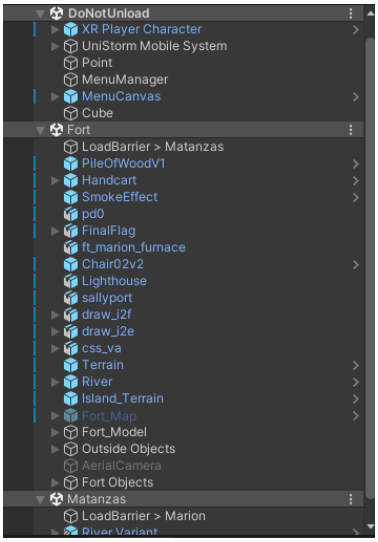
Load/Unload Barriers

Objective

Implement a system that loads and unloads each fort and its surroundings as a unit without performance impacts.

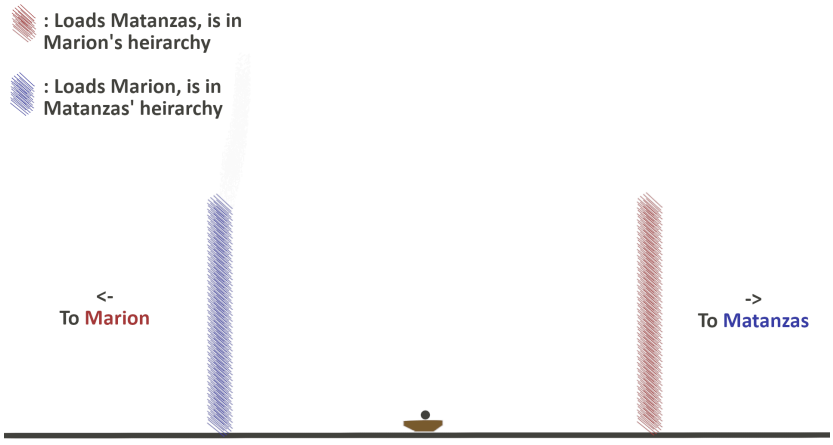
In order to keep unused scenes from using unnecessary resources, we implemented invisible barriers that unload whichever scene the user is coming from and load in whichever scene the user is going to. This is

accomplished with Unity's additive scene loading tools. To load scenes additively, first assets must be organized into separate scenes, including a DoNotUnload scene. The DoNotUnload scene works as a place for all objects that should not be unloaded on scene change, such as the player controller, environmental sounds, and lighting. After being separated into scenes, each scene can be loaded into the same editor space in separate scene hierarchies as shown in figure 61. Everything within a scene belongs only to it, but loading them all in the same editor space allows us to seamlessly line up each scene to be loaded as if they are part of a continuous environment. We are making use of line of sight to ensure that loading does not appear as a "pop-in" by only initiating the load/unload script when both major scenes are out of sight.



[Figure 65: Scenes]

Our load/unload process consists of two invisible barriers, each within the hierarchy of the scene they unload. Each is placed past where the other would be, as shown in Figure 62.



[Figure 66: Load/Unload system Illustration]

Only one barrier will be loaded at a time, as only one scene is loaded at a time. The space between the barriers exists as a buffer if the user backtracks, and the recently unloaded scene needs to be reloaded. Using this method we can guarantee that only the scene the user is going to is loaded, and the scene they are coming from is unloaded. We experimented with using flags or other forms of keeping track of which scene is loaded using a single barrier, but we determined that this was a simple solution that tied its states directly to toe load states of each scene. Additionally our intermediary river terrain, which these barriers are placed in spatially but are not in the same hierarchy as, will be placed in the DoNotUnload scene, as there is currently not a state where we would want to unload it.

While the method and code works in a separate testing environment, when tested in the project we ran into an issue loading the Fort Marion scene. The scene unloads correctly, and the Fort Matanzas scene loads correctly, however on loading the Fort Marion scene again, all associated audio files play simultaneously. We think a combined factor of all audio playing at once and loading all assets in the Matanzas scene is causing the program to freeze when reloading this scene. This has been placed on the Jira as a bug and we plan to look into ways to stop the audio from playing on load, and if there are still performance issues, we will look into loading the scene gradually, starting with the terrain and outdoor models, then the fort, then all indoor models.

Future Tasks

Boat Ride

Overview

One of the main features of our project is the interactive boat ride that allows the user to travel along the Matanzas River between Fort Marion and Fort Matanzas. This feature is designed to give users an immersive way to experience the river environment while connecting both historical sites within the virtual world. The boat ride will function in both directions, meaning users can travel from Fort Marion to Fort Matanzas, as well as from Fort Matanzas back to Fort Marion.

User Interaction and Restrictions

The boat itself will move autonomously along a predefined path, allowing the user to remain seated and enjoy the experience without needing to control the movement directly. To maintain immersion and prevent gameplay issues, the user will not be able to jump off or enter the water at any time. The water is not a walkable or swimmable area in the project, so the player’s interaction will be limited to staying safely on the boat throughout the journey.

Minimum Requirements and Potential Enhancements

The minimum requirement for this feature is that the user can successfully ride the boat from one fort to the other and back again. However, we are

also considering several possible enhancements for future iterations to make the experience more engaging. Some ideas include:

- Allowing users to control the boat's movement manually.

- Adding interactive or entertaining elements during the ride, such as a short minigame, ambient background music, or historical narration.
- Introducing multiple boats traveling simultaneously along the river.

These additions would make the boat ride feel more dynamic and interactive while preserving the educational and immersive aspects of the project.

Technical Implementation

From a development standpoint, several technical tasks must be completed to bring the boat ride to life:

1. Path creation: We need to code a set path for the boat to follow along the river, likely using Unity's spline or waypoint system to ensure smooth, natural movement that accurately follows the curvature of the river.
2. Speed and movement: We must determine the optimal speed fast enough to keep the ride engaging but slow enough to allow users to appreciate the scenery. The movement must feel stable and comfortable, avoiding sudden turns or camera shakes that could cause nausea or disorientation.
3. Seating interaction: We will implement a system that keeps the user stationary while the boat is moving, locking their position to prevent glitches or unwanted movement.

4. Safety restrictions: We will include collision and boundary checks to make it impossible for the user to jump off or leave the boat.

Final Considerations

Overall, the boat ride will serve as both a functional and immersive link between the two forts, offering users a smooth and engaging way to explore the Matanzas River environment. Once the base system is working, we will refine the ride to ensure it feels natural, visually appealing, and aligned with the project's historical and experiential goals.

Creating User Boundaries

We will create boundary walls to keep users from moving into the water and prevent out of bounds teleportations. These walls will have to permit users to be on the water only while in a boat and restrict movement to areas the user is intended to see. This will involve looking into creating invisible collision planes, as well as how to designate areas the player is forbidden to teleport to with VR controls in Unity. We are anticipating this task will be more difficult than it initially seems, but intend to fully implement it for the sake of project quality.

Populating the Riverbanks and the Area around Fort Matanzas

We will use existing tree and house assets to populate the riverbank, and we plan to create a 2D plane to hold a backdrop forest texture to block line of sight between the forts and block line of sight to areas without geometry. The 2D backdrops will sit behind a couple layers of trees to make

the foliage appear more dense, and create a more varied skyline silhouette to make the world seem less flat. While this is not a perfect solution, we have determined it is a simple solution that will have minimal performance impact as we still have to run this on a quest 2 at a target of 90 FPS. We are considering using multiple planes with transparency as well to achieve a more convincing parallax effect while riding through the river. Based on some of the files we deleted, it seems like the previous group tried something similar, but those assets did not seem to fit the project. The backdrops around fort Matanzas will also create a boundary that prevents the user from playing the designated area, accomplishing part of the "creating user boundaries" task

Build/Prototype/Test Plan

There are many steps that go into building our project into a functional prototype and testing it on a real VR headset. First, we build a .apk file in the Unity editor in the build settings window. This is done by selecting Android as the build target and then checking the development build option. If there are no errors, Unity outputs a .apk file that contains all the necessary resources to run the game. Then, we connect a Meta Quest headset to the computer the project was built on with a USB-C cable and open the Meta Quest Developer Hub software. To install the .apk onto the VR headset, first go to the device manager tab and select the headset that is plugged in. Once that screen is open we simply drag the .apk file to where it says install apk file, and the software installs our project onto the device. To launch it, open the app library on the headset and go to unknown sources

and select the recently installed game. Now a fully working prototype of our project is now installed on true Meta Quest hardware! For testing purposes, we have primarily been using the Meta Quest 3 provided in the senior design lab. If we leave the headset plugged in and the Meta Quest developer hub open while running the project, we can take screenshots and observe details about the hardware during testing which has proven to be very useful.

Conclusions

Over the course of this project, our team has focused on establishing the technical foundations, workflows, and historical references needed to continue developing the VR reconstruction of Fort Matanzas and Fort Marion. While the project is not yet complete, the work done so far has allowed us to better understand the scope of the tasks ahead. The challenges we encountered, such as learning Unity's VR tools, navigating Blender for detailed modeling, managing inherited technical issues, and optimizing large asset libraries, have shaped our approach moving forward and clarified what is needed to achieve the final vision of this project.

Characterization of Results

The results at this point represent foundational progress. Key accomplishments include:

- Initial VR Issues Addressed: Major inherited issues such as the camera height problem and controller interactions were identified and partially corrected, restoring basic VR functionality for testing.

- Early Fort Modeling: Core structures of Fort Matanzas, including the tower, second floor, roof, ladders, and stairs, have been modeled in their early or mid-development forms using historical plans and reference materials.
- Terrain and River Preparation: Heightmaps and terrain tools were used to begin forming the Matanzas River region, creating the groundwork necessary for the future boat ride experience.
- Asset Cleanup: Many unused or inappropriate assets were located and removed, improving performance and keeping the project maintainable.
- Improved Development Workflow: The team now has a stronger understanding of Unity, Blender, Meta Quest development, and collaborative version control, allowing future work to proceed more efficiently.

Summary of Project

The project aims to create an immersive virtual reality experience that reconstructs Fort Matanzas and Fort Marion as they historically existed, ultimately allowing users to explore both forts and travel between them by boat within an education VR environment. The work completed so far centers on establishing the 3D models, terrain foundation, VR functionality, and the project structure. Although, much remains to be completed, including texturing, environmental detailing, and integration of interactive features.

Acknowledgements

We are grateful for receiving assistance for this project from multiple outside sources. The first being access to the senior design lab as a space and also for the hardware it has provided to us. We have used it as a place to meet many times and also have extensively made use of the Meta Quest 3 headset at the lab.

Additionally, I would like to acknowledge Amy Giroux and Brook Miller from the UCF Center for Humanities and Digital Research for not only sponsoring this project but also for providing feedback and assistance as we work on the project. Our sponsors have been active in Discord, responding to screenshots of our progress that we send in chat, as well as answering any questions we ask them. We have also met in voice chat with our sponsors on several occasions to discuss sprint progress and address any inquiries. Lastly, they provided us with many essential files in a timely manner when we began the project. This includes the former team’s final design document, the Unity files for their project, a thorough description of the project, and very well made blueprints of Fort Matanzas for us to follow when constructing a 3d model of the fort. All of these resources have been crucial for our success in this project and we are more than happy to work with the UCF Center for Humanities and Digital Research.

Bibliography

[1]Historic American Buildings Survey: Fort Matanzas, St. Johns County, Florida. Measured by Everett Mead et al., U.S. Department of the Interior, National Park Service, Branch of Plans and Design, 26 Feb.-26 Mar. 1934. HABS No. FLA-55-15-5.

[2]Vegetation and Developed Land Classes of Fort Matanzas National Monument. U.S. Department of the Interior, National Park Service Dec. 2018

[3]U.S. Department of the Interior. (2025, March 2). *Fort Matanzas National Monument (U.S. National Park Service)*. National Parks Service. https://www.nps.gov/foma/index.htm