Fort Marion in VR

Team 17

Sterling Downs

Landrey Martin

Howard McDavid

Juhwan Park

Enrique Rodriguez

Phillip Zou

Team Sponsor

Dr. Amy Giroux

Contents

1. Executive Summary

Senior Design allows for undergraduate students to cooperate and work in an environment that allows them to create, design, prototype, and create a product for the purpose of demonstrating a finished product for their respective sponsor. Members of the project are chosen based on a ranking system to ensure that interest in the project remains high through the two semesters it takes to create the product. It's crucial for students, especially in Computer Science, to be introduced to these topics and practices before entering the workforce as these are skills that can be universally applied, regardless of what field they choose to enter in or project they end up working on.

Before starting the project, students must learn to ask questions and establish guidelines for themselves so they understand the scope of the project they are working on. Their assigned sponsor then establishes must have features and additional features that would be nice to have if time allows. This process allows students to understand how to pace themselves when balancing multiple workloads given a hard deadline.

During the project, students are expecting to engage in active communication with their groups. This can take place in the form of Agile principles or some similar practice like Lean. By implementing these practices, students can learn to engage in continuous communication with their group members and their sponsor.

Our goal for this project is to create a VR recreation of Castillo de San Marcos in VR for the preservation of National Parks. As more time passes, historical landmarks continue to degrade in quality caused by factors such as environmental effects and human interaction. The materials used to create many landmarks as we know today, were not meant to stand the test of time, and as such, many landmarks across the world are undertaking large reconstruction efforts to mimic what landmarks used to look like before becoming eroded. This project intends to do something similar by creating an interactive environment that can't be damaged by any factors. This allows us to preserve the fort as it stands currently and allow users to experience previous versions of the fort. Using this project as a baseline, future projects could be undertaken to preserve other monuments/landmarks in a similar fashion. The possibilities and interest in this area allows for many people to interact with history in ways they would've never had otherwise.

2. Project Narrative

Castillo de San Marcos, also known as Fort Marion, is now a historical landmark located in St. Augustine, Florida. The National Park Service currently offers tours around the fort but would like to be provided a VR experience for the public to be able to access. They want users to be able to learn everything there is to know about the fort without having to be there due to physical or geolocational restrictions the user may have. They would also like to add features to the experience that they cannot in an in-person tour. Currently, the public is limited to photos and tours of Castillo de San Marcos. There are some areas that are off-limits to the public, even barriers in areas that are not accessible to the public. Visitors are not permitted to touch most of the structure or things within the walls in order to preserve the site. One major feature they want to implement is to allow users to interact with pieces of the fort. They would like the virtual visitors to have the capability of interacting with doors, windows, the drawbridge, cannons, and more around the fort.

There is so much history behind this 300 year old structure, which is why the structure is on display for the public to learn all about its important role in Florida history and beyond. It's also the reason the National Park Service wants to make learning about the fort more accessible to the public.

Castillo de San Marcos was built by the Spanish between 1672 and 1692 to replace the previously built outpost at the same location. The original fort in that location (the original Castillo de San Marcos) was a wooden structure and in horrendously unsafe conditions by 1668. After they were unable to defend St. Augustine against pirates that were seemingly planning on seizing the fort as their own, Spain finally kicked into action and pushed for reconstruction of a more efficient outpost to protect Spanish vessels traveling back to Europe. They sent more troops and money to St. Augustine to assist in the reconstruction and defense efforts [1].

One of the most unique qualities of Castillo de San Marco is that the fort is built out of coquina rock. Coquina rock is quite the intriguing choice for a foundation material, as it's supposed to hold up against heavy beatings from weather, beatings from the ocean, and attack from invaders. There has been no record of another large structure or building being built out of coquina before the fort, so the settlers in St. Augustine were not sure how it would hold up and only time would tell. The material ended up being more efficient than anybody could expect. It held up perfectly against attack, to the point where cannonballs

directly hitting the walls would rarely leave a mark on it due to the coquina rock absorbing the hit. Since it was built in the 17th century, Castillo de San Marcos has never been captured in battle, although it has been occupied by several forces through the centuries following [2].

3. Motivation

Sterling Downs As someone with an interest in history, this project stood out to me a lot. The idea of building a representation of Fort Marion in virtual reality not only makes visiting the fort more accessible to people around the world, but also offers new ways to depict the fort. Additionally, virtual reality puts control in the hands of the user and allows them to do things they wouldn't normally be able to do—such as firing the cannons.

Not only does this project have a good purpose, but I also see it as a great opportunity to expand on my existing skill set. Being able to further explore technologies such as virtual reality and 3D modeling is something that I have always wanted to do, but never really had enough time nor purpose to do so. Since the project is designed to be maintained and expanded upon, it will also be interesting to follow any future changes or additions to the project.

Landrey Martin This project called to me as soon as it was being pitched to the class. There were so many different aspects that intrigued me and solidified why I wanted to be working on it. I have always been a fan of visiting historical locations and learning the stories surrounding them. By working on this, I can help others learn all about this historical landmark, no matter how far away they are. I believe this could be the beginning of many similar areas and attractions being more accessible to the public, as well as another step towards preserving these historic landmarks. The less people that physically visit them, the better we can maintain the structures.

As it was being pitched, I was reminded of an internship I had in high school with the facilities planning department for my public school board. I was working with floor plans and designing building remodels/demolition plans through most of my time there; however, as I was leaving, we were beginning to take 360-degree aerial view photos for future reference.

I have a small amount of 3D modeling experience; however, I was using a different software than we will be using on this project. I am excited to get to learn all about the fort, VR development, 3D modeling, and more as we work on this project. This project has me extremely excited to expand my experience beyond what I've been learning through my coursework and I'm glad that we're going to be developing a PC version alongside our VR tour of the fort. The PC version will provide accessibility for those that do not have a VR headset, but still would like to learn all about Fort Marian.

Howard McDavid My main motivation for choosing this project is I thought it would be a fun project to work on. In addition, I've had over 3 years experience working on 3D models for poorly optimized games. Although the bulk of my experience has been mid- to high-poly railroad related models, I have always wanted to expand into lower-poly building models and this is the project that would jump start that work. I have also had some experience in Unity, but none with VR development, so this project would also help further my skills in terms of Unity VR. I have also always had an interest in history, and, while I have never been to Fort Marion, it seems to be a very interesting bit of history.

Juhwan Park Having learned how to design websites in Processes of Object Oriented Software Development last semester, this time I want to learn and build a VR environment and compare and contrast how different this experience will be from the previous one. I am also very excited to touch back on blender, the 3D object designing software that I have played with when I was in middle school. Simply thinking of how we will put the 3D model from blender with Unity to make objects interactable also delights me.

Building Fort Marion in virtual reality and as a PC version will allow people to view and explore the historical fort without having to get there physically. As traveling to the fort will be difficult for people further away, this ease of virtual access for those that desire to explore also excites me. Having explored it virtually may also motivate users to explore in person and I believe that this project helps us open more possibilities to develop virtual environments, virtual access for people and ultimately create a learning environment which I believe would benefit the society and improving the society is exactly what I want to do.

Enrique Rodriguez My primary motivation for choosing this project was a result of my recent interest in VR development and my general love of historical

projects. As I approached the conclusion of my undergraduate degree, I took up a hobby in Unity game development. The abundance in online resources made learning Unity an endeavor that is both incredibly fulfilling and fun to pursue. Whereas most CS classes revolve around learning a new programming language, learning Unity allowed me to learn about optimization, user interactivity, and intelligent game agents to name a few.

Additionally, I have always had a knack for anything historical. Besides Computer Science, History is a field that I love to learn about and wish I could have a career in as well. As a result, I have always been highly supportive of projects that contribute to various services such as the National Parks Services. The idea of combining both CS and History to create a product that can be used to help teach people across the world about American historical features naturally drew my attention. This project allows me to explore further into VR technology and the game development process while allowing for the opportunity to contribute to the preservation of historical monuments.

Phillip Zou My immediate interest in the project was the combination of a project based in history with a STEM solution. I felt it would provide an interesting application of the skills we have learned in a field that is often not talked about in our curriculum. This was also an opportunity for me to experience and promote a national park. Additionally, building something educational that is also interactive and interesting poses a unique challenge that piques my interest. I hope to create something that will be educational but fun and engaging in order to encourage awareness and interest in our national parks. Finally, I wanted to try doing some work with 3D modeling and working with Unity since I have not had an opportunity to learn them so far.

3.1 Goals and Objectives

- The main goal is to recreate Castillo de San Marcos in both a desktop and VR experience.
- Other Goals:
    - General surrounding scenery (view of the water, Anastasia Island, lighthouse, city)
    - Visual and audio enhancements (sea birds flying around with audio, flags, etc.)
    - Inclusion of display boards and interpretive materials – these would be interactables
    - A map for the user to open to show the fort so they can navigate
    - Interactables
        - Interaction with cannon
            - Prepping/Loading cannon manually
            - Possible game/sharpshooting
        - Furnace, spews fire/sparks
    - UI to change the skybox for different weather
    - Past versions of the fort
    - NPC’s
    - Controller Support
    - Voice acted historical information blurbs
    - Visual/Audio Tour
- Objectives:
    - Create a realistic model of the fort

    - Create a game using Unity to allow users to tour the fort
    - Create an avenue for the National Parks Service to help educate guests about Fort Marion
- Function of the Project:
    - The main function of the project is to educate the public on Fort Marion. In addition to this, the project is to also have certain objects and minigames. These will assist in keeping the project more entertaining and keep the attention of any guests who may want to experience it.

4. Criteria and Constraints

- Design Criteria
    - The project must be VR compatible
    - The project must have Mouse and Keyboard compatibility
    - The project must have a historically accurate model of the fort
    - The user must be able to interact with basic fort features
    - The user must be able to adjust, load, and fire the forts cannons
    - The user must be able to access various historical prompts
- Constraints
    - Commercial Constraints
        - Budget must not exceed **$299 dollars**
        - Project must be completed before Fall 2022
    - Non-functional Constraints
        - Input system determined by what peripheral the user is using
        - User should not experience interruptions in gameplay

- Compliance Constraints
    - Refer to the Legal, Ethical, and Privacy Issues section
- Sensory Constraints
    - Fort must accurately represent the real life blueprints
    - The forts surroundings must be accurate to real life surroundings
    - User must be to scale compared to the fort

5. Broader Impacts

Our project has wide reaching applications in the realm of education on both a local and global level. Locally, this project provides an interactive, in-depth educational experience of local history. Since it is virtual, it provides local schools and programs a way to access the national park without dealing with the logistics of making a trip to the physical park. With further development and interest, the project may be expanded, allowing for customizable tours and experiences which in turn can help the end user have a more fulfilling experience with the park.

For example, schools or families could have virtual field trips, or the tours could be provided in multiple languages and formats. This in turn would foster interest in preserving and protecting our national parks and resources as well as generating potential tourism for the state of Florida.

On a larger scale, hopefully this project will spark interest and further development in creating virtual reality versions of services and parks provided by the local and federal government such as zoos, museums, other national parks, etc. These virtual museums and parks could then be accessed in multiple languages, allowing for other cultures and countries to experience American culture and history. If other countries were to create their own virtual tours this could provide a means for virtual cultural exchanges, broadening the accessibility and drastically lowering the cost for interested parties. Furthermore, this allows the US National Park Services to preserve a virtual version of this park and many others, which gives us a record for future

generations and researchers, which would be invaluable in case they are lost, damaged, or destroyed in the event of a natural or man-made disaster.

Since the goal is to have a low cost and accurate virtual representation of the national park, it provides a more accessible platform for improved education. A virtual reality national park has the potential to impact many under- represented groups. A virtual park allows those with mobility disabilities to experience national parks of natural or historic importance that they otherwise would be unable to visit. Furthermore, the ability to give tours in a variety of languages, in text and/or through audio/visual, drastically broadens the availability of the national park/museum to minority groups. Having tour guides who can speak or sign multiple languages would be cost-prohibitive. However, a virtual, customizable experience will reduce these costs to something much more manageable allowing for the national park services to deliver a satisfying, engaging, and educational experience to a much wider audience. Additionally, a virtualized national park allows for users to visit said park for a much lower cost, both financially but also environmentally. A virtual trip will not have the same environmental cost as making a long distance trip to the location. Hopefully, this project will encourage engagement in the end user, providing education to a broader audience and fostering public and government interest in having other parks and museums virtualized. This would not only provide more jobs for those working in STEM fields, but also provide exposure for the application of virtual reality as an educational tool.

6. Legal, Ethical, and Privacy Issues

Designing the Fort itself in Virtual Reality does not pose any legal, ethical or privacy issues as the blueprint of the fort is already given and we are allowed to take photos of the fort. Adding interactions with the cannons do not pose any issues as well, as long as it is not too violent for the intended audience and not designed so that users are directed to aim for a specific group of people which may incite hatred and division. Creating NPCs (Non-player characters) could become an issue if we attempt to recreate people from the past, as they may be inappropriately or incorrectly depicted. To address these issues, we could design the cannons so that users are prompted to aim for specific buckets and design the NPCs as random people that would look like people from the

relevant time period and not have them make extreme actions or say extreme phrases.

For legal items, Dr. Giroux informed us that she typically has the student teams license the software they create under MIT licensing, so it will most likely be the license we will be under as well.

Some general issues with VR environments include individuals engaging in indecent exposure, virtual groping, and committing tort (e.g. negligence, invasion of privacy, intentional infliction of emotional distress). In our project this may occur if we allow complexity such as real time interaction with other users by using their avatars, voice chatting, texting, and so on. To avoid this we can simply disable all interaction with other users. Other ways to address this, while allowing interaction, would be developing algorithms to automatically censor inappropriate language. In text it would be censored the moment right before it is sent, and in voice chatting perhaps we can implement a second of delay to process and censor inappropriate words and phrases but this may be unrealistic as a second delay in voice communication may be irritating for users that want real-time communication and users could avoid this censorship by speaking slowly or spelling out characters. To address this, users may be granted a block and a report option where the block option completely disables specific users from interacting with the user and the report option allows us to review the specific users' activities and decide whether to warn or ban the users from using our program.

Another issue with this environment is that if we do ask the users for some basic information such as their email addresses and passwords, there is a potential risk of their private information being stolen by hackers. To avoid this, we could either design the program so that no users will need authen- tication. If we ask for their information, we could use popular and/or trusted encryption so that not even we can fetch their raw information from the database and so that it will be difficult for the hackers to steal. If they manage to break through, the best we can do is to warn users in the database of the situation and change our encryption method.

7. Requirements

7.1. Performance

- An average of 90+ frames per second
- Poly count optimization with Level of Detail (LOD) rendering
- Maximum of 500 to 1000 draw calls per frame
- Maximum of 1 to 2 million triangles or vertices per frame
- Baked lighting and ambient occlusion
- Texture compression
- Unity culling

7.2. User Experience

- 1:1 scale 3D replica of the fort based on surveys
- Free roam
- Weather-relative skybox
- Informative text descriptions of historic features
- Interactable map for navigation
- Populated environment with surrounding scenery
    - Anastasia island
    - St. Augustine Lighthouse
    - Matanzas river
    - Location-related foliage
    - Rocks

    - Local bird species flying around
- Fort design & staging
    - Coquina wall textures subsampled from 4k images
    - Storage crates and Barrels
    - Stairs
    - Terreplein
    - Ramparts
    - Furnace spewing sparks
    - Room exhibits
- Interactive prop placement
    - Muskets [inspectable]
    - Clothing [inspectable]
    - Bronze and iron cannons [multi-stage prep/load/fire]
    - Drawbridge [toggleable]
    - Doors [toggleable]
    - Window shutters [toggleable]
    - Sallyport closures [toggleable]

7.3. Compatibility & Accessibility

- Support for Oculus Quest 2, Oculus Rift, Valve Index, and HTC VIVE
- Non-VR version for PC running on Windows
- Configurable options
    - Audio settings

    - Control mapping
    - Graphics
- Controller support for PC
- Text prompts offer voice-overs or text-to-speech.

8. Ideas

8.1. Sterling Downs

- JIRA to help organize a roadmap and assign tasks for the project as well as following AGILE methodology with Scrum sprints.
- Existing asset packs for props and foliage to populate the scene and reduce the total amount of assets that need to be created via 3D modeling.
- Lay framework for managing commonly used components in the scene.
- Take pictures to be applied as textures and use photogrammetry techniques to better replicate important aspects of the fort in a virtualized 3D environment.
- Test the product periodically on different platforms to ensure compatibility and that target performance metrics are being met.

8.2. Landrey Martin I would love for us to dedicate some time to add as much detail as possible to the surrounding areas of the fort, as well as an option to display the fort under different conditions, such as different times of day and weather conditions. Something I would like to implement is a menu to allow users to select how they want the weather/time they want to experience during their simulation of walking through the fort. This menu would allow the user to select beyond a default perfect weather condition and time of day experience. It would be interesting if the user could select the current weather and time at the fort, which would require us to track the weather status within the area.

8.3. Howard McDavid One idea I have had for this project includes additional exploration of areas around the fort, but not in the immediate area. This would include the ability to travel to the lighthouse, climb it, and turn the light on and off.

Perhaps, this could be implemented with a night mode, of sorts. This mode would allow the user to view the fort with period lights as it would have appeared after sunset. With this, the user can turn on the lighthouse and it will spin until the user turns it off. It would allow the user to see the locations of any ships they may be firing upon in the cannon minigame.

8.4. Juhwan Park It would be exciting to make it massively multiplayer online so that users can see who is online and what people are exploring in real time but this may be a very unrealistic goal as hosting servers may be costly and unnecessary as there would be very few returning users once they have explored all features and without much advertisements, the player base may be rather small as well. Some of the objects such as buildings and roads outside the fort could be roughly developed so that when looking out the fort it would not be a simple plane of nothingness. We could add some detail for the grass and also have the sun and moon slowly moving in and out of vision depending on the time of viewing the fort.

8.5. Enrique Rodriguez Currently, we are implementing a cannon minigame where the user can adjust, load, and fire a cannon to aim at a target/ship, however, it would be interesting to enhance this feature even further making it into a competitive game with leaderboards/PvP. Both players would be situated on ships and would have to fire at each other, having to take into consideration travel speed of their ship and bullet travel of the cannon ball to win. I also was thinking about expanding this project by adding additional forts as part of a national parks project but that would involve numerous groups across the US.

8.6. Phillip Zou Since this is an educational project, I believe accessibility is key. I would like to add an audio/visual tour, with multiple languages as a stretch goal. I think that adding a local multiplayer mode as well as an online peer-to-peer mode will also allow for people to engage and experience the park with a party of their choosing, so they can visit with their family or classmates. It would also be interesting if we could have simulated typical days at the fort as well as progressive aging on the walls to show how the damage has affected them. Many of these suggestions run into the problem of budget and scope, however I believe they work well as stretch goals, or perhaps as a future iteration of the project for another team.

9. Division of Labor

9.1. Sterling Downs

For this project, I will be working on the codebase as well as designing and testing interactables and populating the scene with various props. For the codebase, the goal is to keep it clean and maintainable. I plan to use modular code with design patterns as needed and build a strong solid foundation for the project early on. It is also important that performance requirements are being met, so I will be designing systems to obtain and keep track of performance metrics over the course of our development. In regards to interactables, they need to be designed in such a way that is consistent with their real-life counterparts and function in a similar way. The goal is to create an experience that the user can enjoy, so if needed, some quality-of-life changes will be added to either reduce the complexity of an interactable or make it more forgiving. The design of the interactables will be done with Blender and then exporting the model into Unity. In addition to this, various props will need to fill the scene to give it a more immersive feeling. Walking into an empty room is not an enjoyable experience. Therefore, I will be helping to populate these rooms and depict them in such a way that is consistent with how they might have looked during that time period. As the project progresses, I may take on additional responsibilities as needed.

Cannon Models

The Castillo de San Marcos fort features a wide variety of cannons and mortars throughout the premises. The most common cannons include the 6-lb and 8-lb cannons, which will be the main focus for cannons included in the project. While the interactions and procedures associated with these cannons are identical, their appearance is not. The 6-lb cannon and its related cart are smaller in size than that of the 8-lb cannon. Additionally, upon visiting the fort, it was noted that the 6-lb cannon has a smooth surface, while the 8-lb cannon appeared to have a hammered and worn appearance to it. While these differences may be a result of preservation applications, we will be creating the associated models as they appeared upon visiting the fort.

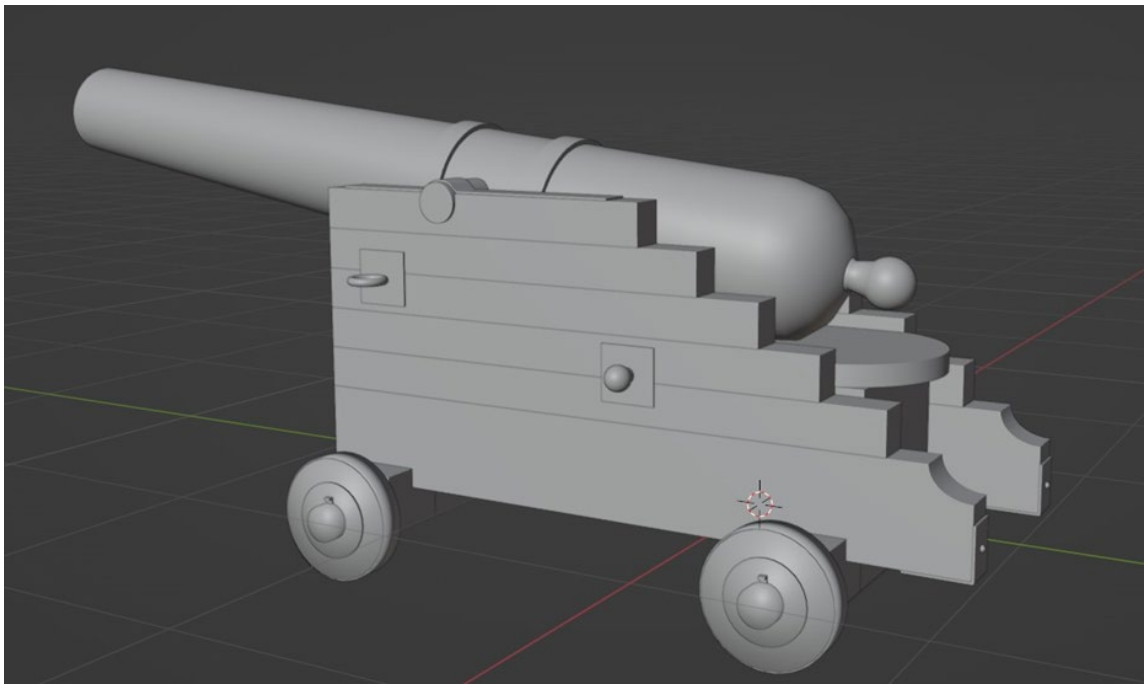


Figure SD.1: 3D model mesh of 6-lb cannon designed in Blender

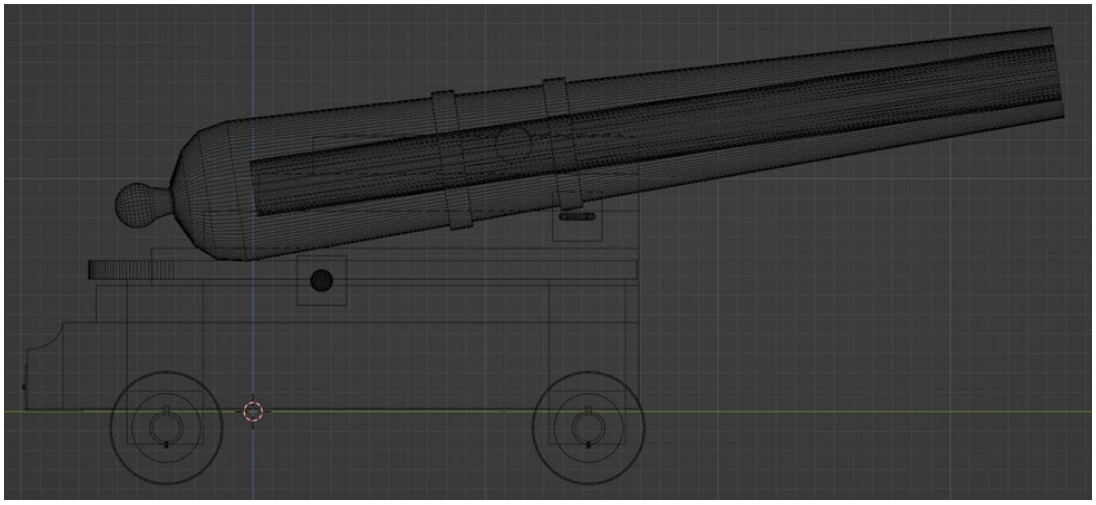


Figure SD.2: Wireframe of 6-lb cannon showing edges and inner-structure

The above images of the mesh and wireframe of the cannon shows the shape of the cannon as well as its inner-structure. The entire mesh is made up of approximately 16,000 triangles and is designed in reference to the 6lb-cannon our team saw when visiting the fort. The mesh still has room for some optimizations as needed, but the overall model has been designed in such a way that allows changes to be easily made if necessary.

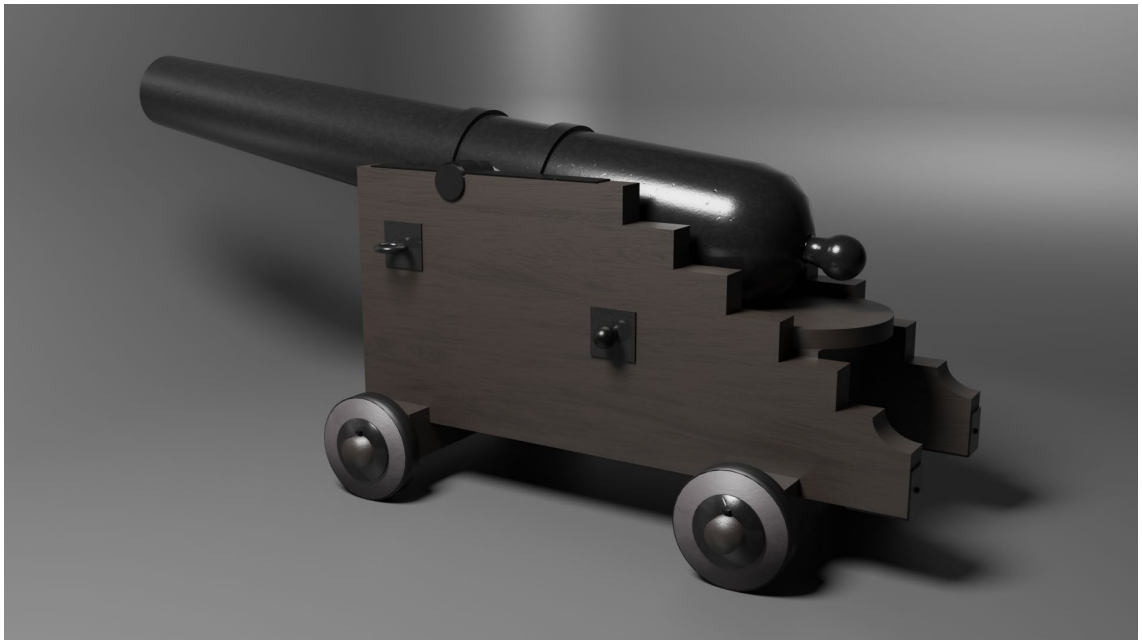


Figure SD.3: Various materials applied to cannon mesh (not final)

From the overall mesh of the 3d model, we can then apply materials to the individual meshes that make up the cannon. These materials are created in Blender's in-built material system and shading workspace that includes a node-based system for combining and fine-tuning the overall texture and material. While still a work in progress, the cannon shown above uses a combination of 8 different material

Using the node system, we can fake details at **no extra cost** to our model. This technique is used to add scratch details to the cannons. A bump node is used in conjunction with a noise and wave texture to generate scratches in a pseudorandom way. Additionally, a color ramp from black to white (or vice-versa) can be used in combination with the outputs of these textures to make the details appear in separate batches rather than constantly being spread throughout the material.

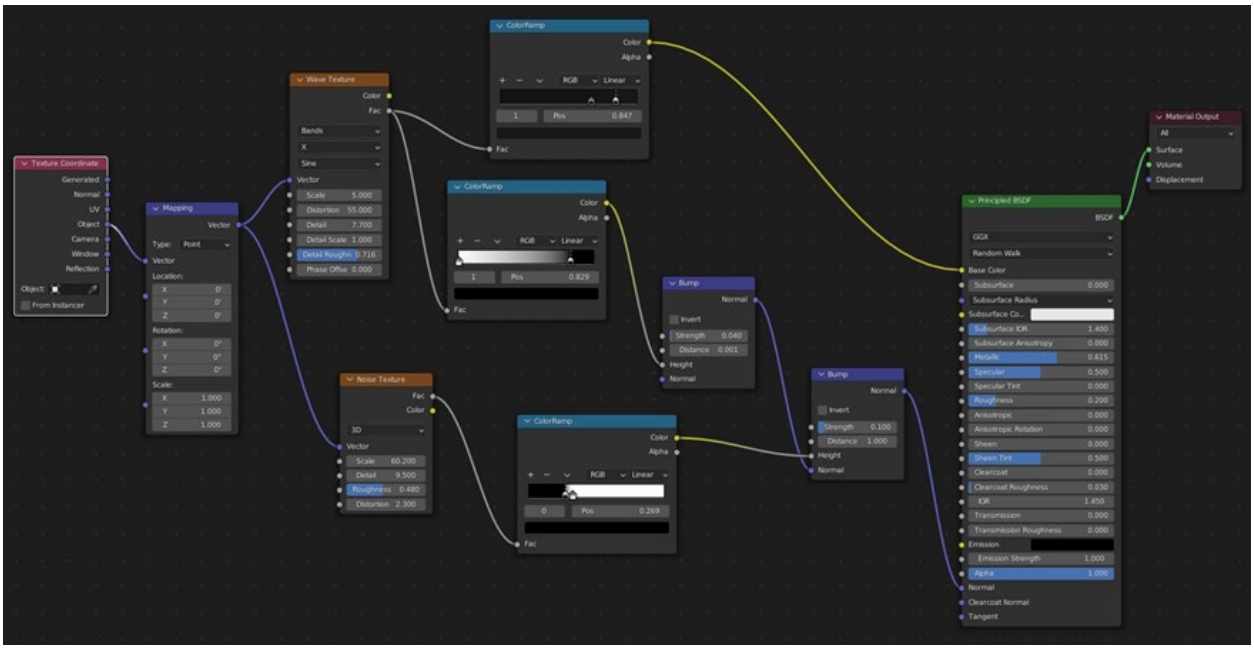


Figure SD.9.4: Material for cannon barrel defined with Blender's node-based system

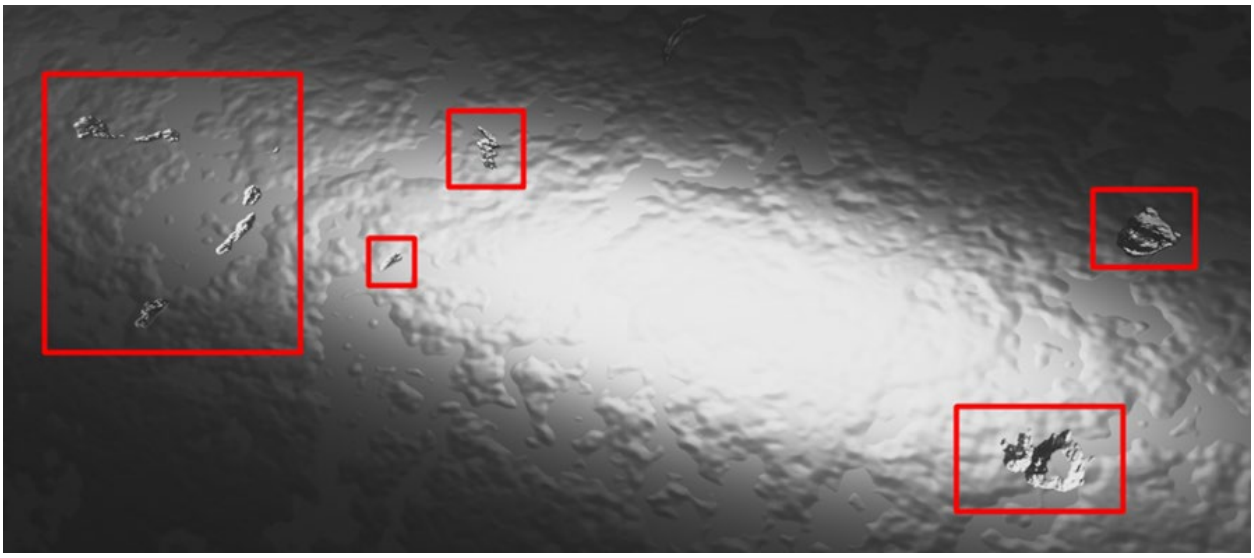


Figure SD.9.5: Resulting scratch details with textures and bump mapping

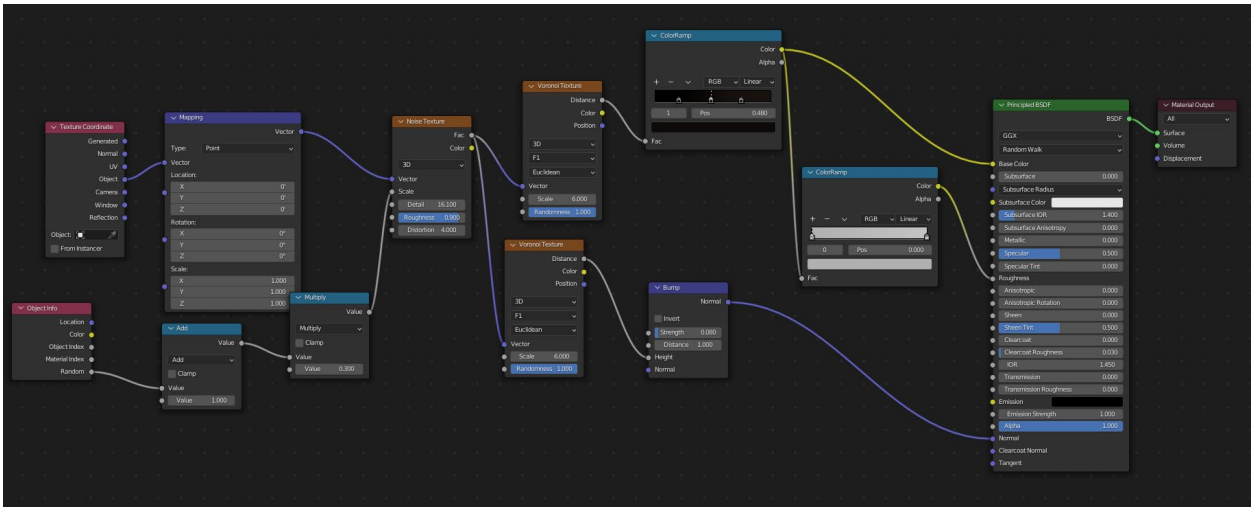


Figure SD.9.6: Material for cannon cart wood defined with Blender's node-system

Figure SD.9.7: Resulting procedural wood-grain material

The individual cannons also require different size cannonballs as follows:

Cannon Size	Cannonball Diameter
6-lb	3.49”
8-lb	3.95”

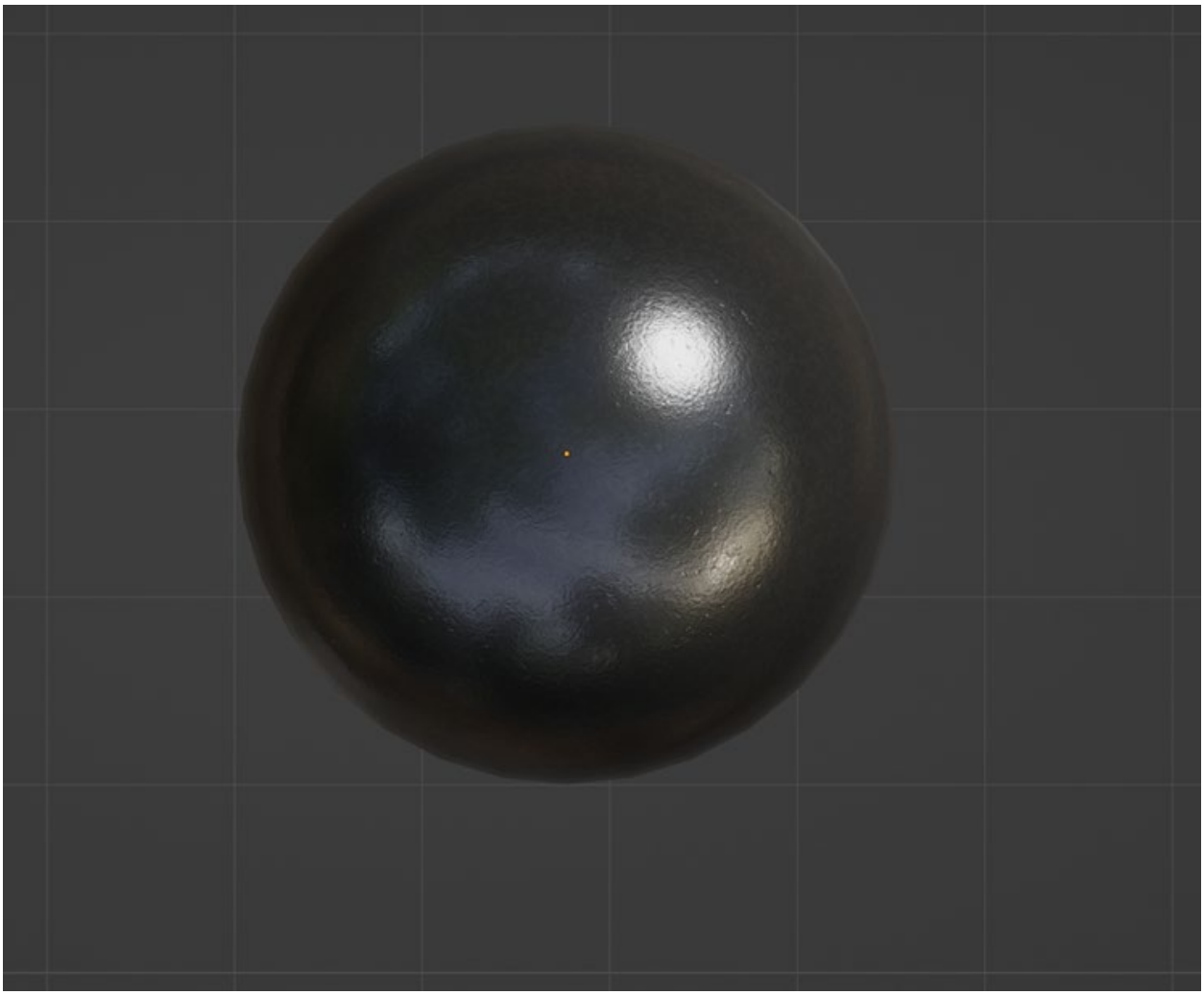


Figure SD.9.8: 6-lb cannonball model designed to-scale

Cannonballs must be designed to match the specifications of their respective cannons. For instance, an 8-lb cannonball will not fit in a 6-lb cannon and while the opposite is not true, it is not practical. Thus, the different cannonball sizes will be restricted to only interacting with their corresponding cannons. Furthermore, when the model is imported into the Unity scene, each type of cannonball will be assigned individual simulation weights as mentioned in ***Simulating Weight For Interactables*.**

9.2. Landrey Martin

My role on this team is mostly focusing on the environment, as well as some modeling. I have researched and found resources for several environmental factors, as well as the best ways to implement them into our project. I got to look into skyboxes and what our best option is between finding an asset versus creating our own out of images we took while visiting Castillo de San Marcos. Another thing I was responsible for was looking into implementing birds into the project that looked and acted as realistic as possible. I also looked into adding water to the project, as the fort is located on the bank of the Matanzas River.

I was able to model two objects for this project. One was a small barrier/fencing located inside a room, which was used to separate guests and a display area. The larger item I modeled was the flag and flagpole that are located on the top of the fort. I had to learn how to implement wind forces into the model in order to achieve a realistic flag motion.

Finally, I was responsible for researching and summarizing Castillo de San Marco’s history. Since the structure is on display as a very important historical monument, it’s important for anybody involved to know why it is such an important place. By knowing its history from many influences over the years, we are able to understand some of the reasons as to why some aspects of the fort look different from others.

Gate Models

For our current day simulation of Castillo de San Marcos, there are several areas that are blocked off to allow visitors to see props/ artifacts from the past in the fort, but not interact with them. One way that Parks Services has been able to accomplish this is by creating small fences and gates as barricades. To display those barricades in our simulation, we had to model it as an asset using the image displayed as figure LM.1 as a reference.

Figure LM.1.1: Image used as reference when modeling the gate and fencing.

As seen in figure LM.1.1, we did not manage to get any dimensions to get the most accurate recreation of the gate or fence. Through using the wooden planks on the floor and the width of the room, I managed to get a rough estimate on the sizes I needed to make the objects. And come up with a good resulting model.

I decided to use AutoDesk Inventor to design this model since I have prior experience designing with AutoDesk software. The UI and designing process was still familiar to me, even though it has been 5 years since I have last touched Inventor. It did take a little bit of time to get my mindset back into the best way to approach the model. However, once I got back into it, I was able to finish the model very quickly.

I approached the model in three parts. The first part was the gate piece in the middle, which appears to open if provided force. I decided to create the gate as a standalone piece since it will be the only moving piece in the model. The second part was the fencing to the left and right of the gate, which both appear to be two identical panels. I decided to create one model and duplicated it for each side of the gate. Once I had the model created for the gate, as well as the fence, I had to tackle the third part. This was the most difficult aspect of the model. I had to apply joints and constraints to allow the gate to open 90 degrees forward, towards the back wall of the room.

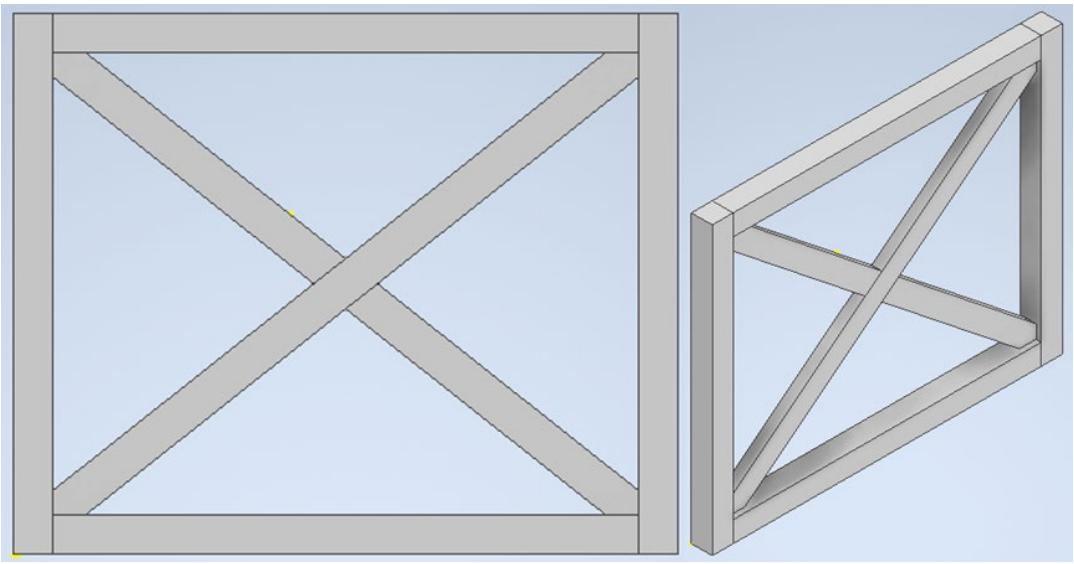


Figure LM.1.2: Front view of the gate

Figure LM.1.3 Angled view of the gate

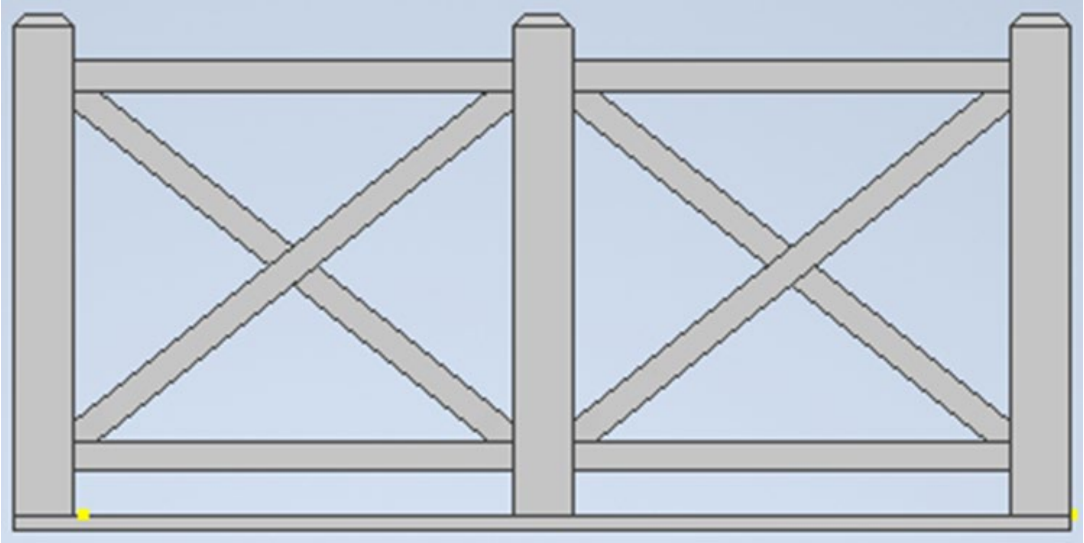


Figure LM.1.4 Front view of fence

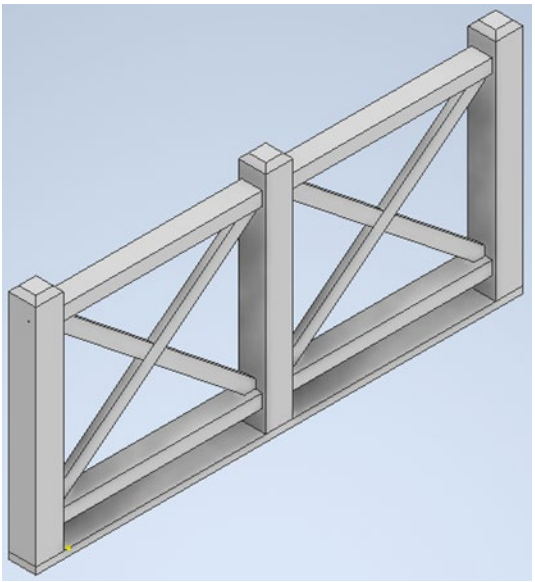


Figure LM.1.5 Angled view of fence

Implementing the joints and constraints proved to be a massive challenge for me. The gate piece wasn't as wide as the large pieces of wood on the edges of the fence piece, so I had to find a way to ensure the gate was centered along that large piece on the edge of the fence. I also had to find a way to ensure the top and bottom of the gate piece aligned with the nearly matching parts that were on the fence. After hours of tinkering, I ended up adding an extra extrusion to the fence piece that would provide a perfect alignment for the gate. Once I had it all aligned

nicely, I was able to cut the extra piece out, so it didn't affect the look of the final model. Once the gate was aligned as needed, I had to address the opening of the door.

There is a joint option that is called "rotational", which allows you to select two faces or vertices and will connect them in a way that will allow the object first selected around the second selected point. This feature displays a small image when you hover over each point, but before you select, to show the designer which axis the point that is being hovered over will rotate on. This allows you to select proper points to ensure the object rotates either on the x axis (horizontally) or the y axis (vertically).

Once I was able to apply the rotational joint accurately, two small issues arose. One issue was that since the edge of the gate was constrained to the edge of the fence, it would rotate through the fence. To prevent this, I added an offset to the constraint just wide enough for the gate to open without hitting the fence piece. The final issue was that the rotation went 360 degrees around. There is a very setting within the joint menu to specify the start and ending angular rotation. Once set, the gate will now only open in one direction and will only rotate 90 degrees.

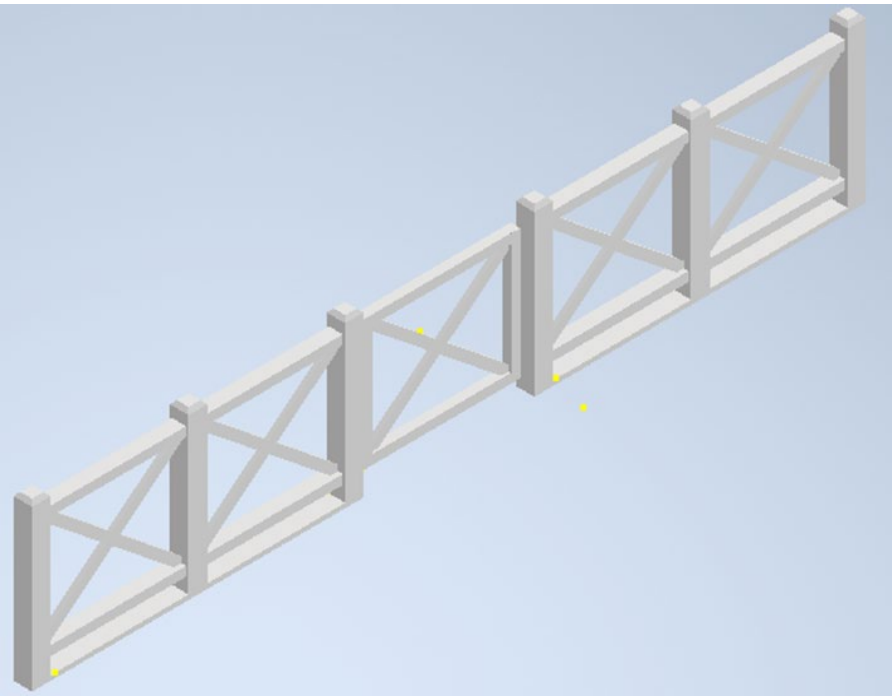


Figure LM.1.6 Fully complete model of barricade with gate closed

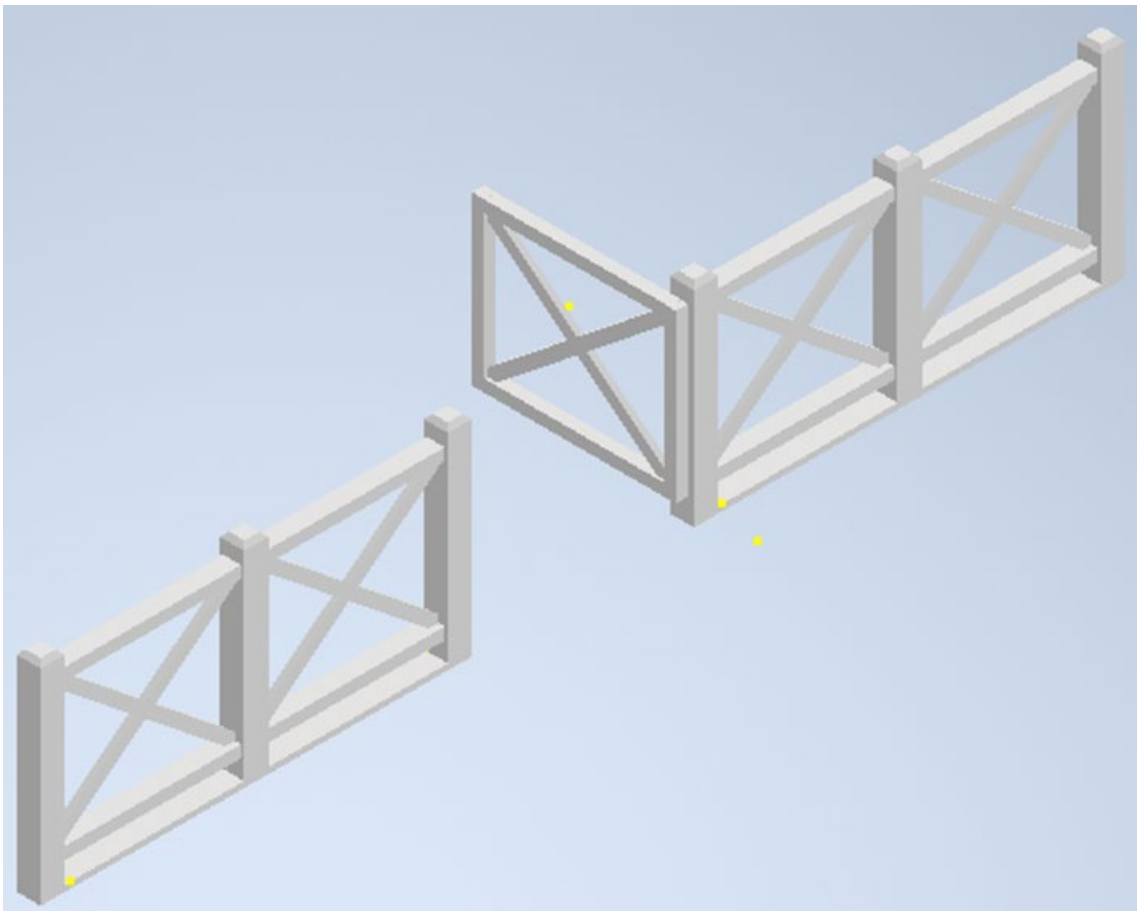


Figure LM.1.7 Fully complete model of barricade with gate open

Flag

I am responsible for creating and implementing a realistic flag into our project. I had to look at several tutorials and options as to how to get it working, but finally landed on using Blender to create it. This was my first experience with Blender, however, I was able to find several very helpful resources to help me understand how to find settings and do certain things. One of the first things I did was ensure that the measurements were in inches, shown in the menu displayed as figure LM.1.8. I chose inches over feet since it's the more precise option and most measurements I need are not large.

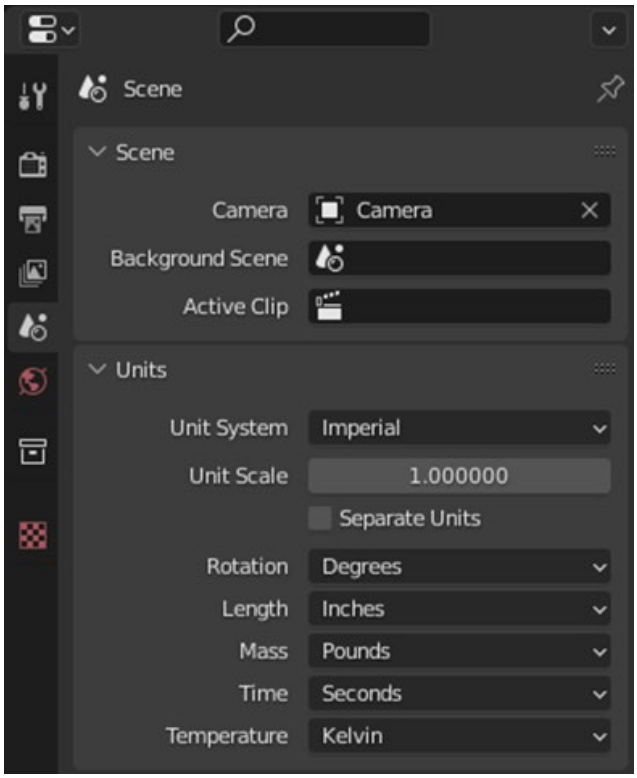
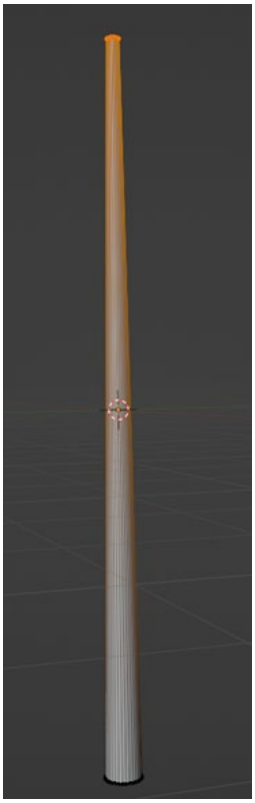


Figure LM.1.8 Scene settings allow you to modify which unit system you use for the project

Looking at images online, I can gauge roughly how tall and wide the flagpole is, as well as the flags dimensions. The flagpole itself is a very odd shape, it is cylindrical but is significantly narrower at the top. By the looks of it, the base is at least a foot wide in diameter while the top is roughly 4 inches in diameter. The height was another object I had to guess based on seeing images of people standing next to it, I made the flagpole 20 feet tall. This is also a standard height for a flagpole.



To create the model of the flagpole, I first created a cylinder mesh and modified the dimensions to be 12 inches in diameter and 20 feet tall. Next was addressing the pole getting narrower at the top. I had to switch into edit mode and select all the vertices that I wanted to modify. Once all the vertices are selected, pressing the ‘s’ button will implement a scaling process. I scaled it to .33 since I wanted the top to be a third of the width in diameter from the bottom.

Finally, the flag itself can be addressed. First, I added a plane to the model. This had to be rotated 90 degrees to be oriented properly. I then stretched the plane to ensure the flag is the right size. I used a standard flag size of 3x5 feet long. Next, I lined up the plane, so it was in the correct positioning and not touching the flagpole. The resulting image is displayed in figure LM.1.9.

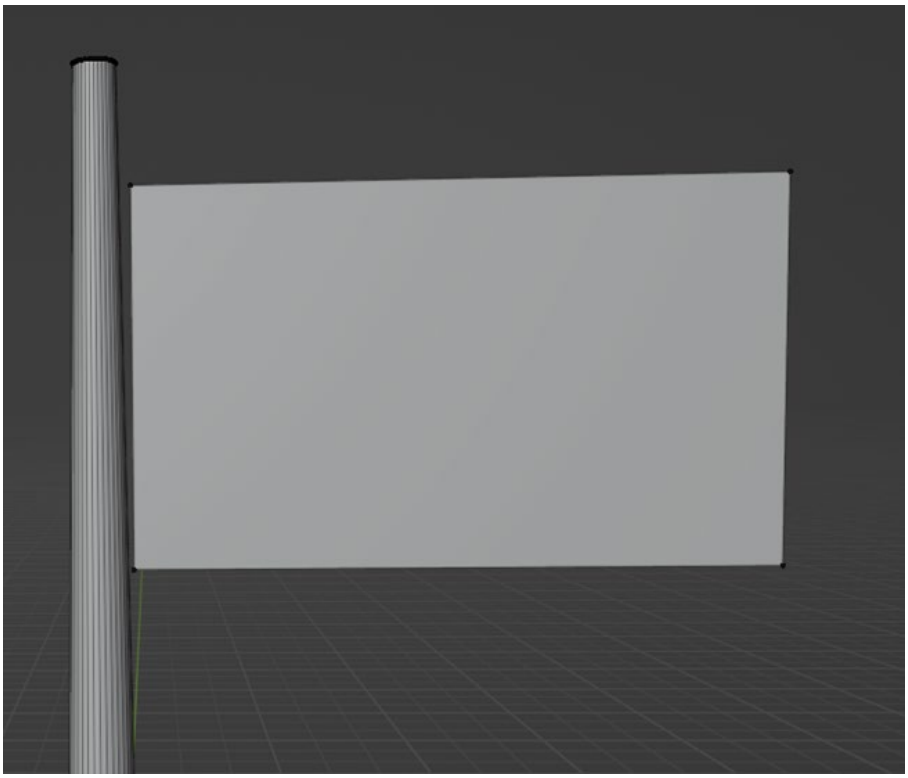


Figure LM.1.9 The plane is the correct size and positioned where it needs to be

Next, I subdivided the plane into 15 pieces to allow for plenty of movement of the flag. This doesn’t affect the physical appearance of the flag, it is a factor that will be affected by physics, which still hasn't been implemented yet. At this point, it’s necessary to add the cloth modifier to the plane. This will begin to display the physical aspects of the flag. Before doing anything else, the flag needs to be secured in some way because at this point it will simply fall to the floor due to the newly inherited gravity. To secure the flag, I will select both right corner vertices on the flag and ‘pin’ them into place. Figure LM.1.10 displays the flag with all subdivisions, as well as the vertices to be pinned.

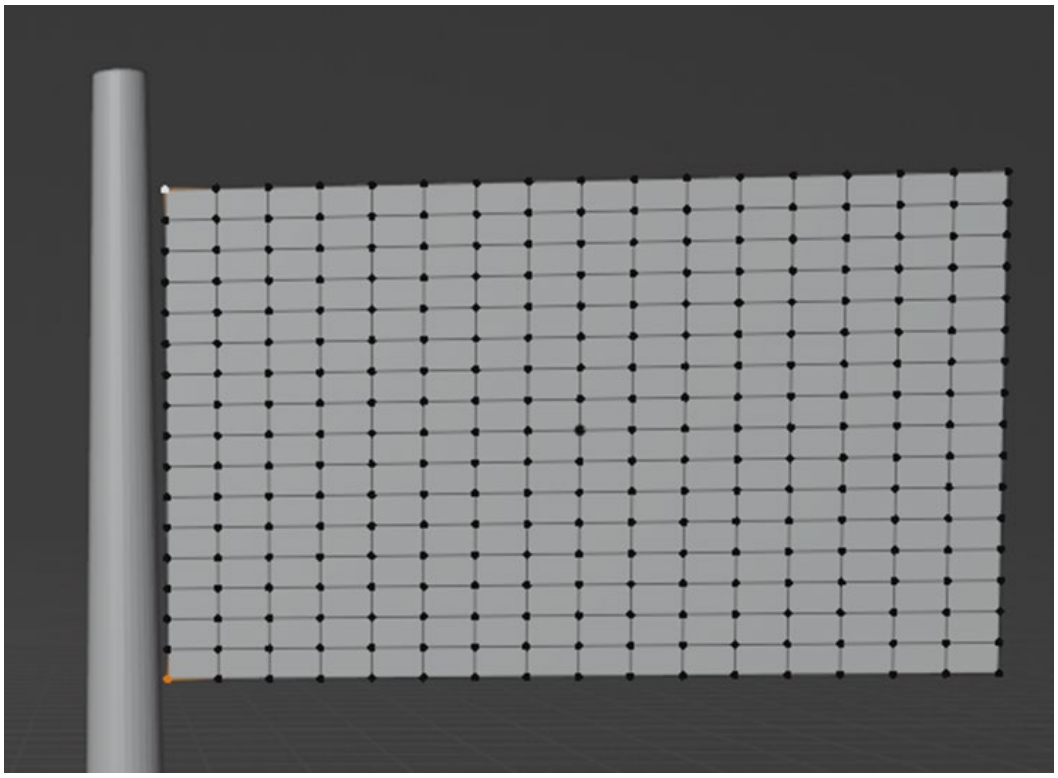


Figure LM.1.10 With all subdivisions displayed, the two used to secure the flag in place are highlighted

With the two pins in place, we can now play the simulation. There are still many factors to consider. First, the cloth will go through itself and the flagpole, which would require newly added collision constraints. Within the cloth settings, you can select “self-collisions” to prevent the flag from intersecting with itself. Then select the flagpole and add a new collision modifier. Once that is added, the flag will no longer intersect with the flagpole either. Figure LM.1.11 displays the flag at a resting state without collisions occurring between it and itself, as well as the flag and the pole.

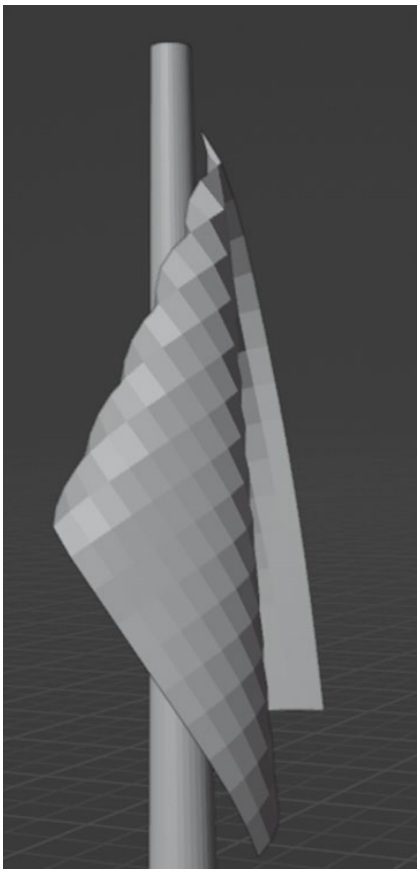


Figure LM.1.11 The flag resting after collisions are implemented

Now the next issue to address is the visual aspect of the flag. It currently displays as a non-smooth object, and it is obvious where each subdivision lies. This can be resolved by going into the object tab and selecting “Shade smooth,” which will resolve the subdivisions being visible. It can be made even more realistic with modifier properties added to the flag. Adding the “Subdivision surface” modifier allows for more smoothing on the surface. Modifying the render value can result in a more realistic looking movement, like how real fabric moves. This smoothed flag is displayed in figure LM.1.12.

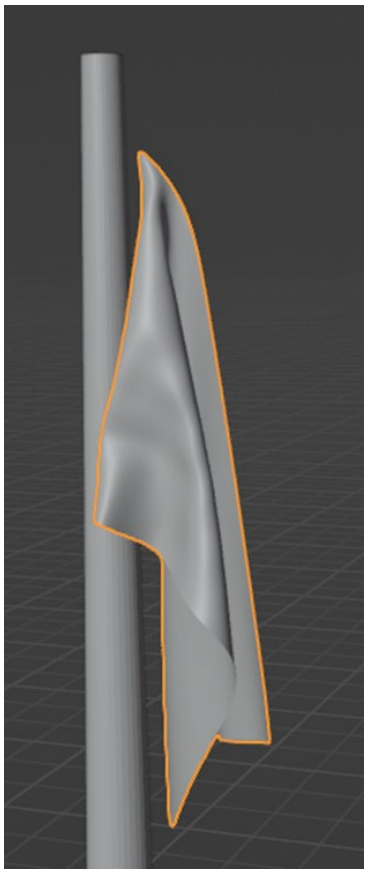


Figure LM.1.12 Flag resting after adding the subdivision modifier

Now since the only force acting upon the flag is gravity, the flag will only move downwards to a resting position. Adding a wind source will allow the flag to move as expected. This is a type of force field that can be added, and features can be modified as needed. This will need to be rotated and placed in line with the flag to ensure the wind moves the flag. After modifying the values of the wind strength and wind factor, I was satisfied with the movement of the flag, which can be seen in figure LM.1.13.

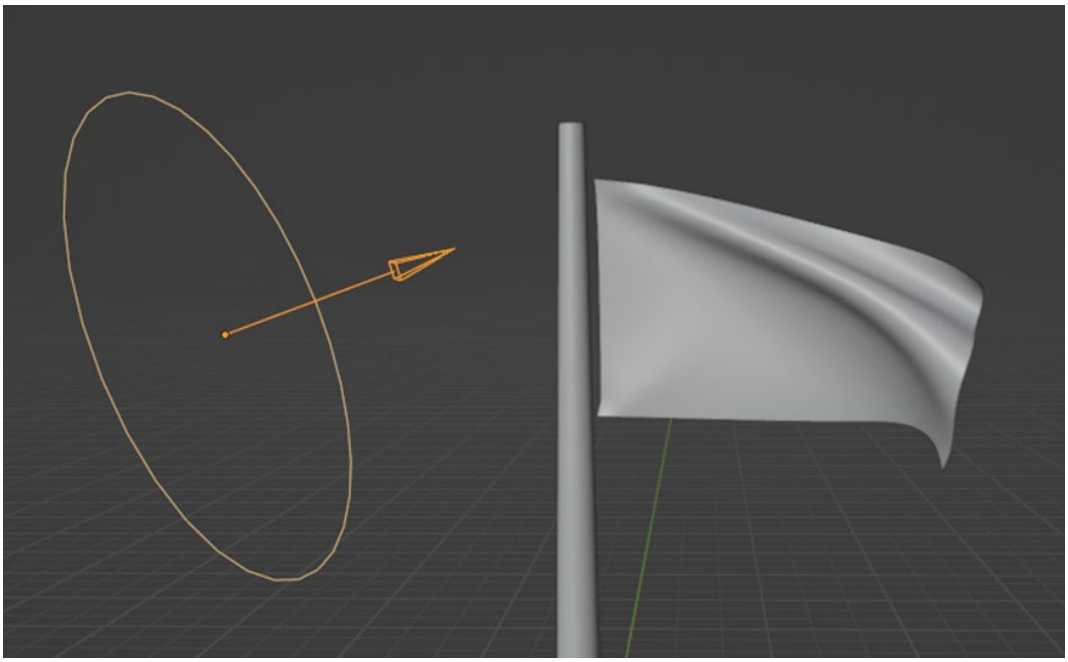


Figure LM.1.13 Still image of the flag with the wind applied

The last aspect to address is the texture on the flag. On most images, it appears that Castillo de San Marcos frequently flies the original Spanish flag, so I have decided to use that. I had to switch to the render workspace and add an image texture and plug it into the base color to get it to display the flag image on top of the plane. Figure LM.1.14 shows Blender's node-based system for textures.

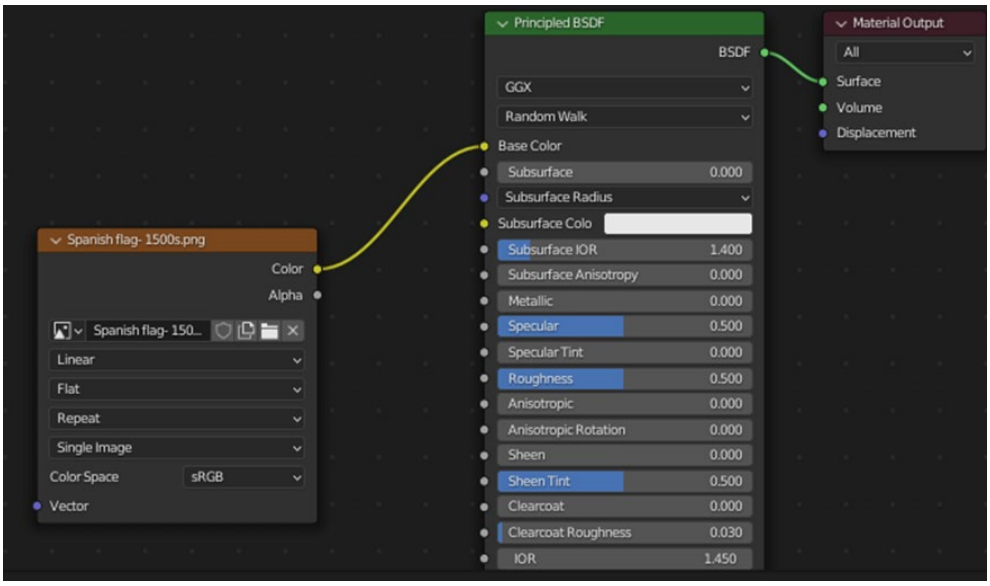


Figure LM.1.14 Rendering tab with image texture input

Switching back to the layout tab, the texture won't display unless the render view is on. The default view is solid mode, so it will appear without the texture until the view is switched. The render view is dark, so I added a sun source, which allows the flag to be easier to see and adds in shadows as the flag moves. Figure LM.1.15 displays the final flag rendering while it is flying with the wind.

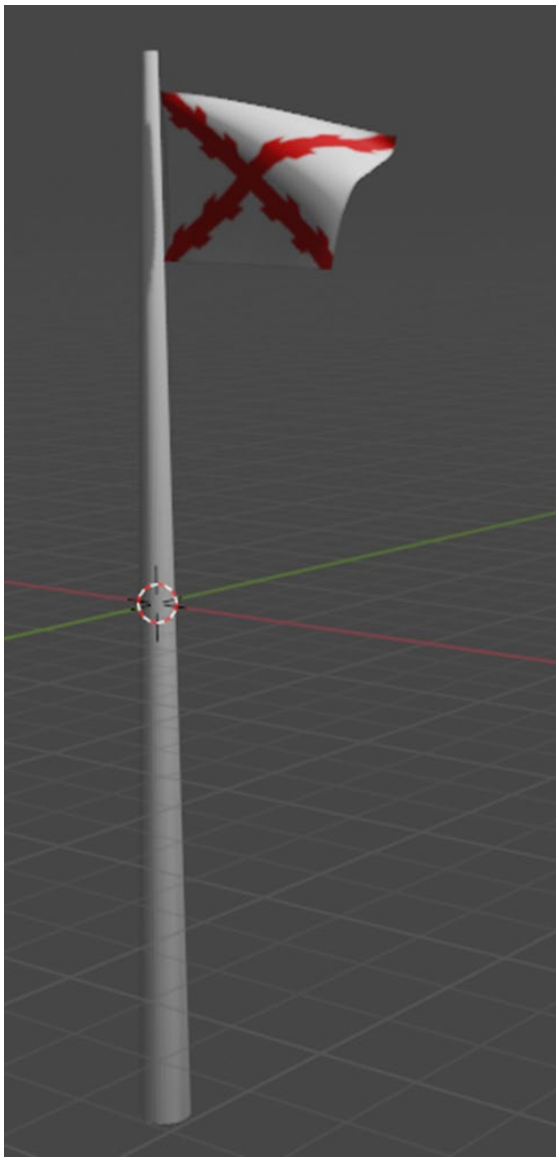


Figure LM.1.15 Finalized model of the flag

9.3. Howard McDavid

My role on this team is that of a 3D modeler. My modeling experience with low poly count, allowed me to take on the challenge of modeling the fort itself. At the very beginning, I was in the research phase. Through the help of our sponsor, we were able to obtain the drawings created by the Works Progress Administration when a survey was done in the mid 1930s. Though these drawings do give a good sense of what the fort is designed like, they simply don't do it justice.

Following the diagrams, I looked through Google Street View, which has pathing along the top of the fort and in the courtyard. Using that, I was able to get a good look at the fort without being there. Through Street View, I was able to see the fort's coquina rock structure and get an idea of how the model should look when complete. From there, I moved on to modeling.

Before visiting the fort, I created a basic 3D model of the fort without any doors, windows, or lookout towers. (Figure HM.1) The model, as shown below, was incredibly basic, only having the outside walls, courtyard, and the stairs leading up to the top of the fort.

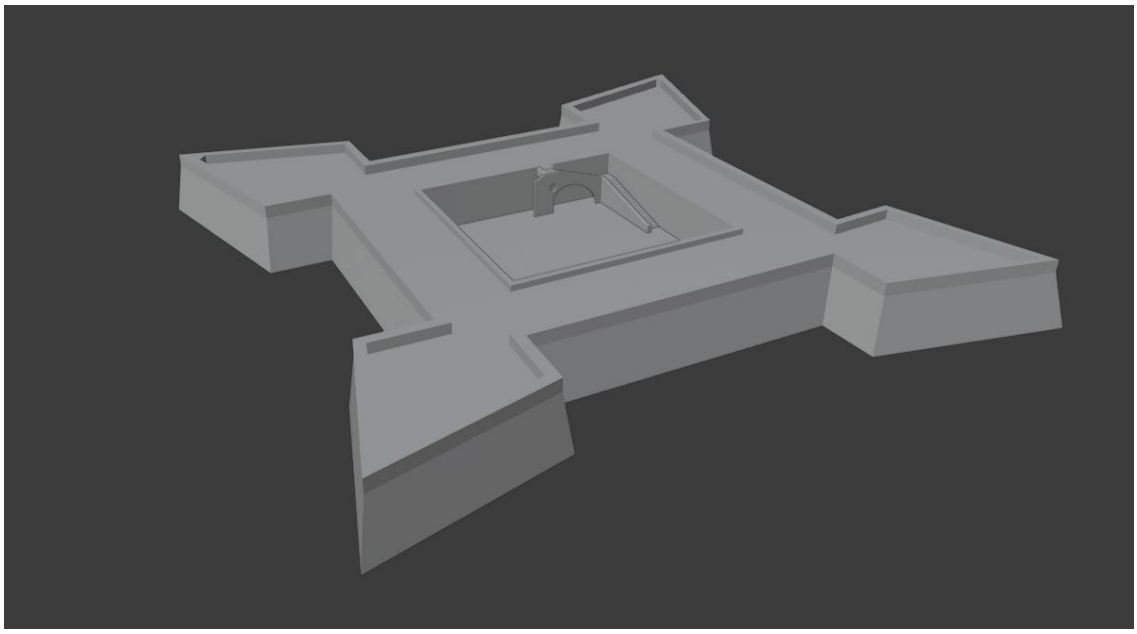


Figure HM.1: The original model of the fort prior to the visit

Following a visit to the Castillo de San Marcos, I had a much better understanding of how the fort was laid out and how to model it. I got back to work

on the model after the visit. Below, the fort is shown with the proper heights of walls, notches for the cannons, and the lookout towers. (Figure HM.2) In addition, the lip around the top of the fort is also modeled.

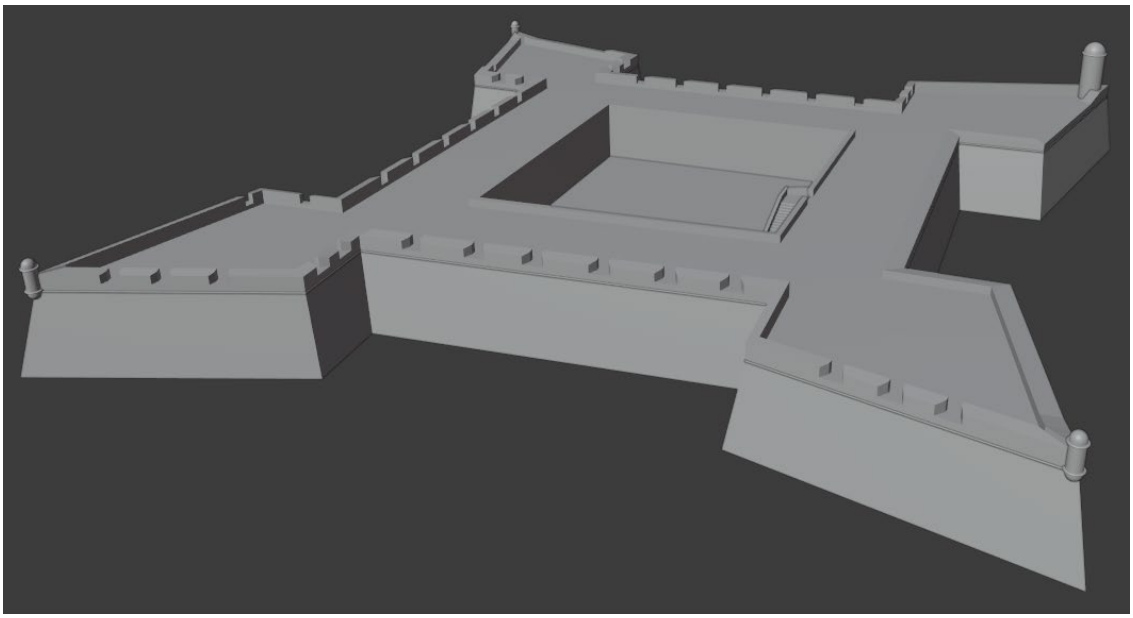


Figure HM.2: The fort model as presented after visiting the fort

Finally, this brings us to the next version of the fort. (Figure HM.2) I set to work on getting each room cleared in the model. This entails using the boolean tool to hollow out an area for the room. Below, the room models are half complete, with only the south and west walls remaining. From the image below (Figure HM.3), it can be seen that the corners of the model and the corners on the drawing don’t align properly. Based on this fact, it is likely that the drawing’s angles were not done to scale, which gives rise to such a large difference from the model itself and the drawing.. From here, the remaining rooms will be modeled, the doors between rooms will be cut, and the exterior windows and doors will be cut as well.

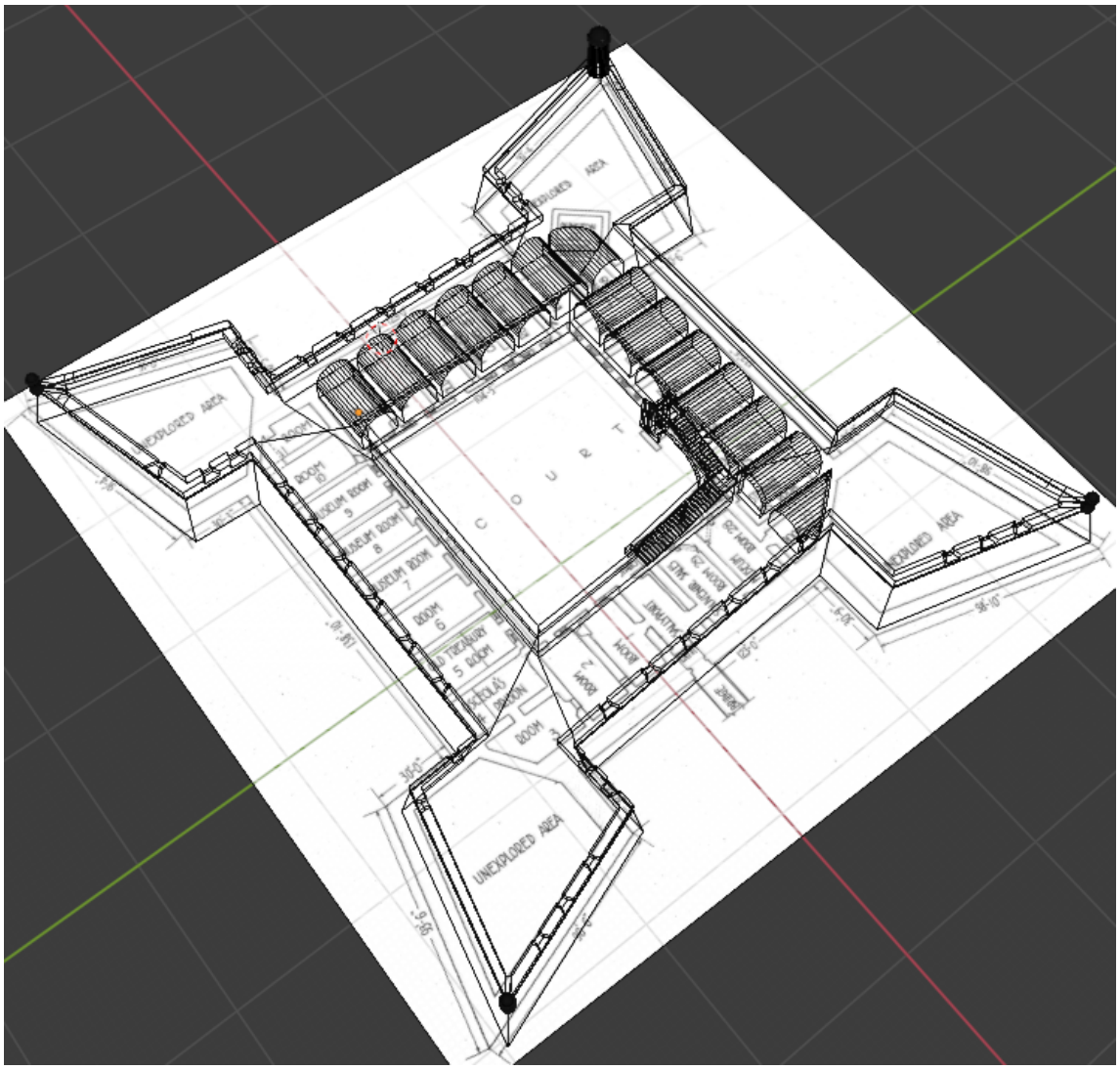


Figure HM.3: The interior room, half created, still missing a many rooms

From here, I simply continued creating the rooms as they are presented on the map. The next thing I did was finish the rooms as they are presented on the map shown above (Figure HM.3). Following the room placement, I went through and created each door and window, with the exception of the exterior door and the chapel door. With the door and window openings in place, I took screenshots and conducted two renders of the model. These images are shown below. (Figure HM.4-HM.6)

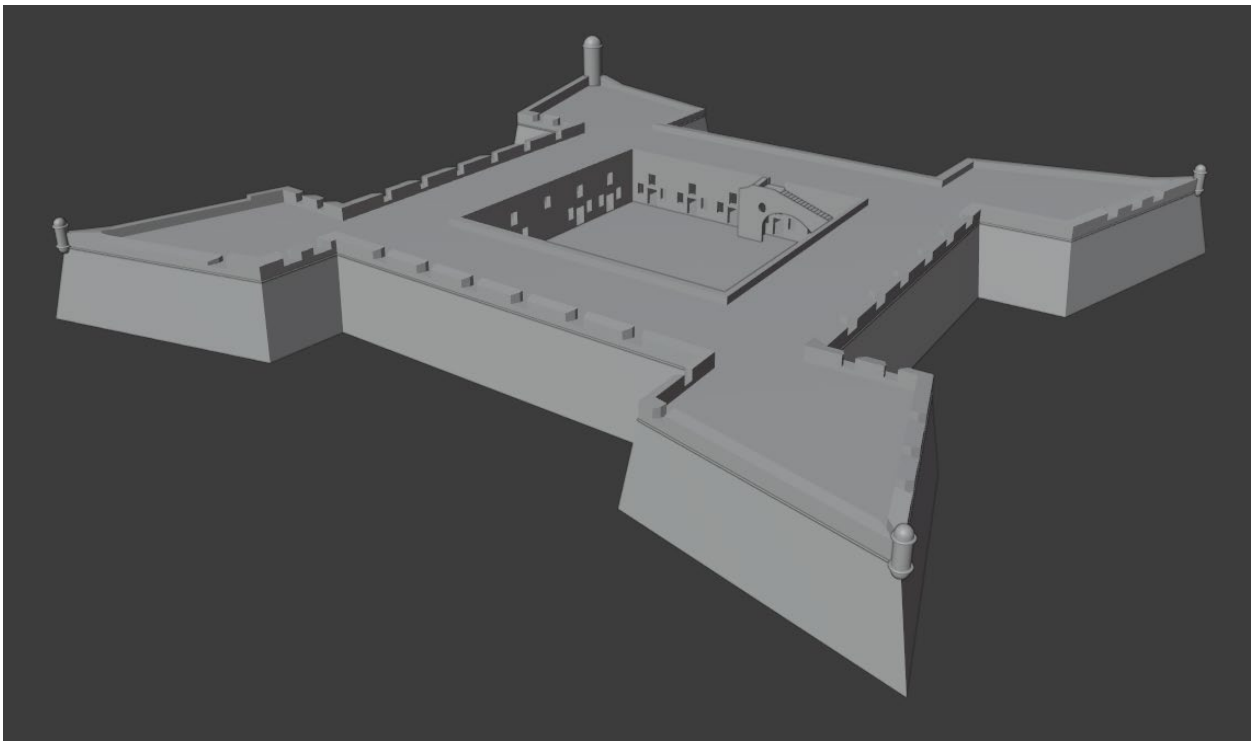


Figure HM.4: Unrendered model of the fort with many of the doors and windows present

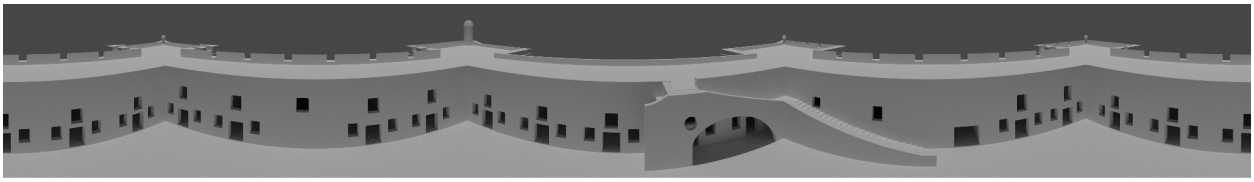


Figure HM.5: Panoramic render of the fort in the same configuration as the above unrendered figure

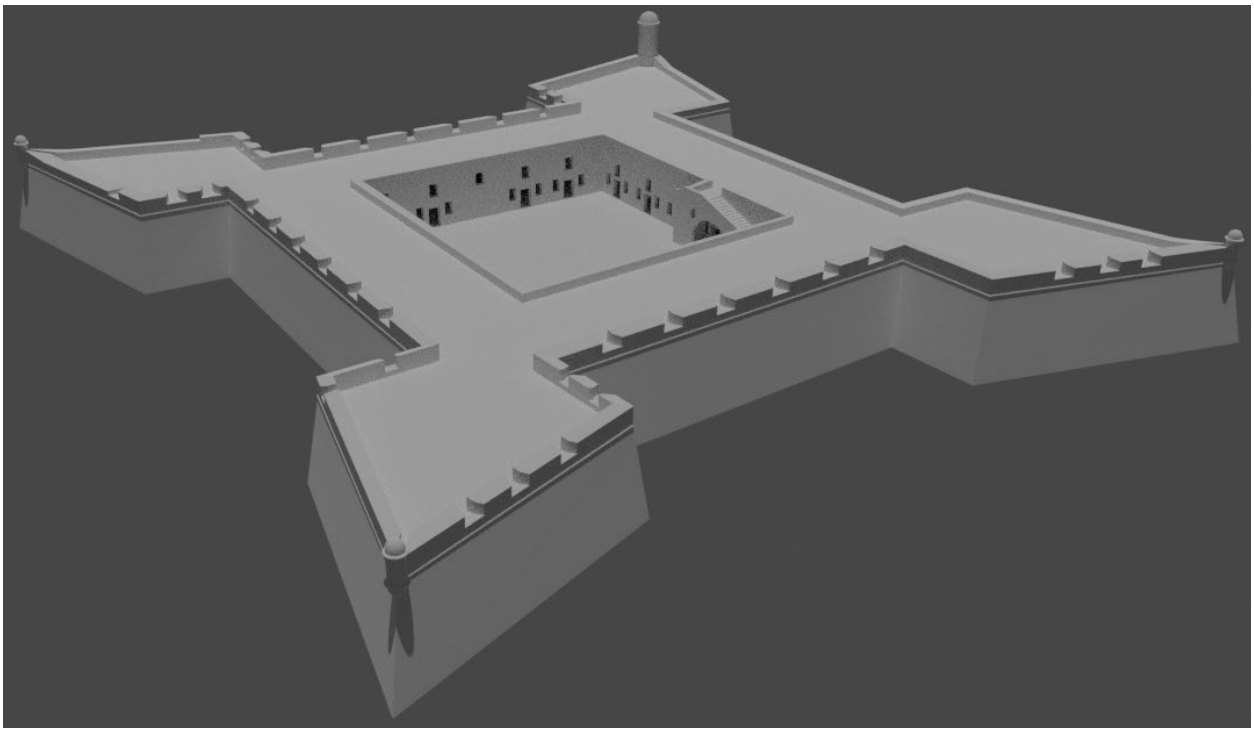


Figure HM.6: A full perspective render of the fort

Following the renders, I completed the interior by adding the corridors between rooms as they are on the maps of the fort. This can be shown in the latest progress of the fort, which is shown below in both a wireframe and solid view. (Figure HM.7-HM.8)

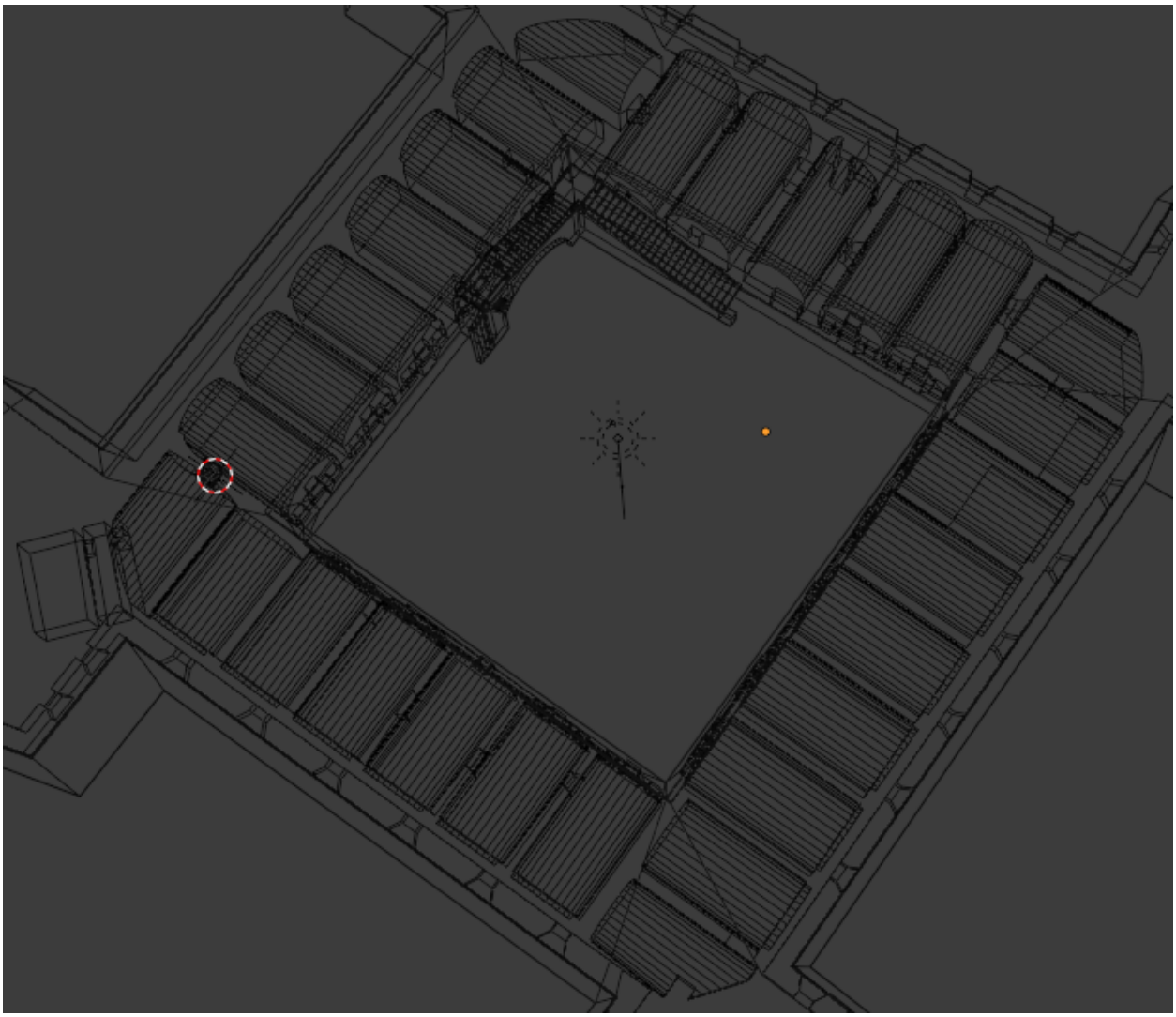


Figure HM.7: A wireframe view of the fort with all rooms, corridors, and windows, with the exception of the chapel door

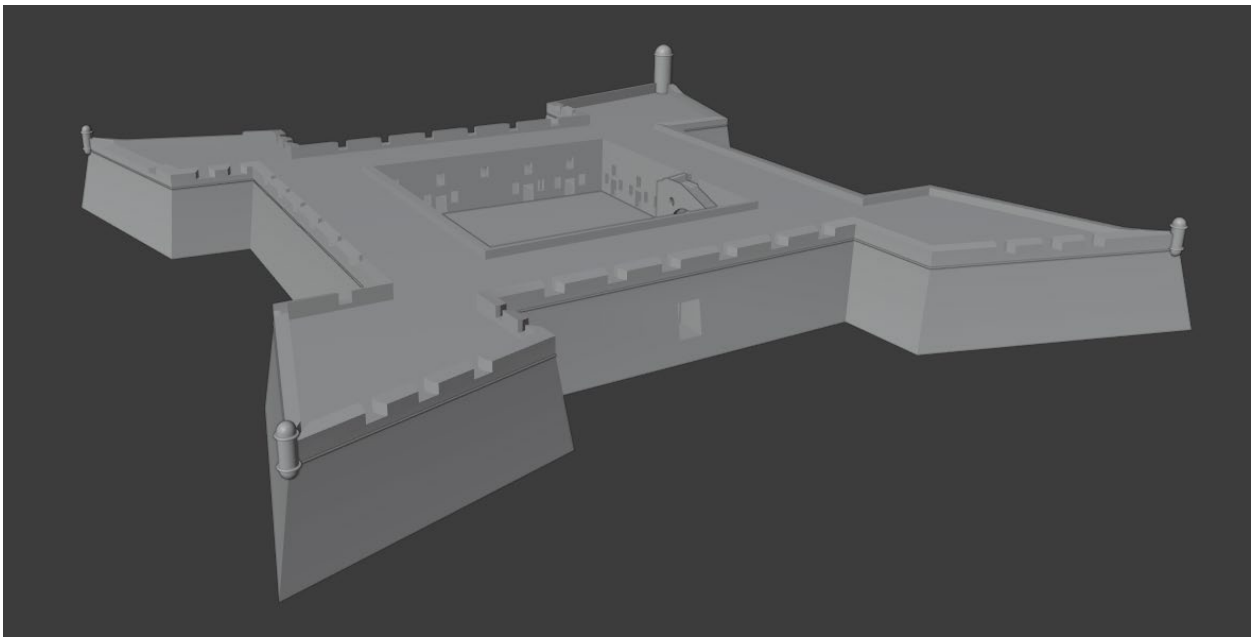


Figure HM.8: An unrendered view of the fort featuring the sallyport door

Upon completion of all interior corridors, I continued with cutting the final door into the model. This door, being rounded on top and being the chapel door, I wanted to ensure that it aligned with the door created by Juhwan. Once Juhwan had created the base door with the arch, I made a cut in the mesh accordingly. (Figure HM.9)

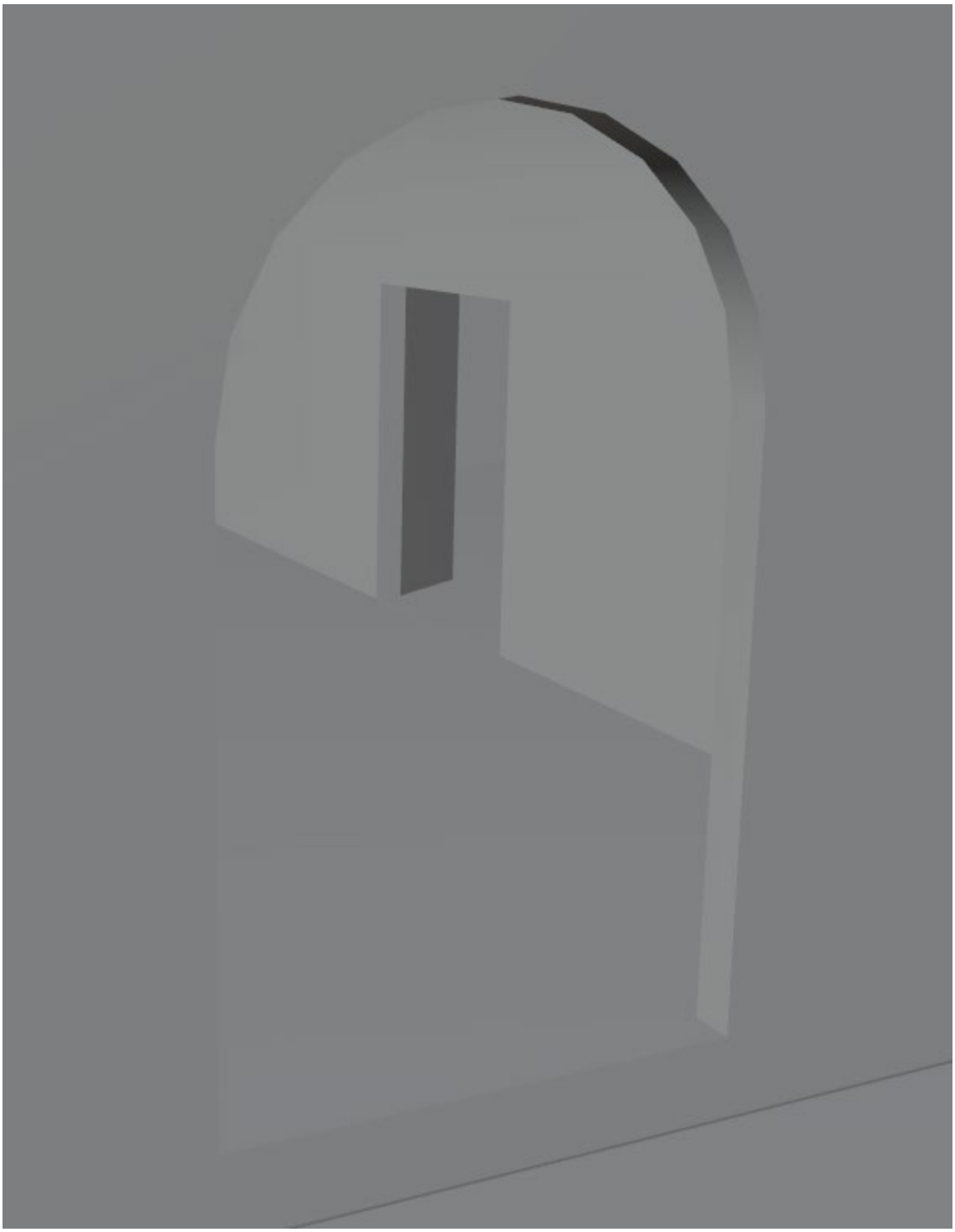


Figure HM.9: Chapel door opening

Following the cut for the door, I completed the first bit of trim work. Starting from a cube, I extruded from the top side and scaled the extruded square to match with the drawings provided. I continued this process until I reached the top of the fort. From there, it was a matter of duplicating the object enough times to align with the drawings. On the drawings, and at the fort, there are two styles of trim work that reach the top of the fort. Below are the two styles present as modeled. (Figure HM.10)

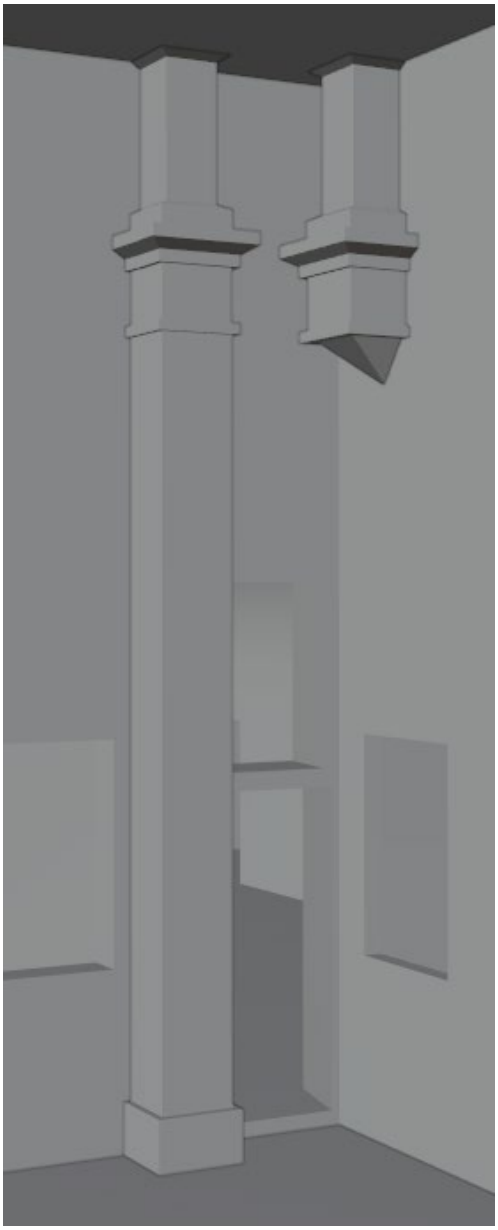


Figure HM.10: Columns featured in the courtyard

Following the pillar construction, the rest of the trim work, being only a small outcrop from the bottom of the fort in the courtyard, was simple. I created a small cube, scaled it accordingly, and aligned it to each of the walls. Following its creation, I used a boolean modifier to cut the holes where doors go. The pillars were created with this bottom trim attached. This trim work is shown below. (Figure HM.11)

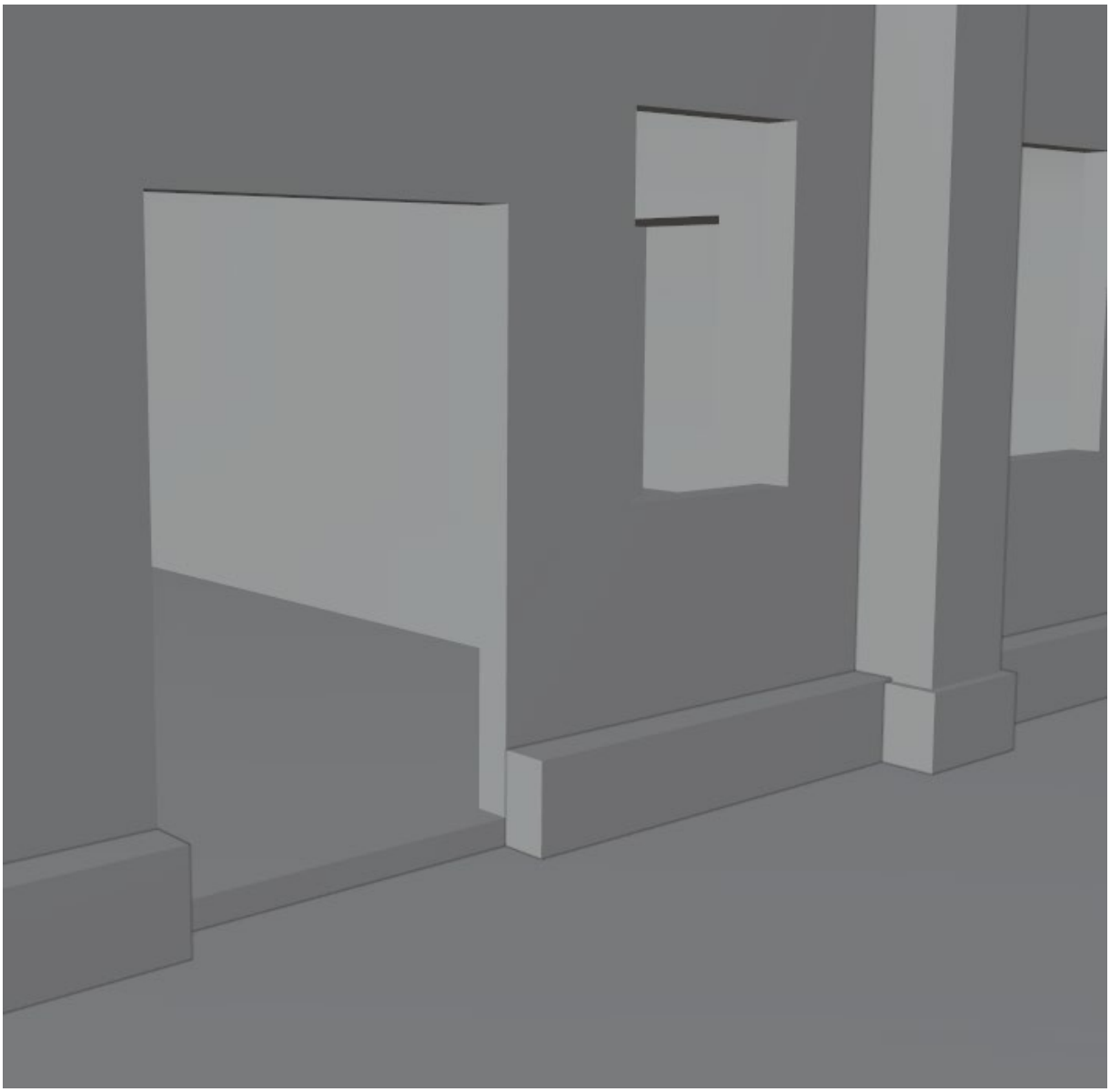


Figure HM.11: Trim work around the courtyard

In addition to this trimwork, I also hollowed out the lookout towers that are on each corner of the fort. Three of the towers, the southeast, southwest, and northwest towers, were the same, which made the process slightly easier, as I could simply duplicate the boolean objects used to make the cuts. The lookout towers are shown below. (Figure HM.12)

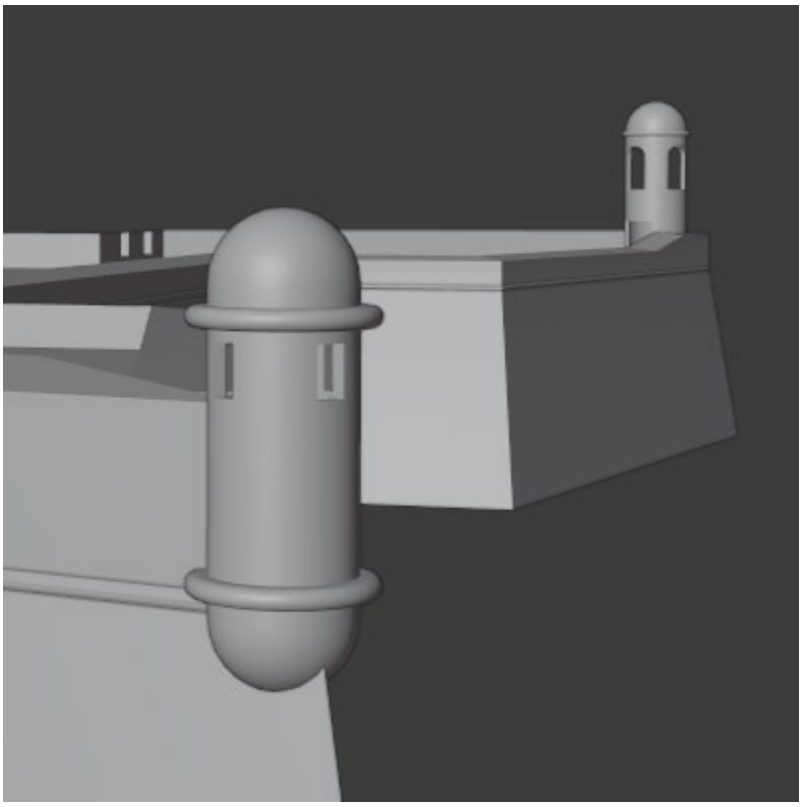


Figure HM.12: Lookout Towers with windows

Following the finishing of the lookout towers and base trim work, I started on the chapel doorway's trim work. Due to the intricacy of the trimwork, I decided it would be best to trace the lines on the drawing and extrude from there. The first column of the archway is completed and can be seen in the below image (Figure HM.13).

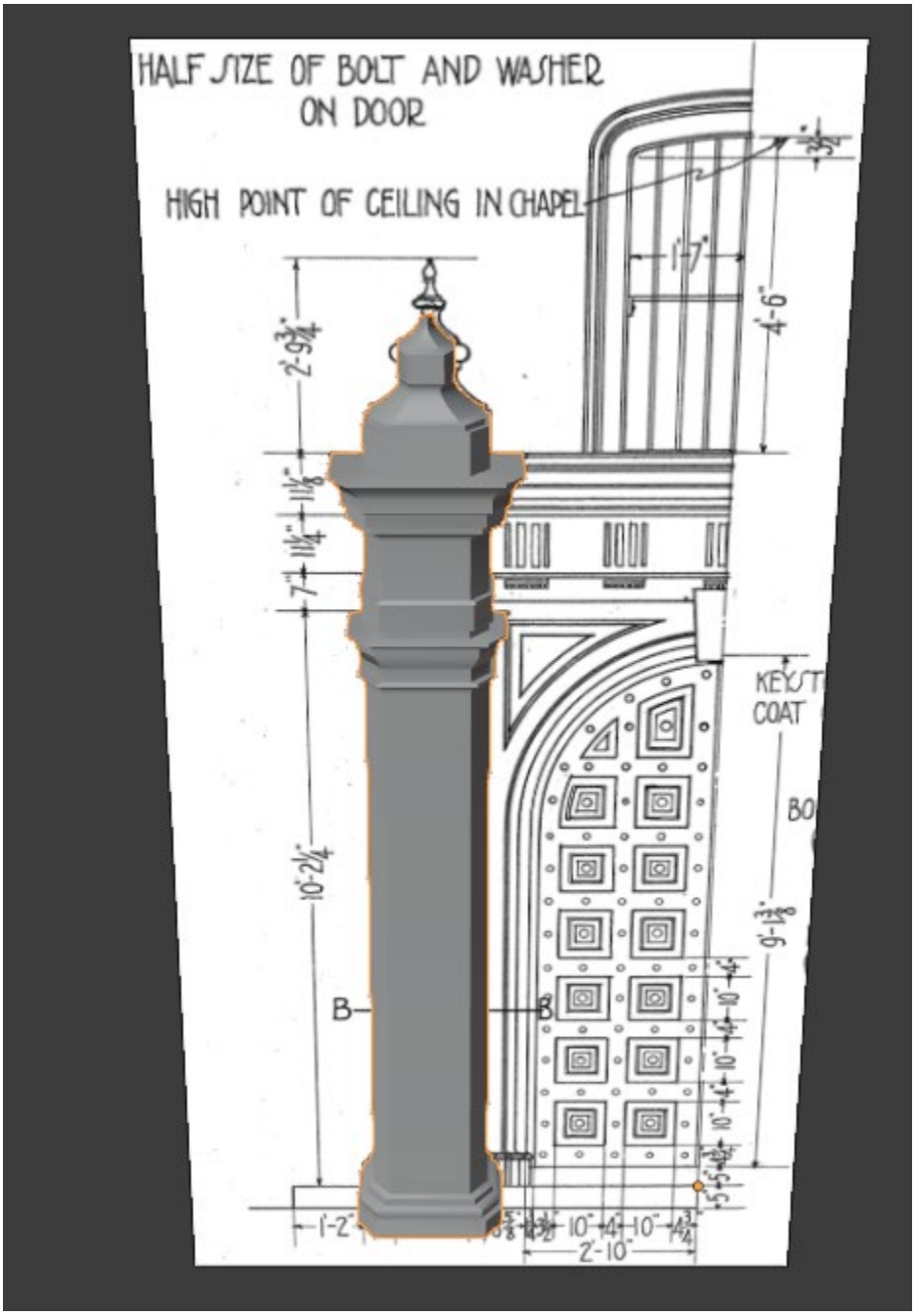


Figure HM.13: One column of the chapel door archway

I have kept some minutes in terms of modeling progress. That can be seen in the following timeline of modeling. (Figure HM.14-HM.27)

2/25/22 @ 10:42 am

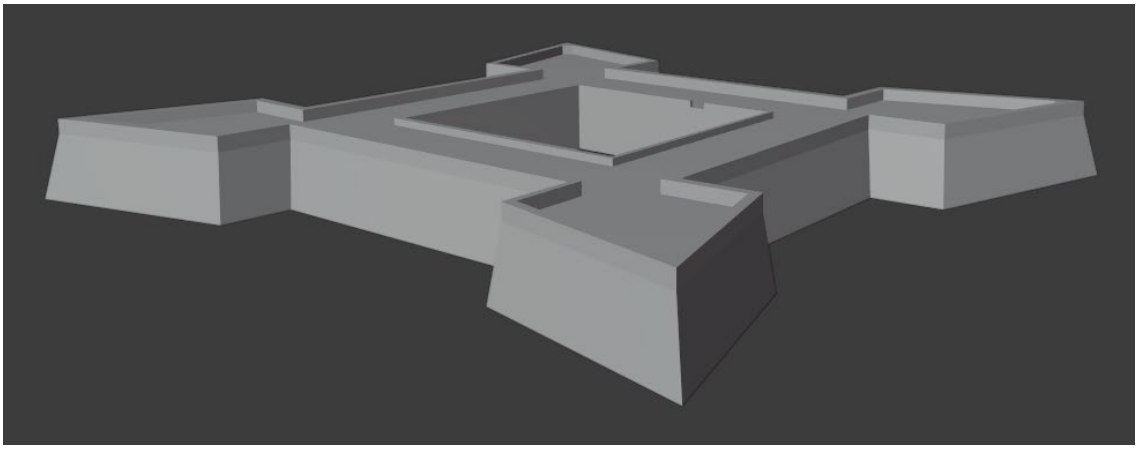


Figure HM.14: Extremely basic exterior of the fort

2/25/22 @ 10:45 pm

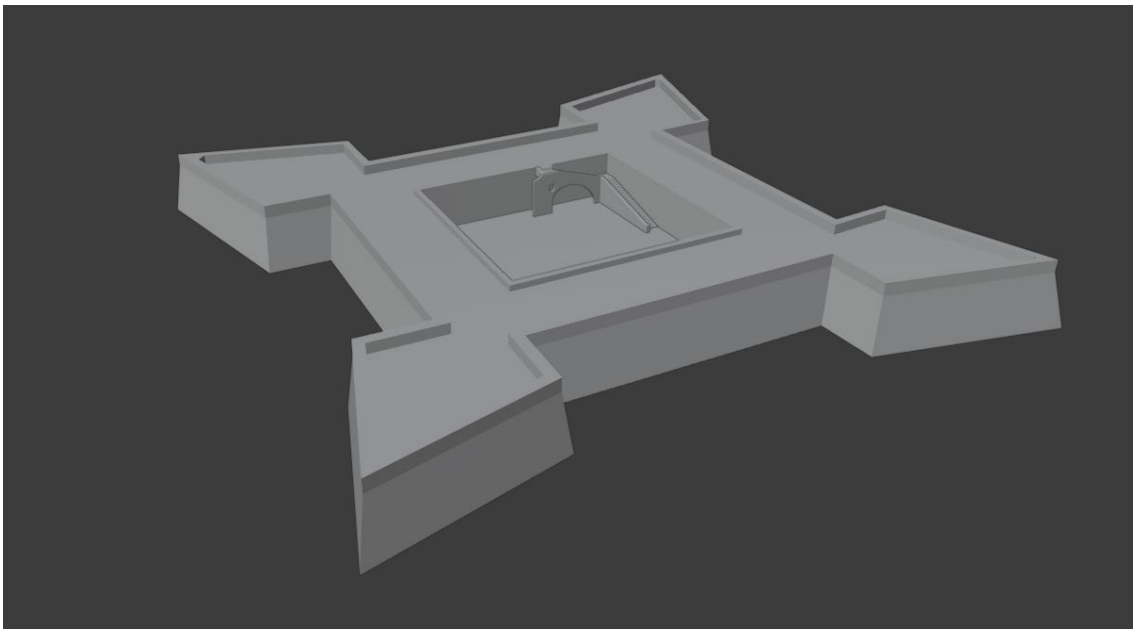


Figure HM.15: Addition of the staircase

3/1/22 @ 10:51 pm

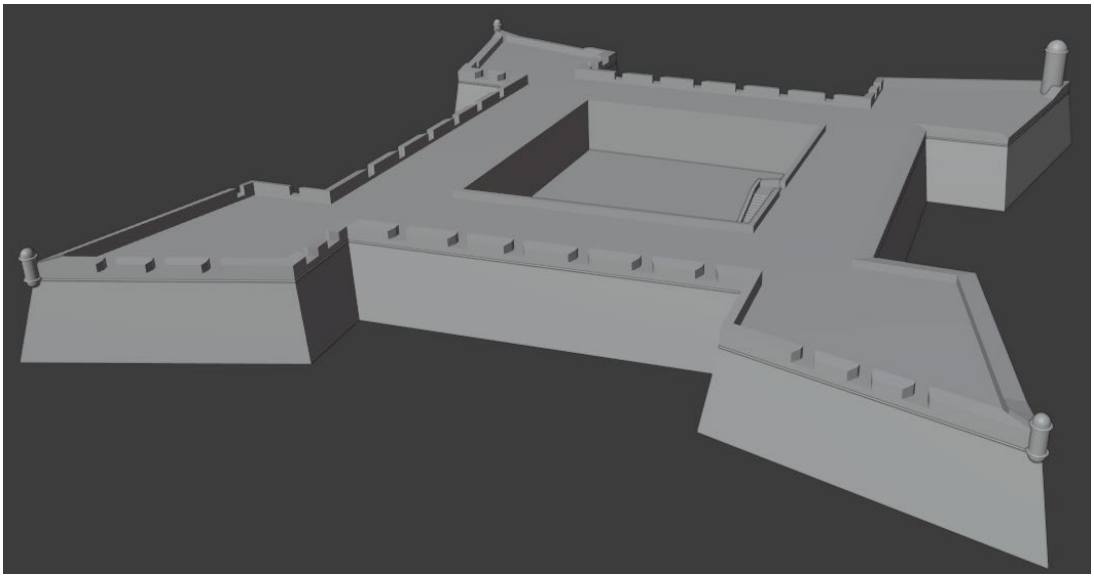


Figure HM.16: Fort with lookout towers and cannon ports

3/18/22 @ 11:11 am

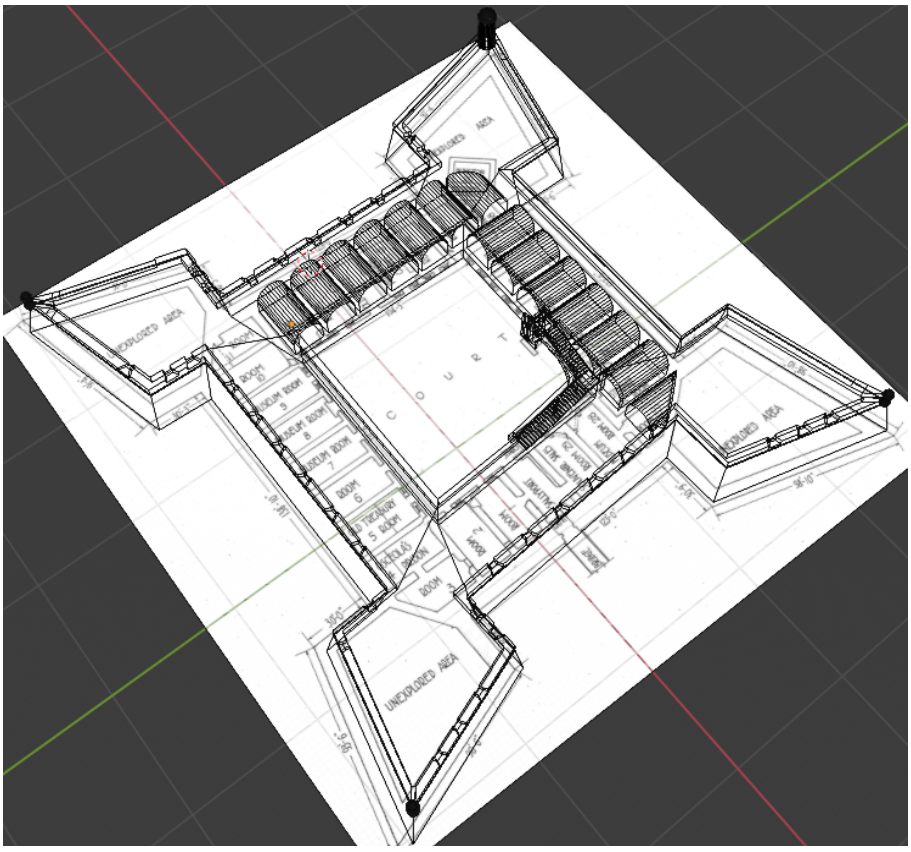


Figure HM.17: Wireframe of fort with half of the rooms implemented

3/31/22

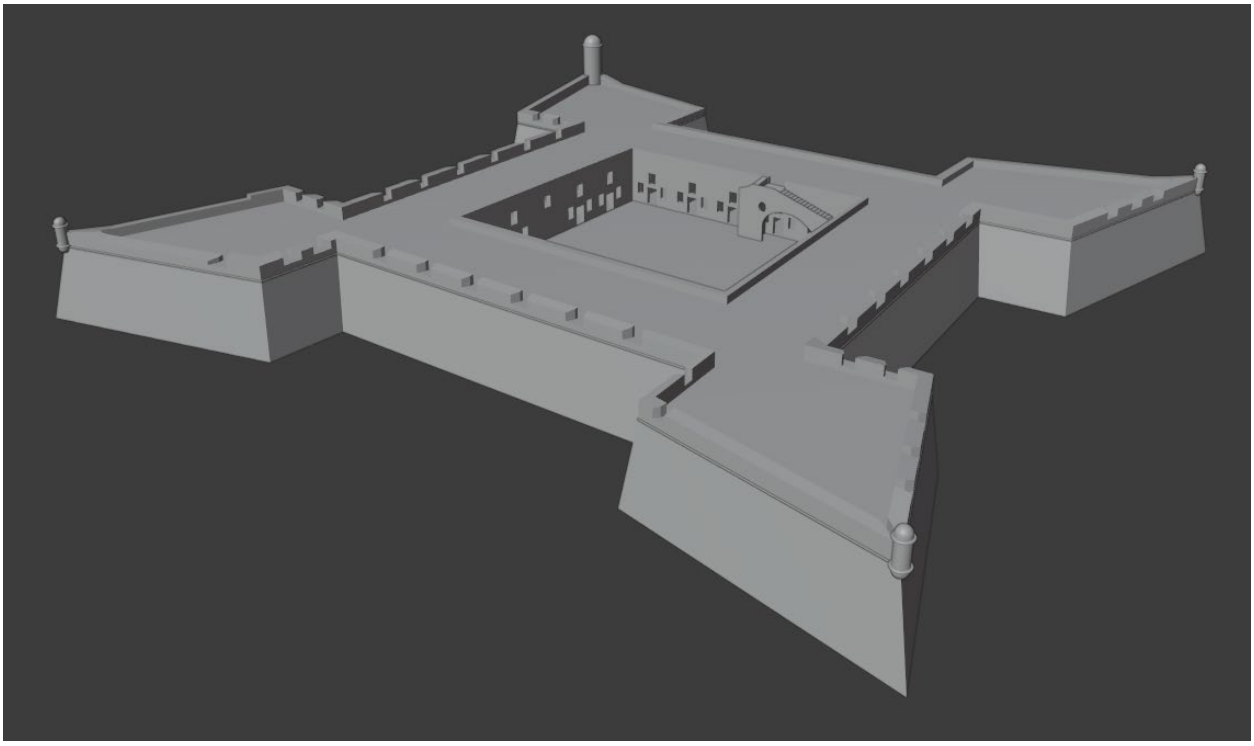


Figure HM.18: Fort with doors and windows implemented

4/1/22

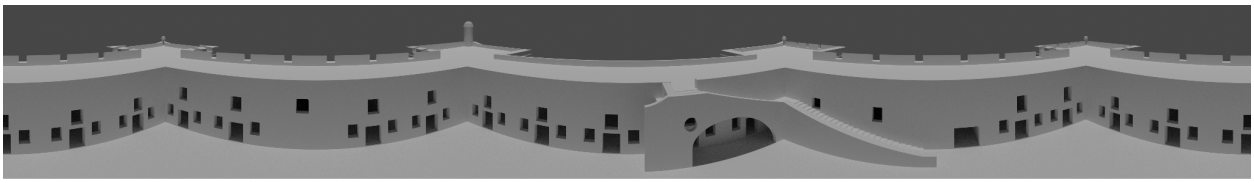


Figure HM.19: Panoramic render of the fort

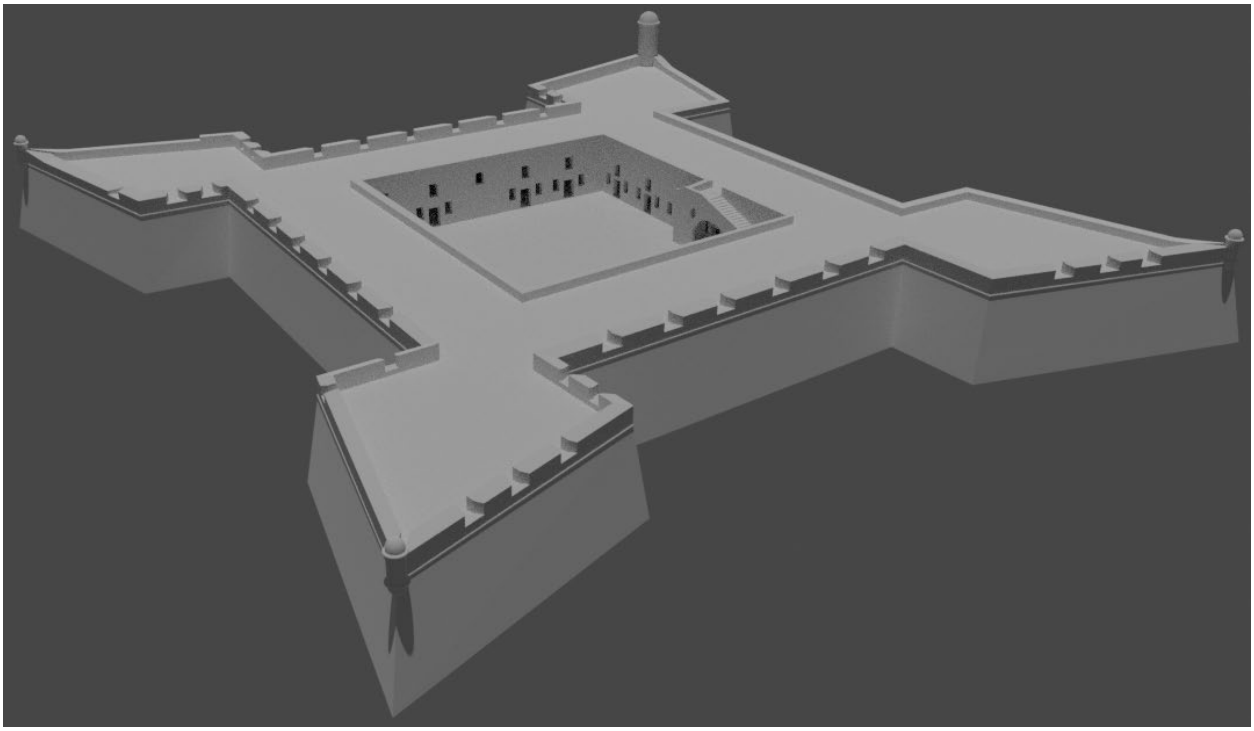


Figure HM.20: Perspective render of the fort

4/8/22

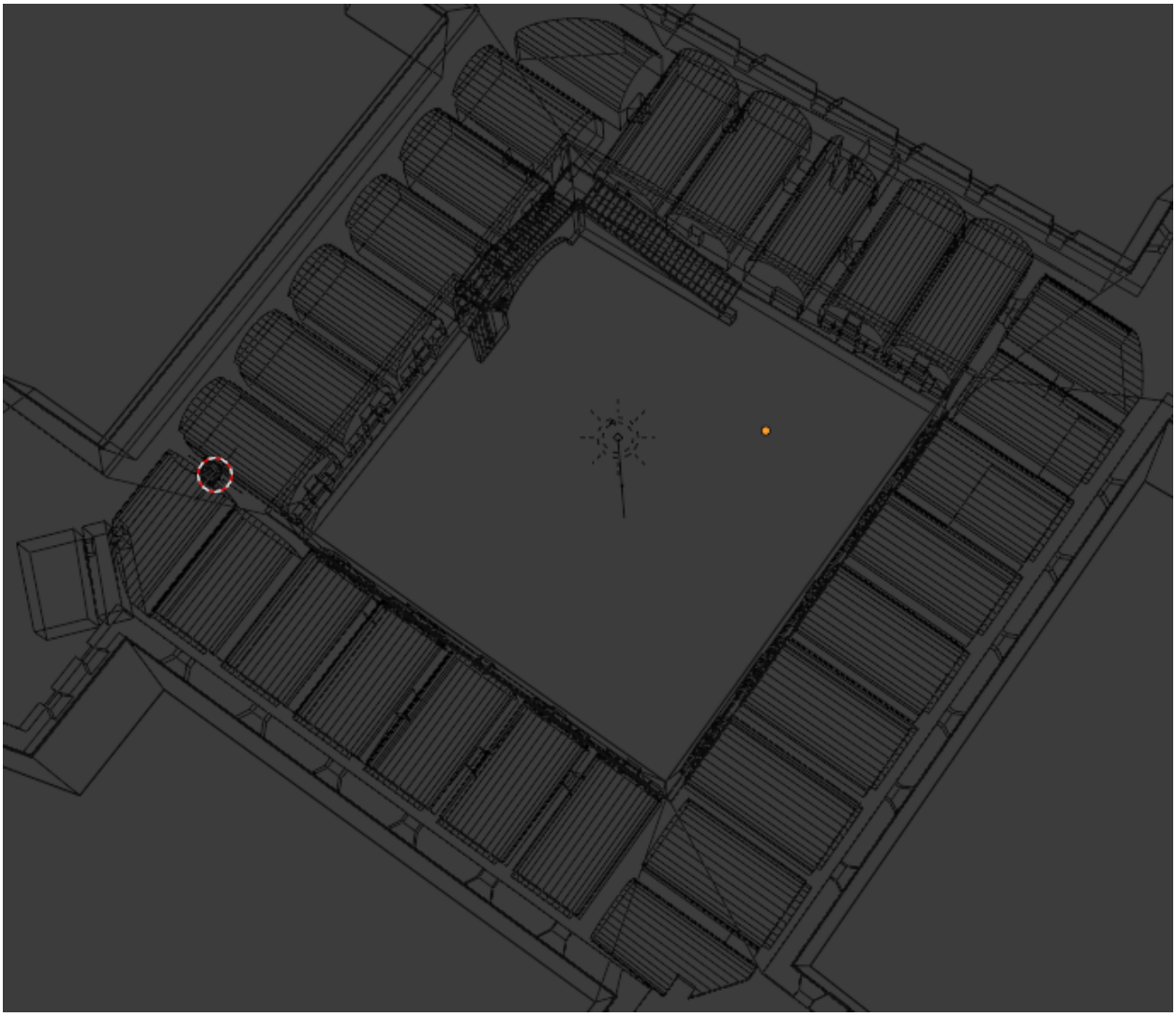


Figure HM.21: Wireframe view of all rooms with their corridors

4/8/22

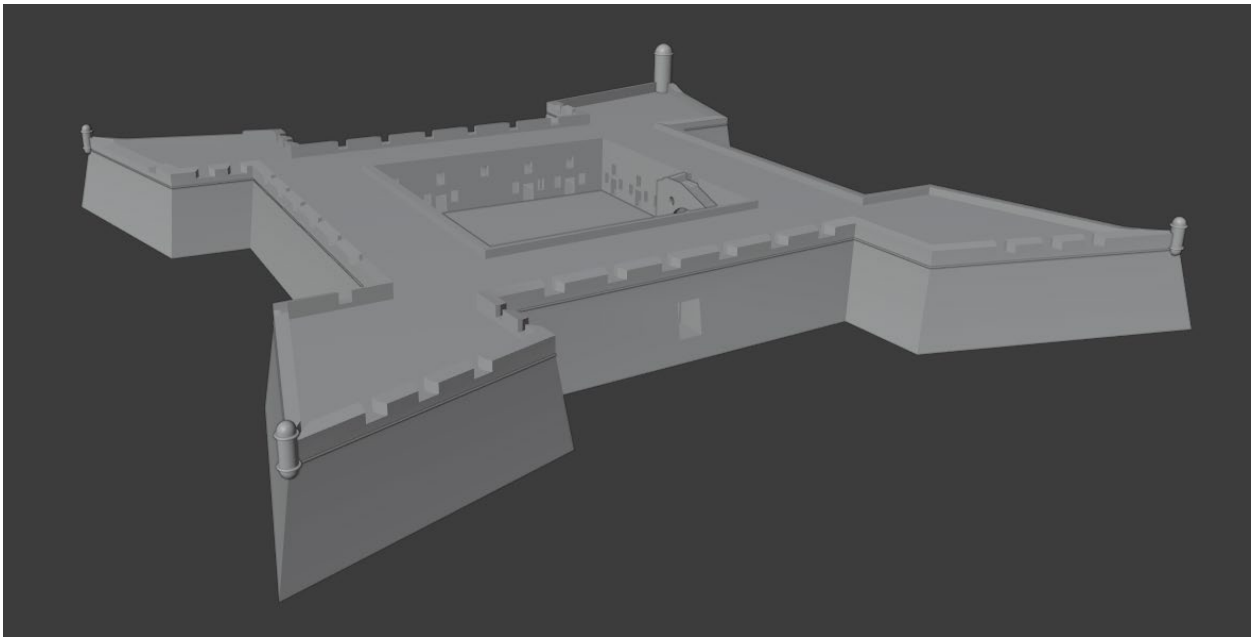


Figure HM.22: Photographic view of the fort with sallyport doorway

4/15/22

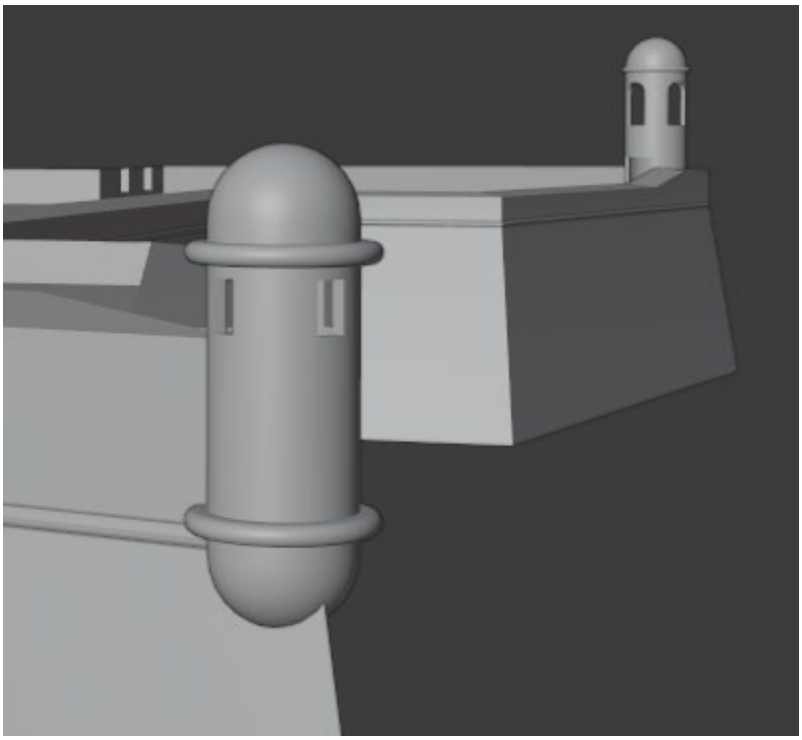


Figure HM.23: Lookout towers with windows

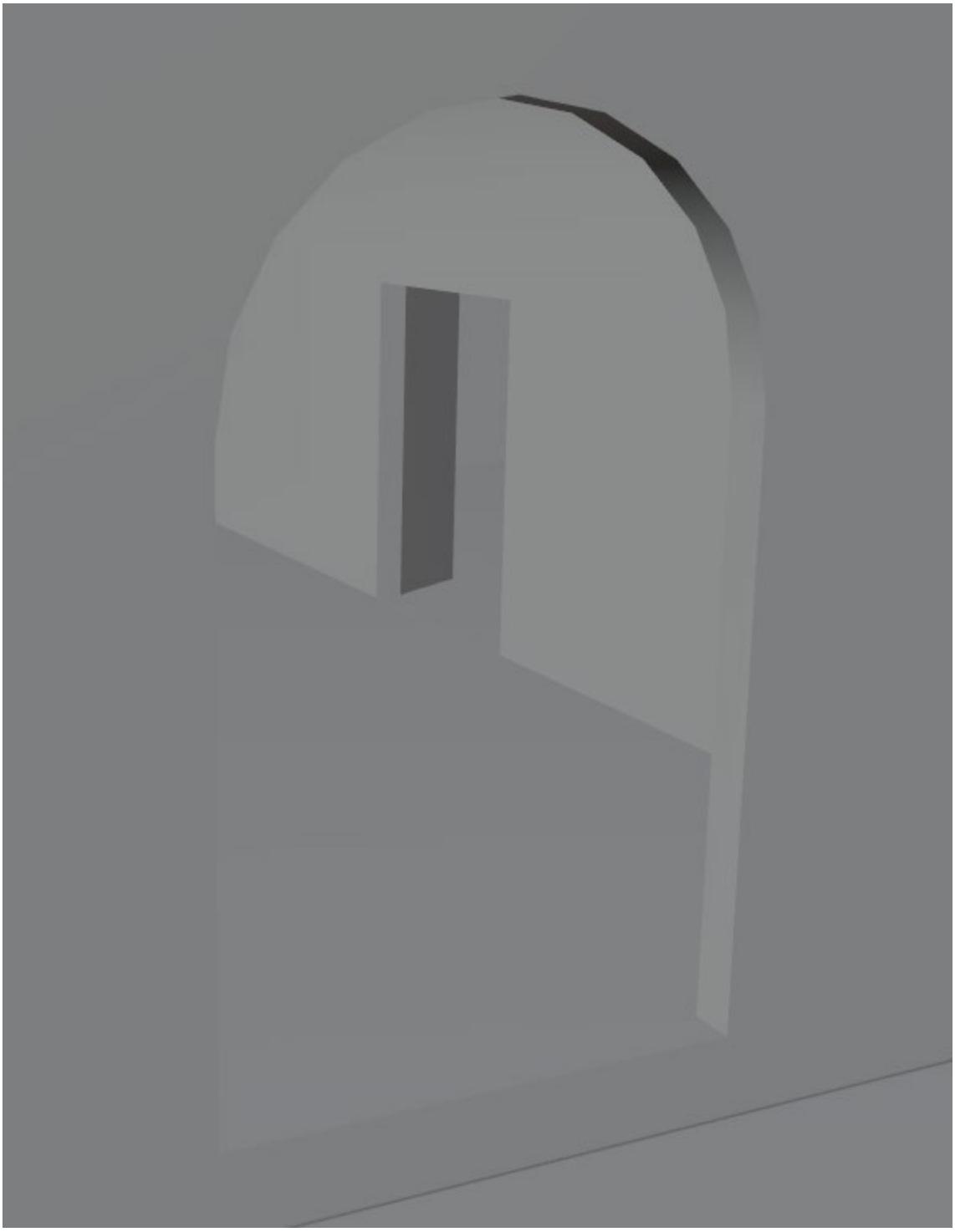


Figure HM.24: Chapel door opening

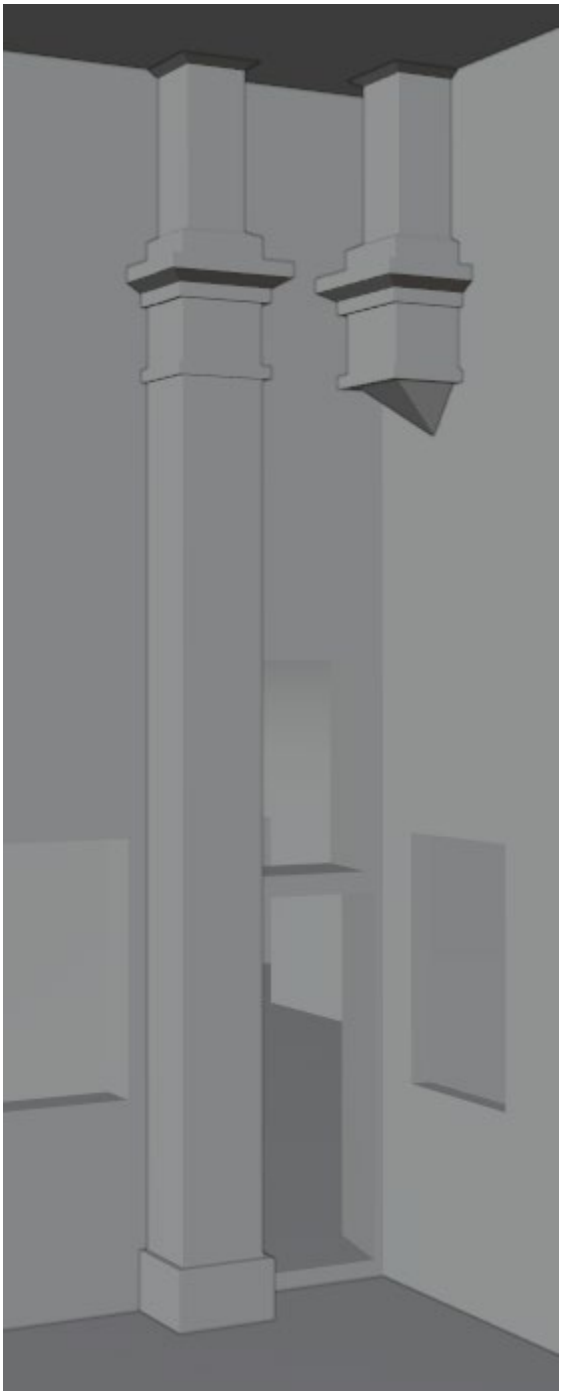


Figure HM.25: Columns present in courtyard

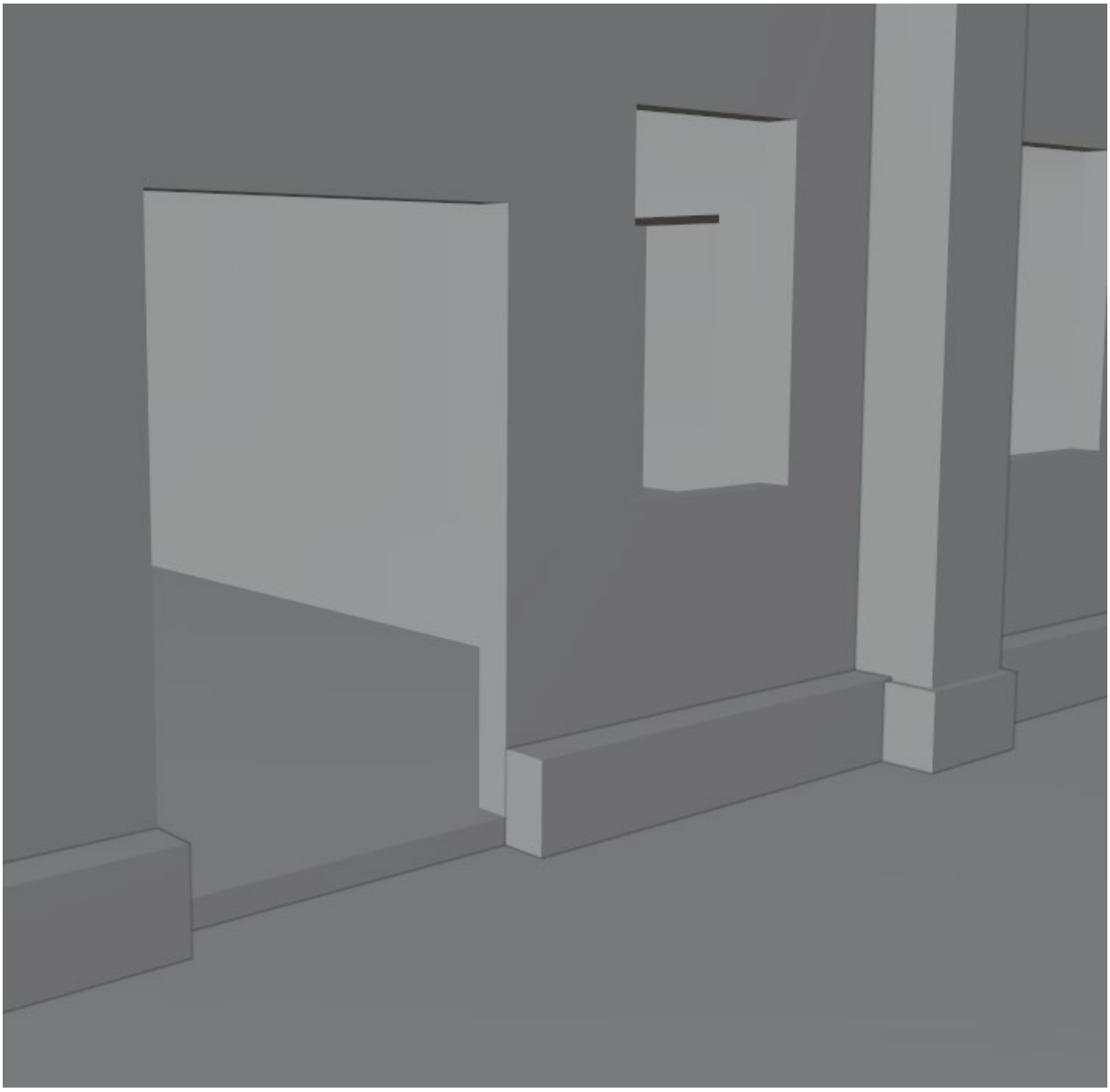


Figure HM.26: Courtyard trim work

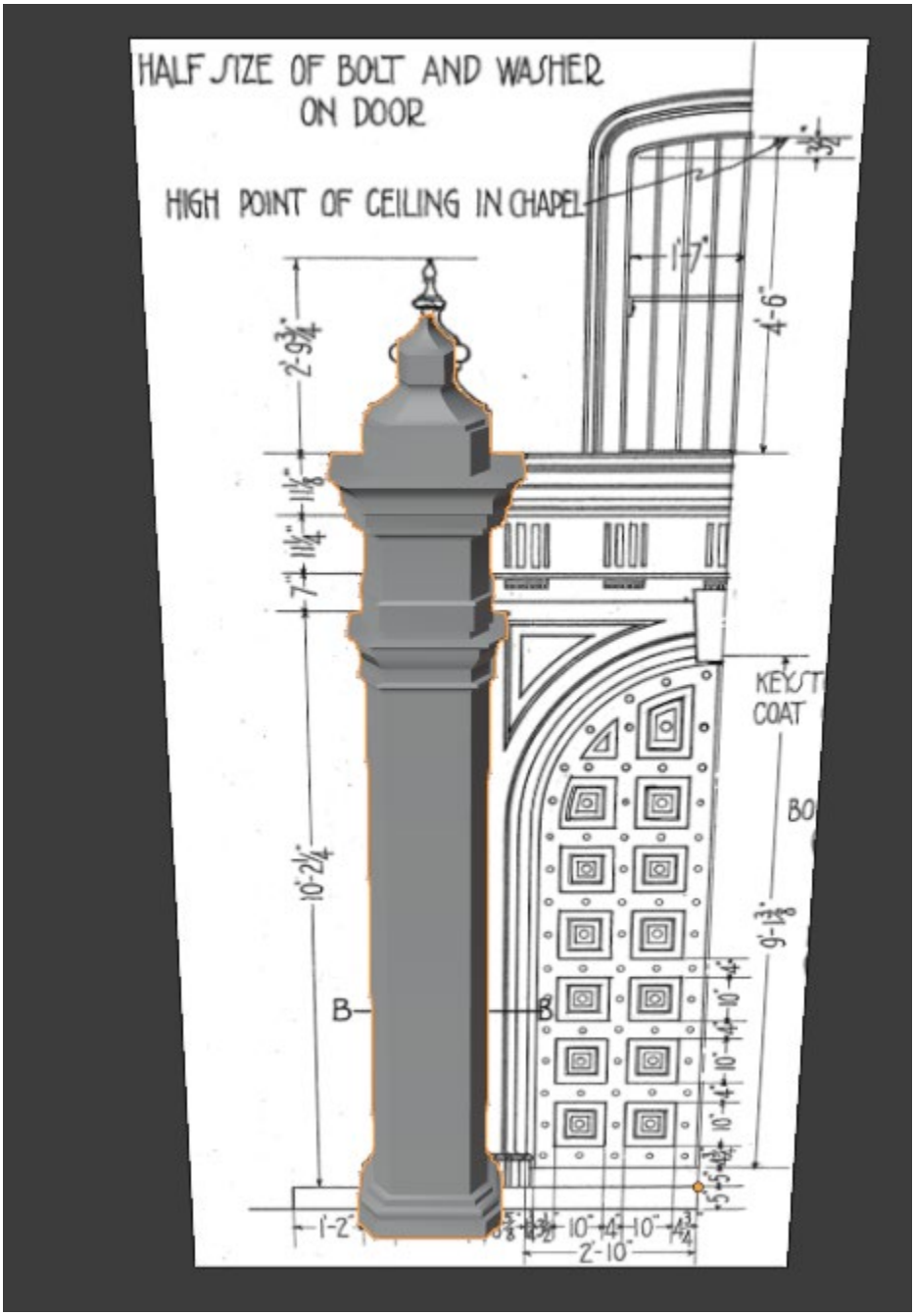


Figure HM.27: Courtyard trim work

9.4. Juhwan Park

Introduction to Blender

As I began working on Blender, I learned these important functions:

- Delete (x)
    - One of the very first things to do is to delete the default cube blender provides on a new project
- Add (shift + a)
    - Creates an object
- User Perspective
    - Front Orthographic (Numpad 1)
        - Main perspective the door was facing
    - Right Orthographic (Numpad 3)
        - Side view of the door
    - Top Orthographic (Numpad 7)
        - Top view of the door
    - Flip (Numpad 9)
        - Was needed when editing sides
- Shading (z)
    - Mainly used the wireframe (see-through) and and solid (default) shading
- Edit Mode (tab to toggle between object mode (default) and edit mode)
    - Allows the editing of individual edges, vertices, and faces of an object
- Box Select (b)

    - selects multiple elements within the box
    - useful when selecting an entire side with or without wireframe shade
- Loop Cut (ctrl + r)
    - “splits a loop of faces by inserting new edge loops intersecting the chosen edge.” [3]
- Duplicate (shift + d)
    - creates a new object identical to the selected object
- Subdivision Surface (modifier)
    - makes the surface smoother by subdividing the object appropriately

Transformation Functions

- All of these functions can be done along an axis by pressing x, y, z, or lock an axis by pressing shift + x, y, z
- Can input values for accurate transformation
- Rotate (r)
    - rotates the selected elements with its origin as its center of rotation along the mouse cursor
- Grab (g)
    - changes the object’s displacement
- Scale (s)
    - scales the selected elements with its origin as its center of rotation along the mouse cursor
- Extrude (e)
    - “duplicates vertices, while keeping the new geometry connected with the original vertices. Vertices are turned into edges and edges will form faces.” [4]

Spanish Treasury Door

After deleting the default cube, a plane with no z dimension was created.
Then the view was set to front orthographic and the plane was rotated along the x axis by 90 degrees to face the front orthographic view.
The image below (Figure JP.1) was then added - it could be added as a reference, background, or planes, and although users seemed to prefer images as planes, I selected 'images as background' as I was more familiar with it.

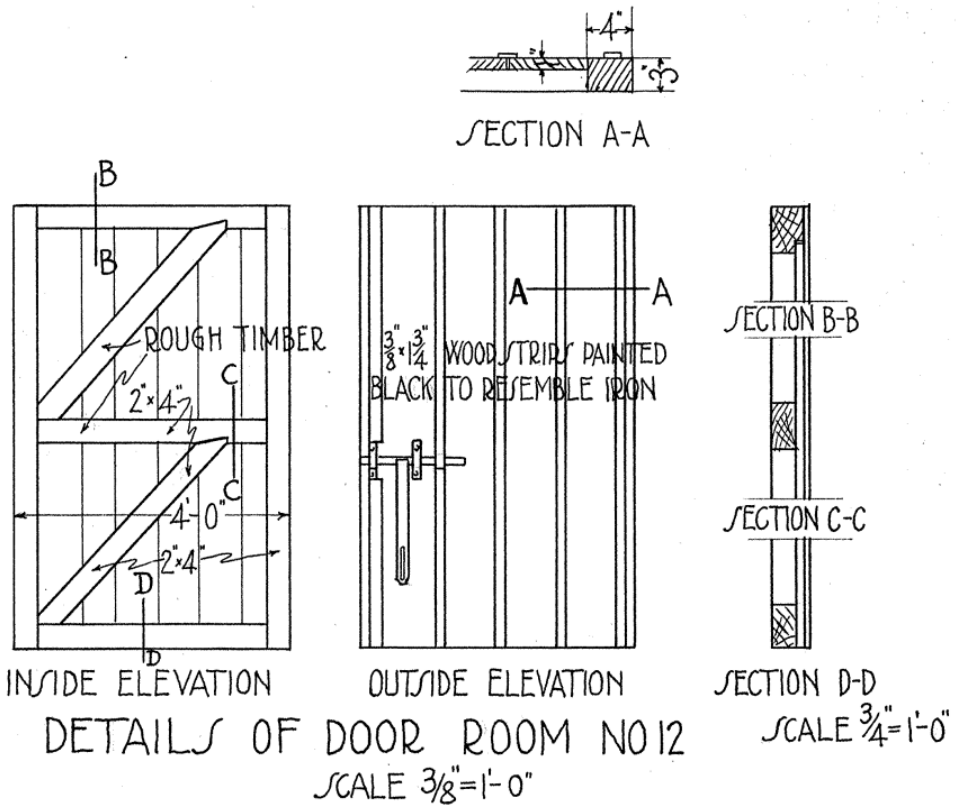


Figure JP.1: The first model I began designing for our project

The only noticeable difference is that an image as background will only be visible in a certain angle while an image as plane will be visible in all angles.
Having the image selected, the plane was repositioned and scaled multiple times until the bottom and the sides of the door were aligned with the bottom of the plane (Figure JP.2).

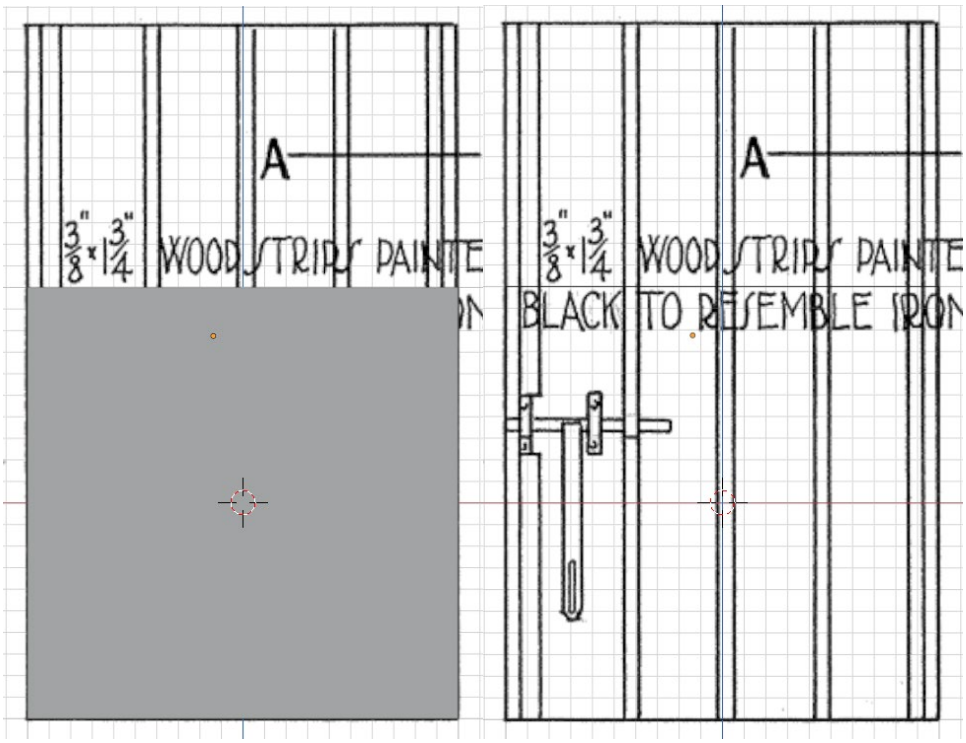


Figure JP.2: Solid vs Wireframe shading

Going into edit mode, the top of the plane was box-selected and extended (grab) to the top of the door along the z axis. Then using loop cuts, the edges in the door were created.

At this time I looked into the images that were taken in the fort when we visited to have a different reference to compare the actual door with the figure. Realizing that this door is nowhere to be found, I consulted the group and Dr. Giroux then began working on a similar but different door (Figure JP.3).

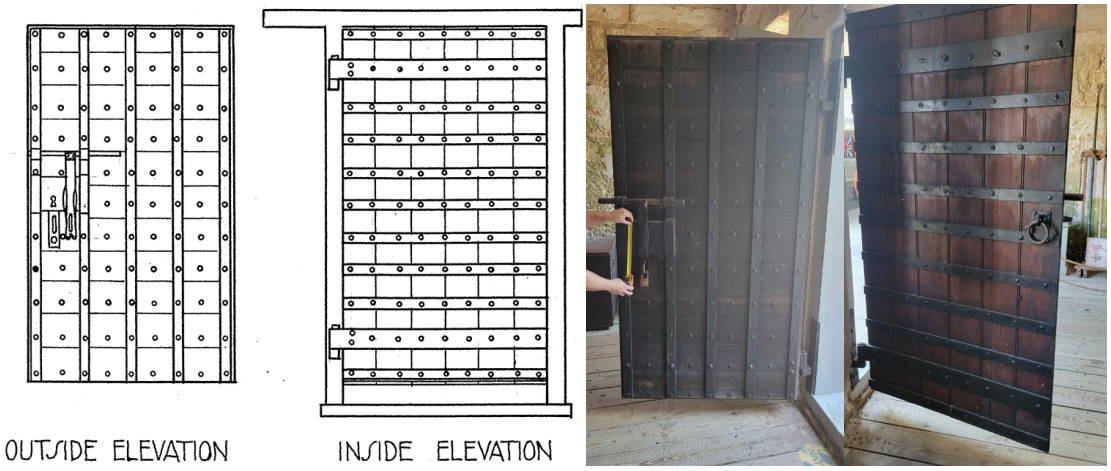


Figure JP.3: blueprint and images of the new door

Note that the front view of the door was designed first.
As shown on the image above, these doors have vertical metal strips attached to them. To design this, after making the loop cuts like the previous model, in edit mode the faces were selected and extruded forward.
Similarly, the extra material on the left center of the door was designed (Figure JP.4, 5).

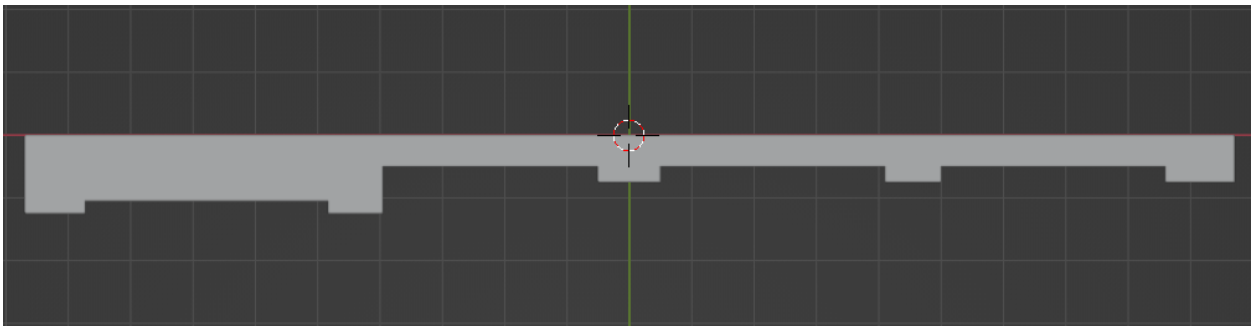


Figure JP.4: Top Orthographic view of the door (with extra material on left)

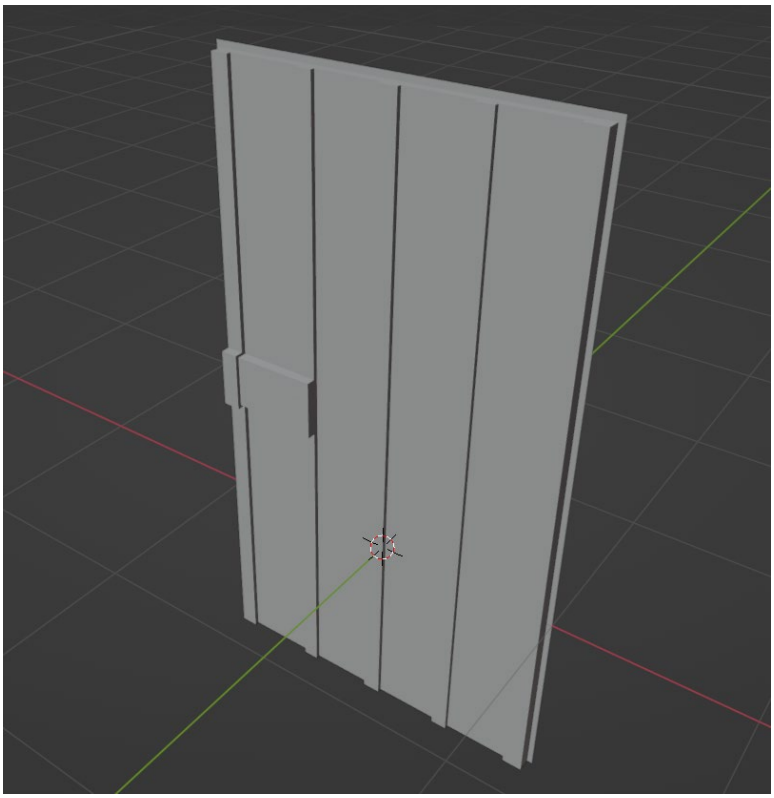


Figure JP.5: user perspective of the model at this time

The door also has bolts attached to these strips. To design this, rather than making the actual bolts that would go through the door, only the outsides were

designed as users will not be able to see through the door and it will make the design simpler.

A sphere was added, then half of the sphere was removed so that it would become a semi sphere. It was then resized so that it would look flatter and smaller, repositioned so that it would be where the very top left bolt would be and then began duplicating, noting that the ones on the strips are closer forward than the ones on the door. After duplicating the semi sphere into a row of semi spheres, the ones on the door were selected and pushed inwards so they would be on the door. Then, selecting all of them together, they were duplicated down the door, creating multiple rows.

At this time, the software was being very slow at selecting objects, as there were over 90 objects in play. Shortly, figuring out that the large number of objects was the cause of this issue, all the semi spheres were grouped into a single object. Around this point subtle differences between the blueprint and the actual door were becoming noticeable, such as the location of the lock and the shown bolts. Seeing that some of the bolts were covered on the left center of the actual door, those semi spheres were removed (Figure JP.6) and in the process found out that a column had a duplicate semi sphere within, as some semi spheres had to be removed twice. This issue occurred because while duplicating, I pressed the esc key which left the duplicated object right where the original object was. Unaware, an entire column with two semi spheres was duplicated down the rows. Now the ten rows and nine columns were checked - whether there were more mistakes or not, this was painful.

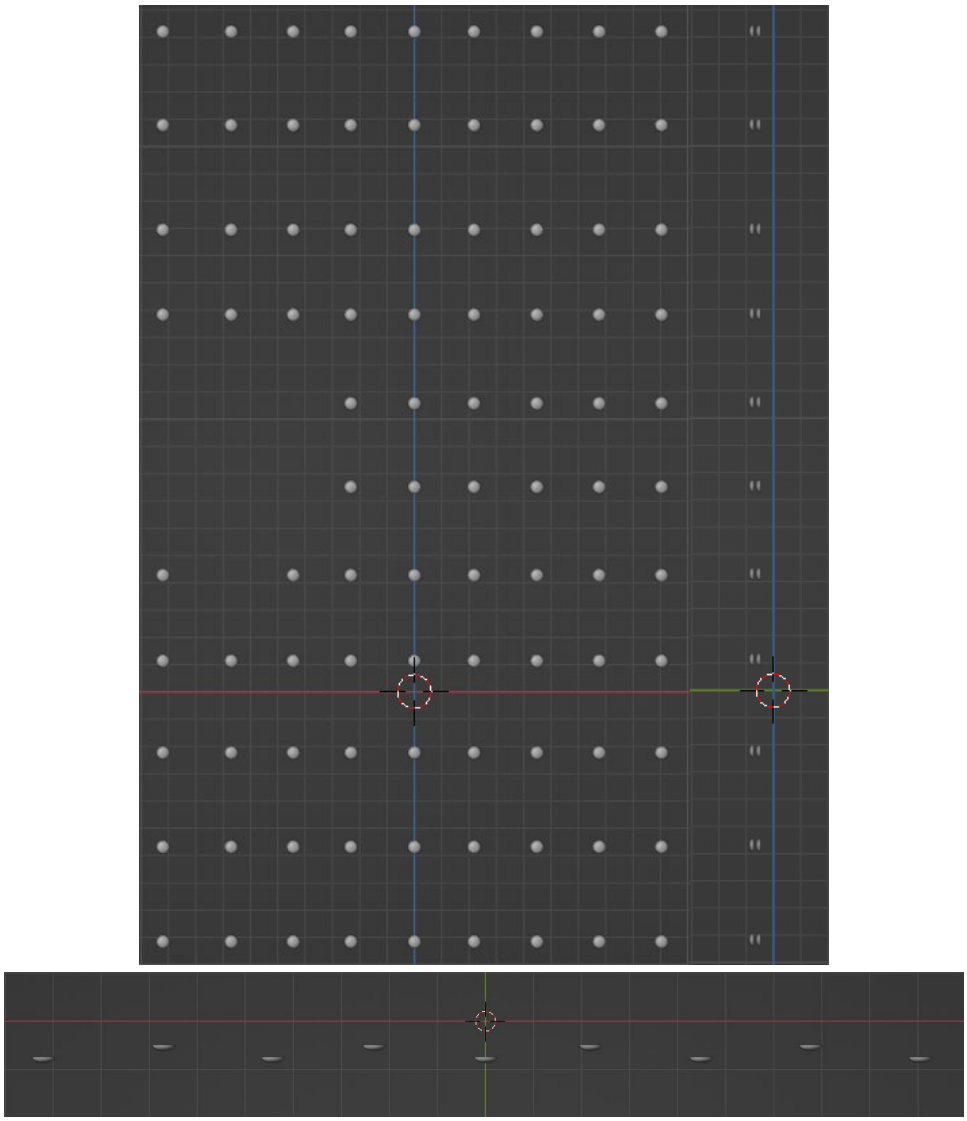


Figure JP.6: Front, right, and top orthographic view of the semi spheres

Designing the lock handle, a cylinder was first created, then the faces of the bases were removed. Next, the entire cylinder was extruded and scaled to give the cylinder thickness. More scaling and repositioning and the rings were complete (Figure JP.7).

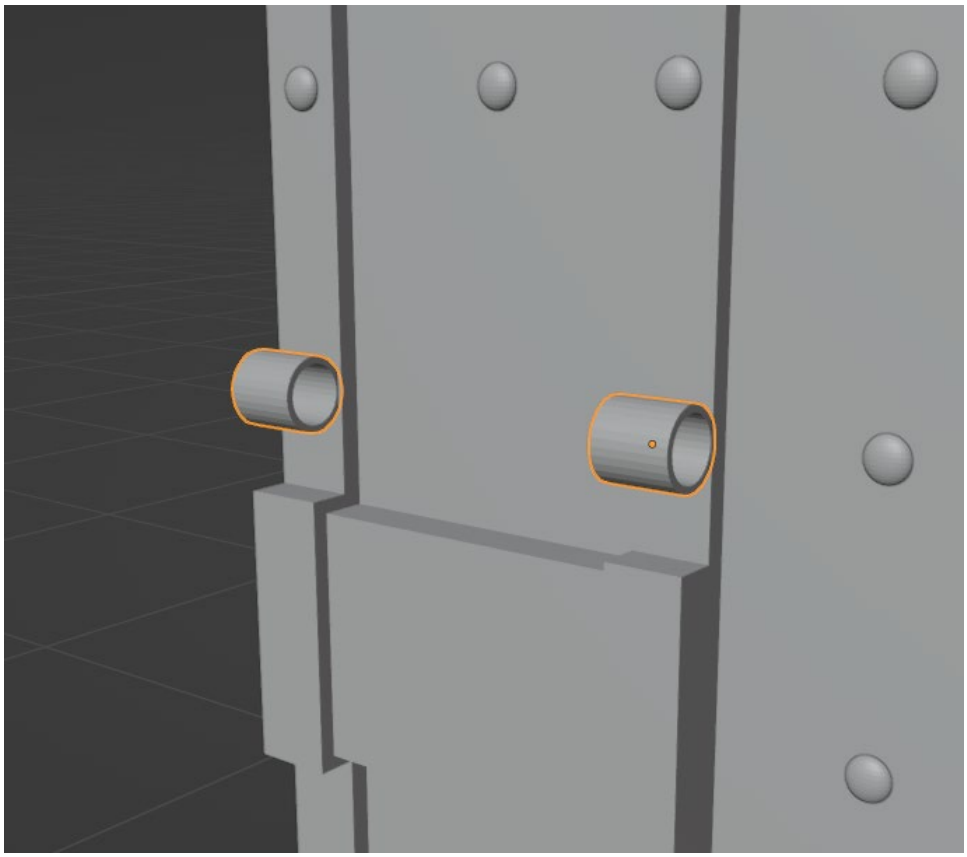


Figure JP.7: the surroundings show where the bolts were not needed

Also note that comparing the blueprint and the actual door, the blueprint shows the rings in between the rows of bolts while the actual door has it aligned to the fifth row of bolts

The lock handle was the most intricate part of this model. The area where the horizontal cylinder is mended to the vertical strip has a cube that is carved out along the four edges (Figure JP.8).

Figure JP.8: the x shape cube being modeled

To mimic this, I learned that the boolean modifier does exactly this.
A sphere was created, aligned with the cube with appropriate distance, and duplicated with equal distance along the four edges so that a moderate amount of the edges were covered by the spheres. Then the “difference” from the boolean modifier between the cube and the spheres was applied (Figure JP.9).

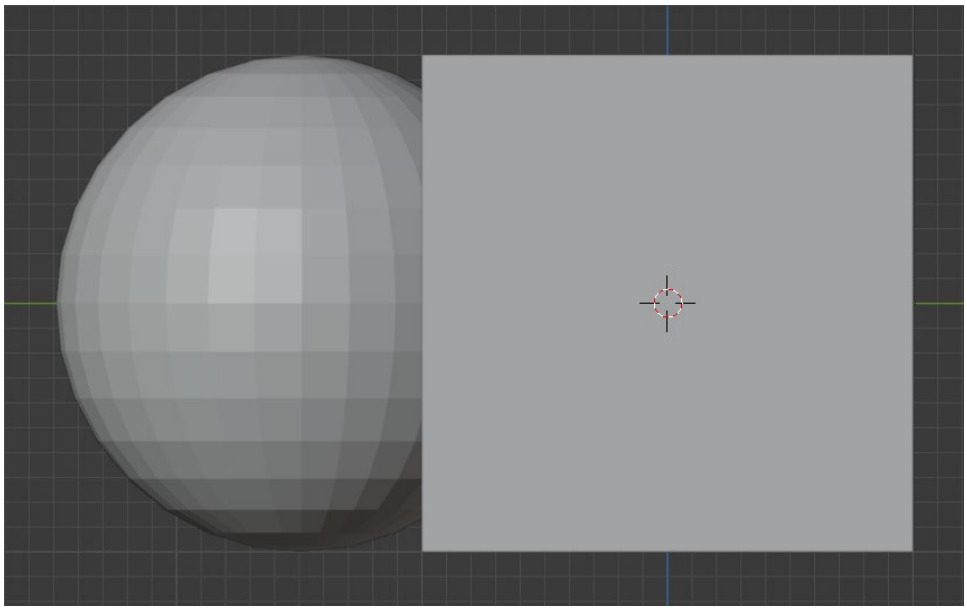


Figure JP.9: demonstration

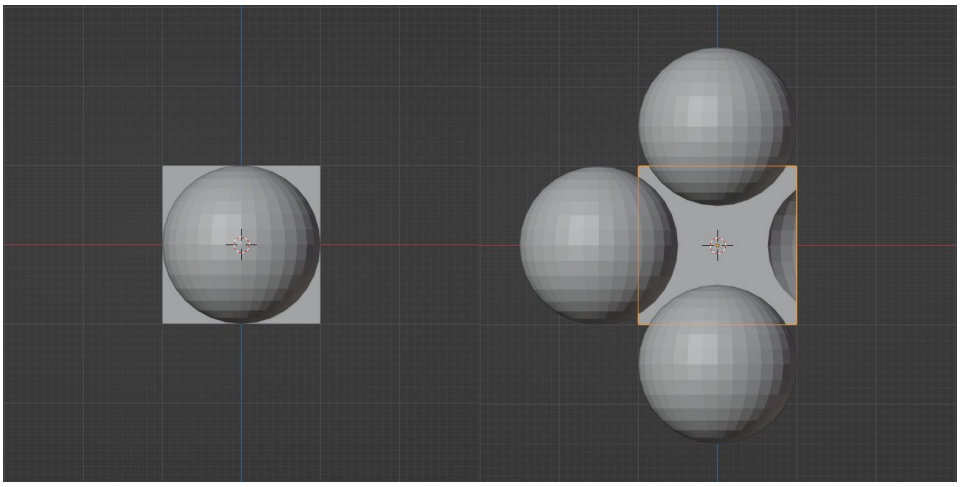


Figure JP.9 continued

One thing I noticed here was how smooth the sphere could be through the subdivision surface modifier (Figure JP.10). It comes at the cost of higher rendering time and higher polygon count however, so after consulting with the group, it was not used.

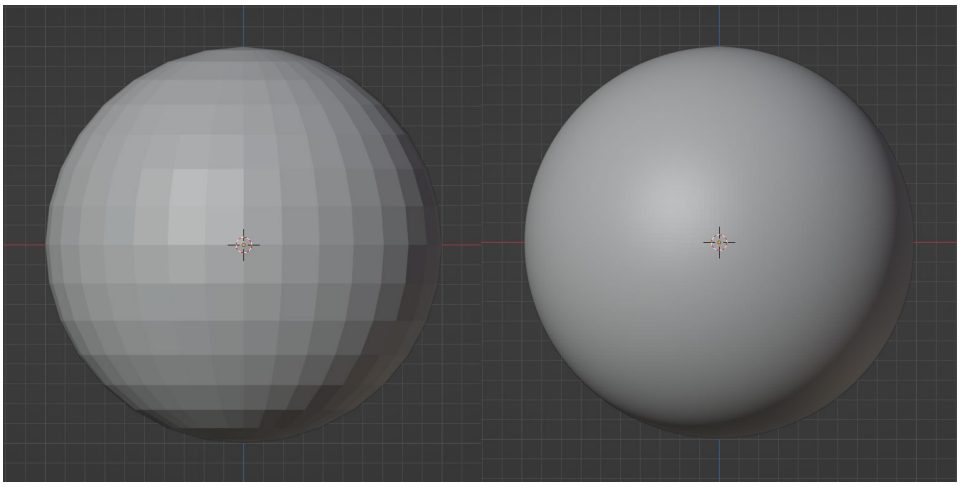


Figure JP.10: Original sphere vs 3 levels of subdivision modifier

Another concern for this handle was the center, where metal is bent inwards, right below the cube (Figure JP.11).

Figure JP.11: Bent-in metal strip

Although it may seem like the same strategy as boolean difference, it can be noted that the metal is carved inwards, and just as the boolean difference carves out the selected object, boolean union adds the object, but it adds the entire object. More boolean difference modifiers would have to be applied to apply union on only the wanted segments. This will be worked on in the near future.

Lastly, the bottom of the handle needed some smoothening and more boolean difference. For the smooth bottom beveling was used and for the hole the boolean difference with a modified cylinder was taken.

The final part of the door was the ring that the hole of the handle fits in. After creating a torus (or a donut), it was made thinner through scaling each vertex along its local normal, and cut it in half (Figure JP.12).

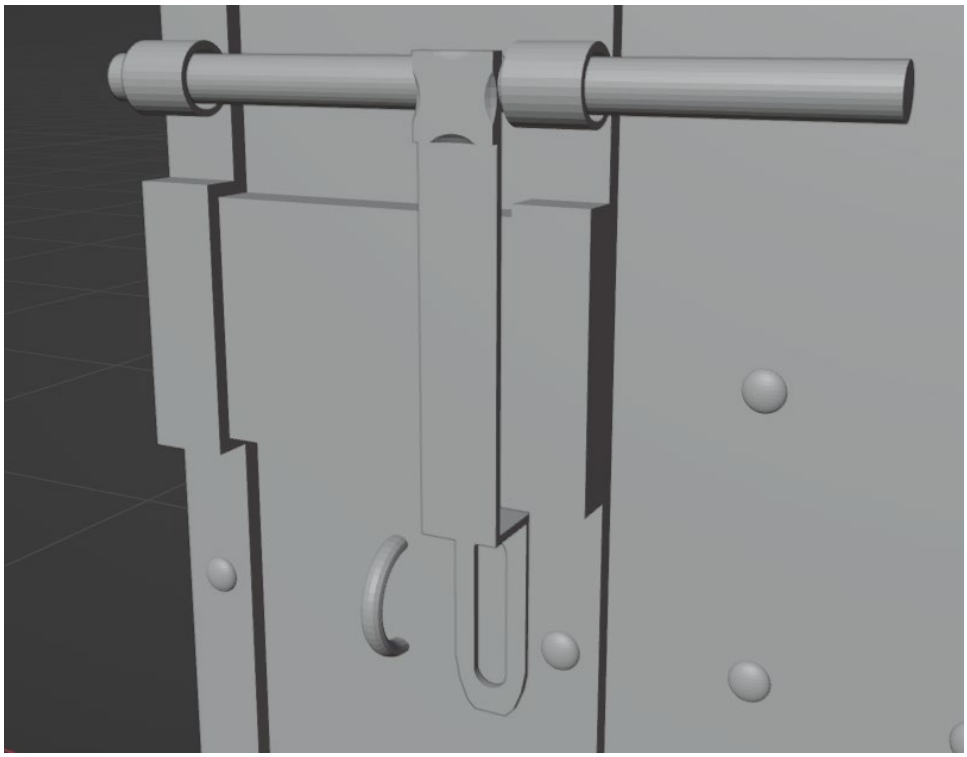


Figure JP.12: finished lock handle

The back of the door was much simpler than the front (Figure JP.13). There are two metal strips on top and bottom that connect to the outside frame of the door and this along with the outside frame itself will be implemented next.

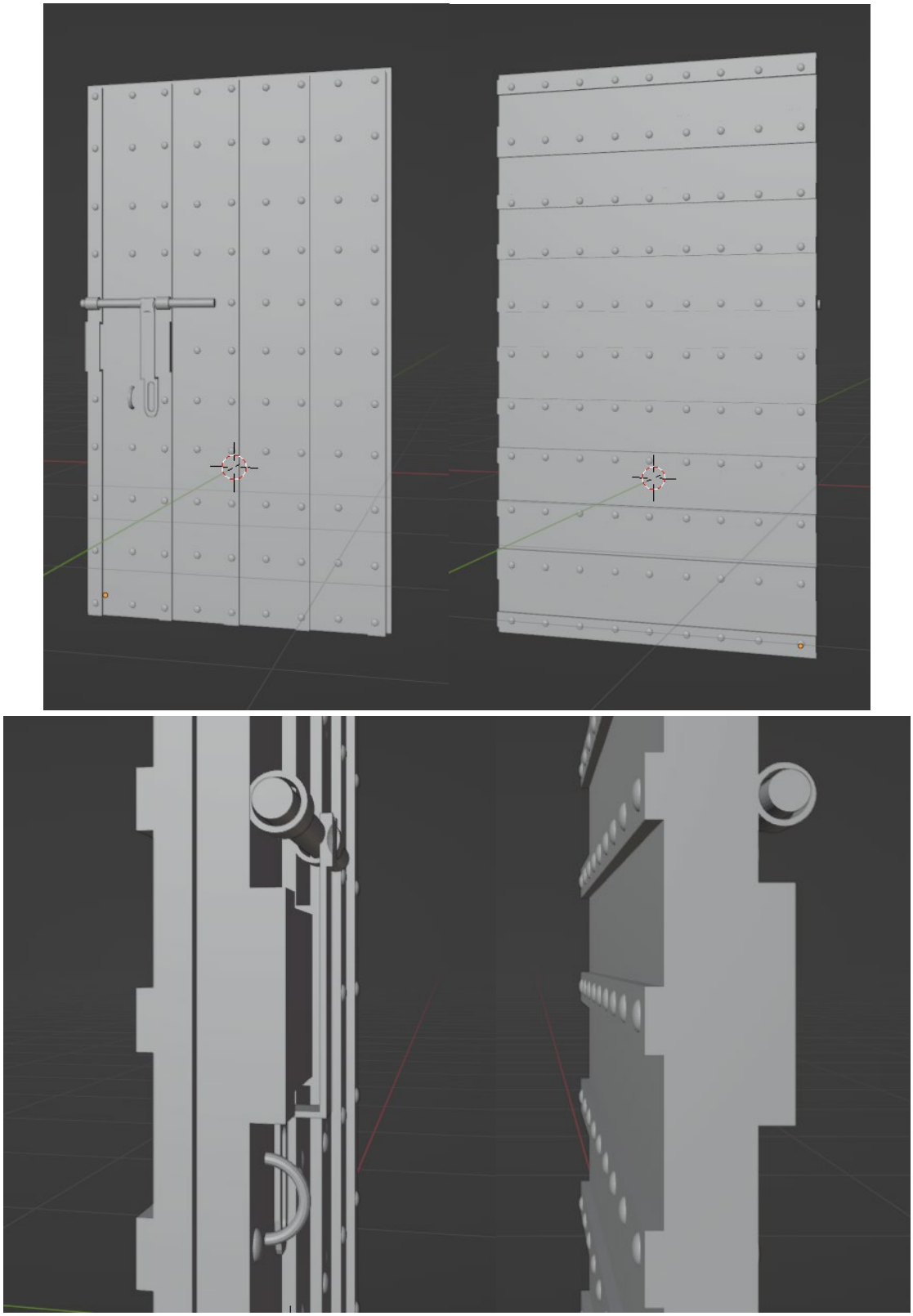


Figure JP.13: front, back, and sides of the complete door

Although not perfect, Dr. Giroux was satisfied with the progress so far and said that the general population would not be checking how precise this model is.

Prison Door

Similarly, the prison door was designed. The difference with this one, however, was that there was no reference in the specification document provided by Dr. Giroux. Looking solely at the picture taken, the picture was adjusted through a program called fspy (Figure JP.14).

Figure JP.14: defining x and z axis through fspy

The program allows users to input an image file and after making the adjustments save as a .fspy file which then can be imported through blender. Upon making the import, a new camera will be created along with the image directly facing it so that the angle of the camera and the image will match the three axes along the grids (Figure JP.15).

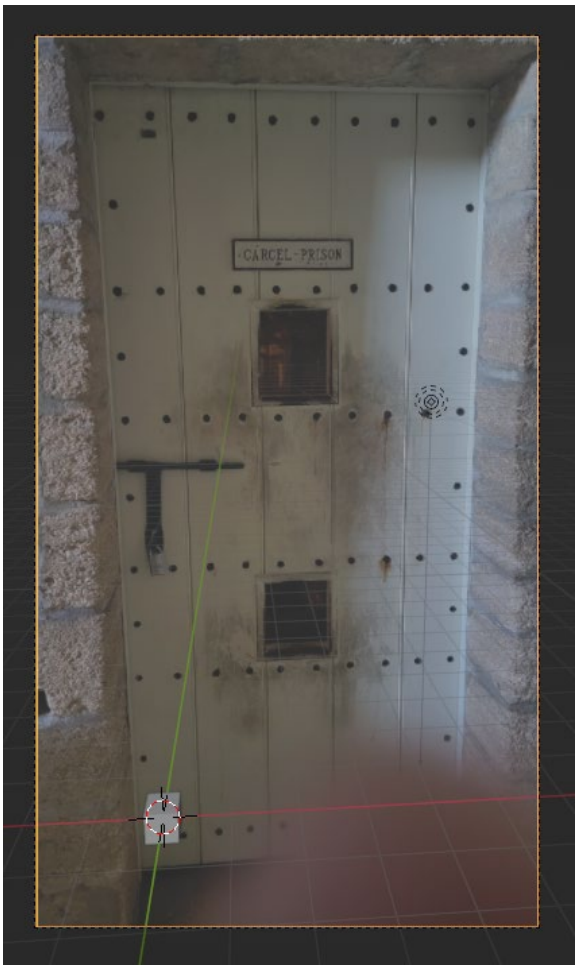


Figure JP.15: it can be noted that the x axis (red) matches the bottom of the door

Because the image disappears in any other perspectives and since seeing the image in other perspectives will not help regardless, most of the modeling was done strictly in this perspective (Figure JP.16).

Figure JP.16: the x and z axes of the image match the adjusted cube

Once the default cube was fitted to the outside of the door, all other elements such as the bolts and lock handles had to be first adjusted along the y-axis in relation to the door as the appearing size of the objects also changed depending on its proximity to the camera - in other perspectives such as the front orthographic, objects appear the same size regardless of their positioning as the view is two dimensional.

because the axis alignment was not perfect, following the image fully was difficult. It was also difficult to know how thick objects were as the only usable reference for the door was this image, meaning there were no other perspectives, so general thicknesses were used.
Applying boolean difference proved to be troublesome as well, which led to creating new faces to cover unexpected gaps, and the image itself was dark around the windows making it hard to decide where the boolean difference would begin. See Figure JP.17 for the complete door.

Figure JP.17: Wireframe and solid view of the prison door

Figure JP.17 continued

Chapel Door

After roughly finishing the prison door design, The chapel door was modeled. Because fspy did not have a zoom-in feature to precisely determine the axes, The built-in magnifier program was used to zoom my laptop screen to 200% (Figure JP.18).

Figure JP.18: Using magnifier for fspy

Once the cube was adjusted to the bottom, left and right of the door, the vertices of the top were scaled to the beginning of the curve of the door. Then the vertices were extruded till it covered the circular top. Then this segment was subdivided eight times (Figure JP.19).


Figure JP.19: Visual illustration of the subdivision on the top of the door

Each subdivided loop was then scaled to match the image. The very tip, however, did not look smooth no matter how much I scaled it (Figure JP.20).

Figure JP.20: an example of scaled tip

After some experimenting, the top segment was further subdivided but the tip remained sharp regardless (Figure JP.21).

Figure JP.21: further subdivision and the top vertices were merged

It was concluded that the best solution was to bring the top down a little so that a little piece of the image would be uncovered (Figure JP.22).

Figure JP.22: It can be seen that the tip of the image is slightly uncovered

Then all the unnecessary edges were removed to reduce as much polycount as possible (Figure JP.23).

Figure JP.23: Wireframe view of the door at this stage

The blueprint of the door was matched to the front next (Figure JP.24).

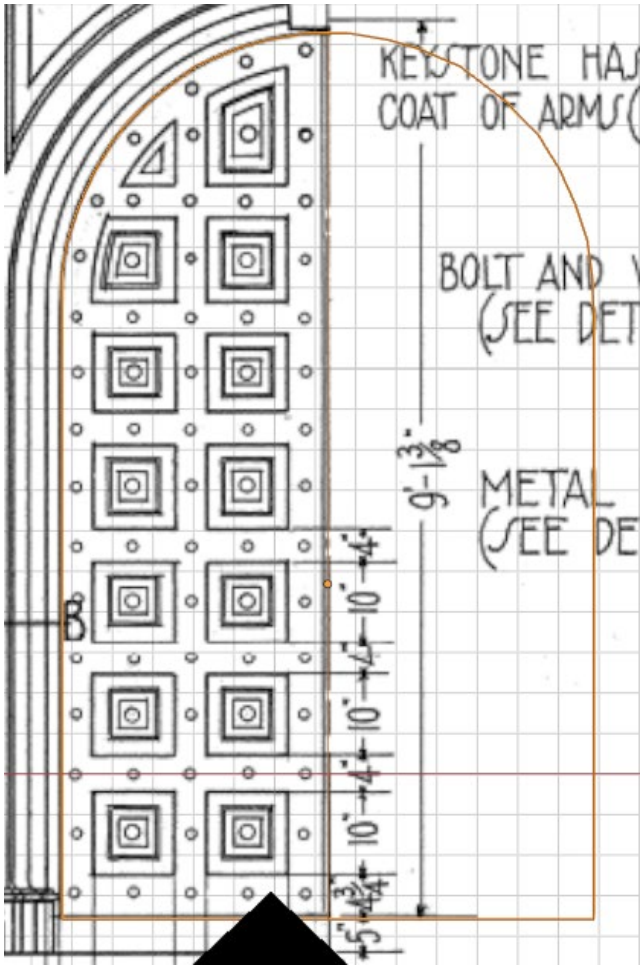


Figure JP.24: front orthographic view of the door

To design the multiple square frames attached to the door, the square frame was scaled, roughly designed off the image, then fitted to the blueprint to compare (Figure JP.25).

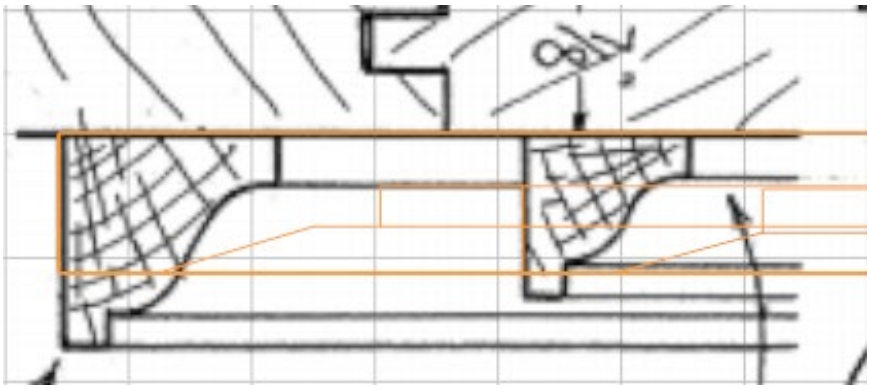


Figure JP.25: It can be seen that the rough design does not match the blueprint

Here it was decided to follow the blueprint instead of the picture taken at the fort and a single subdivision loop was created to further match the curves seen on the blueprint (Figure JP.26, 27).

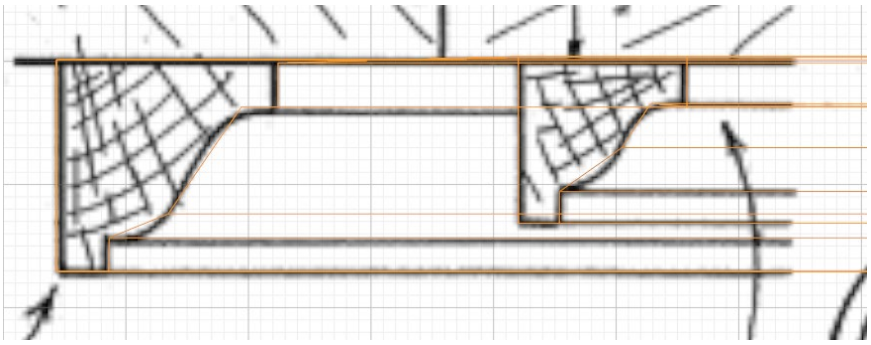


Figure JP.26: The fitted frames against the blueprint

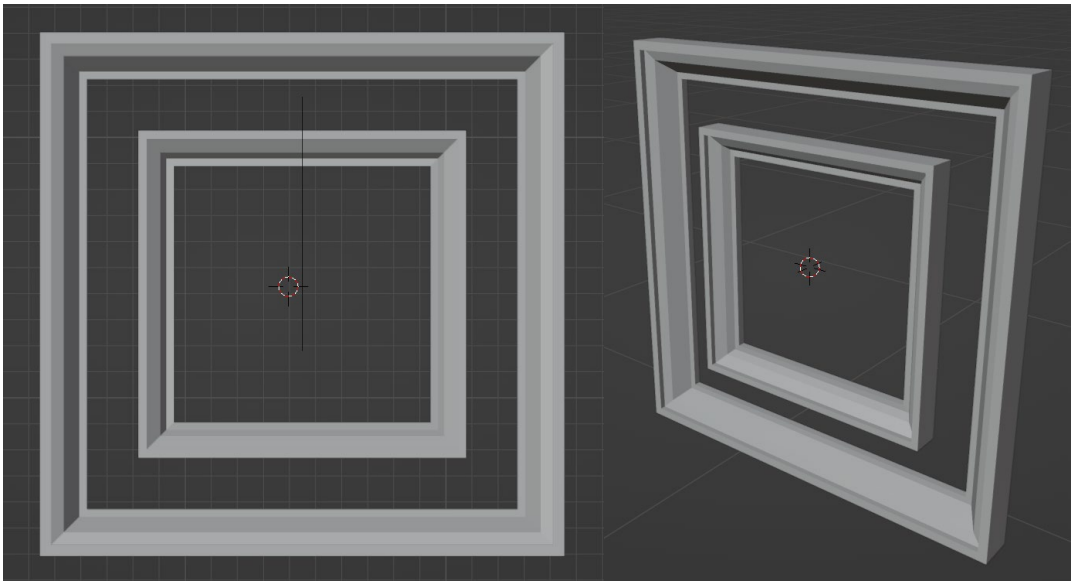


Figure JP.27: Front orthographic and user perspective of the complete frames

The ‘flower’ was designed next. A cylinder was scaled to fit the image, then the fitted cylinder was extruded to build the ‘flower’ based on the blueprint (Figure JP.28).

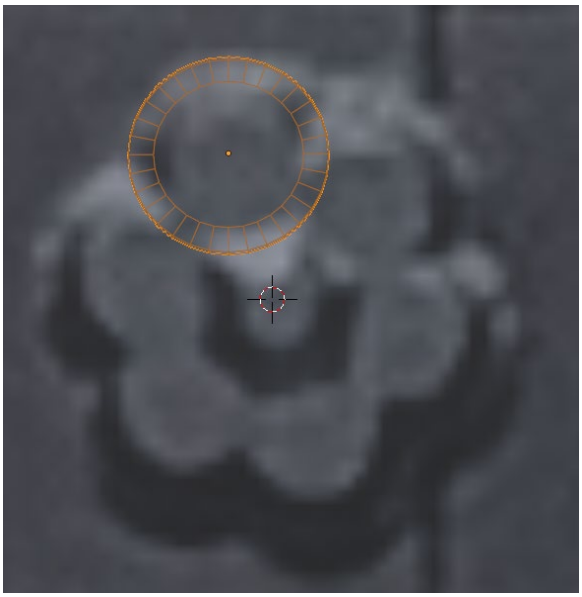


Figure JP.28: Simple fitting of cylinder to the image with additional extrusion

While in the right orthographic view, it was nearly impossible to see how much the vertices are scaling with the blueprint on the background as well. So the blueprint was loaded in the front orthographic view to visualize the scaling. Similarly, it is impossible to see how much the vertices are scaling forward in the front orthographic view so these views were consistently swapped to complete the model (Figure JP.29).

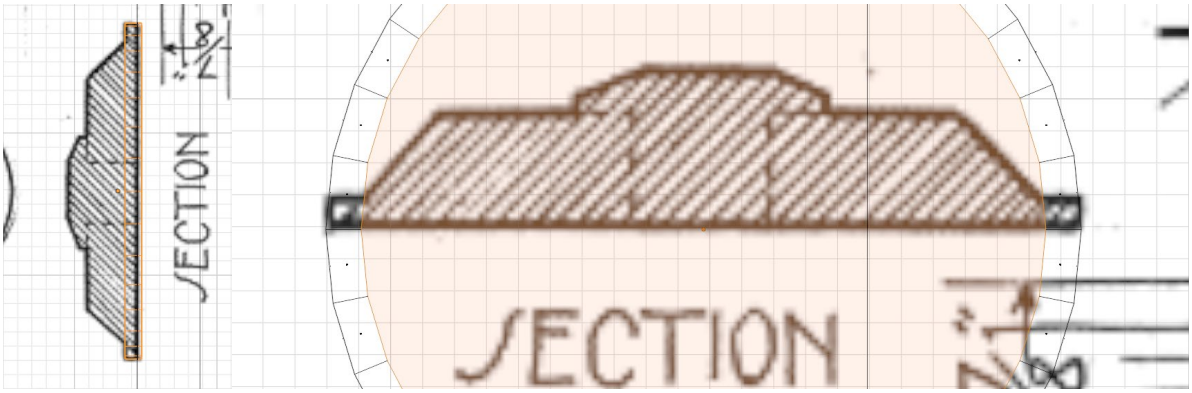


Figure JP.29: Right orthographic vs front orthographic view of the cylinder while modeling

Once the blueprint was completed, a mistake was recognized: the ‘flower’ was modeled with a single sphere instead of six separate spheres (Figure JP.30).

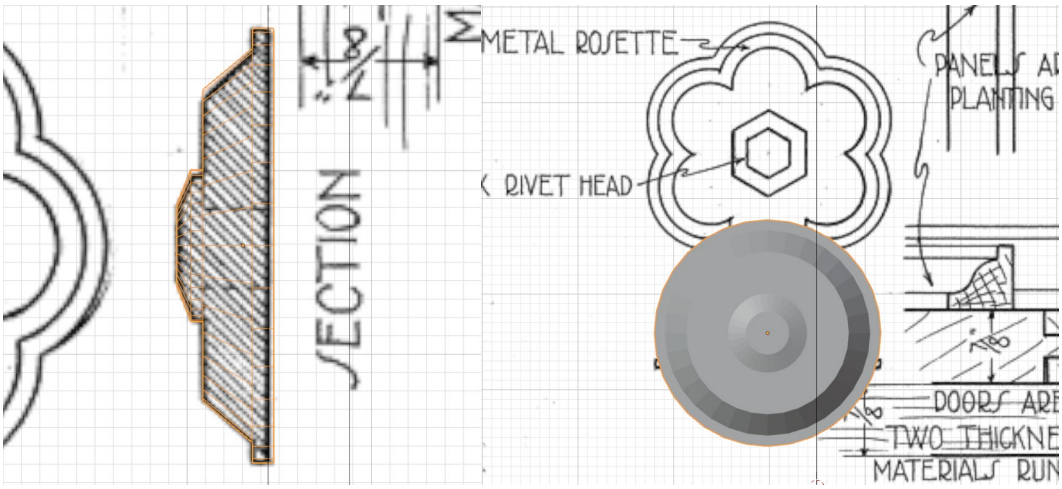


Figure JP.30: fitted cylinder to the blueprint and comparing it to the desired model

Although the cylinder's radius may be significantly smaller, its height is still identical so the blueprint was used to extract the cylinder’s height. One unfortunate error that was spotted recently however was that as the view was being switched between the front orthographic view and the right orthographic view, the scaling to fit the one of six cylinders disfigured the radius of the top of the cylinder (Figure JP.31). This error will be adjusted shortly.

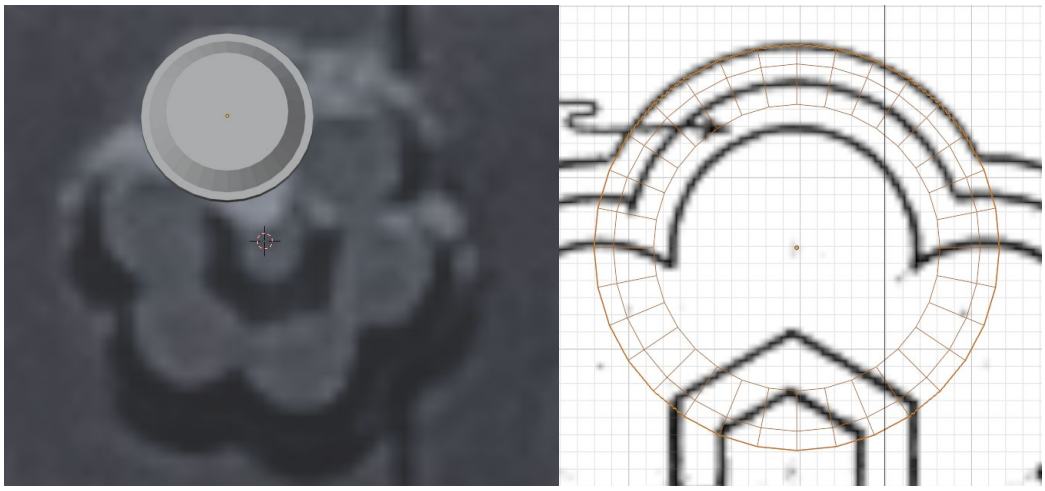


Figure JP.31: It can be seen that the top radius does not match the blueprint

At the time this issue was ignored and the cylinders were duplicated and fitted to the blueprint (Figure JP.32).

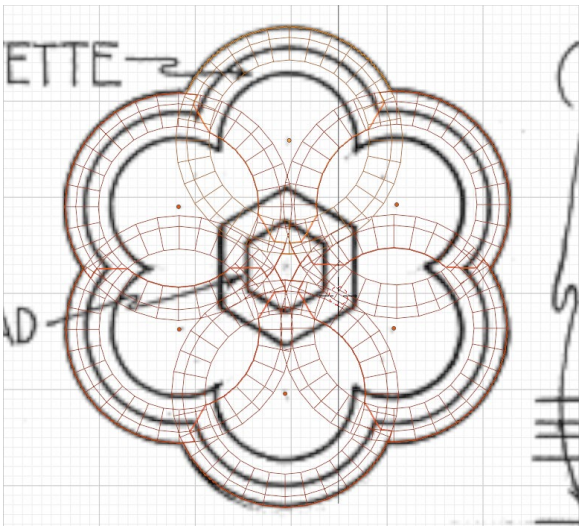


Figure JP.32: Illustration of fitting the cylinders

Then the unnecessary vertices and edges were removed, creating an empty surface. This surface was filled by selecting the vertices that the new face would cover. Initially all top vertices were selected to execute this function but it caused unwanted areas to be covered (Figure JP.33).

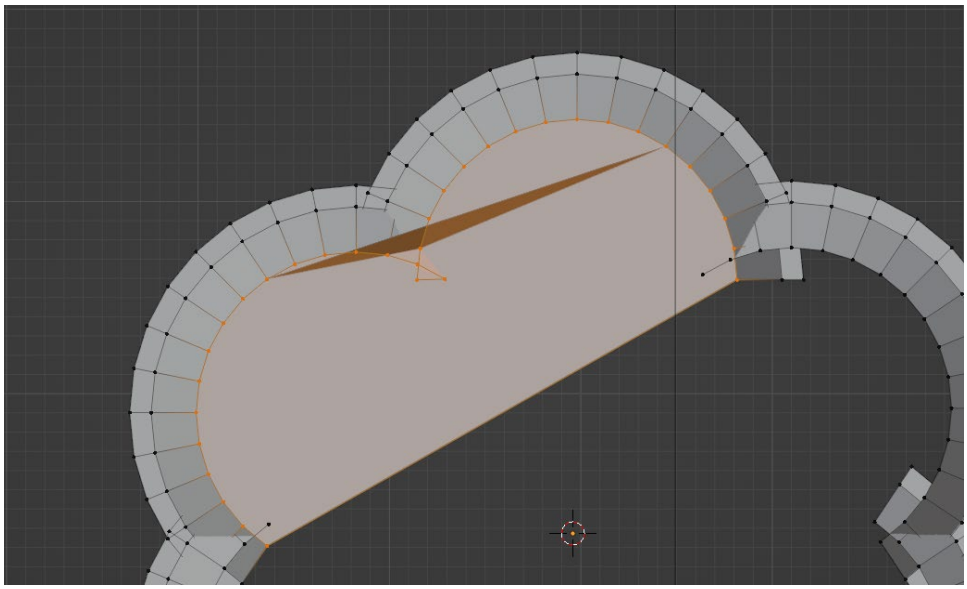


Figure JP.33: Blender's computer calculated covered surfaces

Thus, each portion of the circle was filled individually and then the vertices roughly making the hexagon were selected to fill the remaining space. Similarly, however, this did not fill the entire hexagon (Figure JP.34) so the hexagon was covered by selecting four vertices at a time to create two trapezoids, one on top and one on the bottom (Figure JP.35, 36).

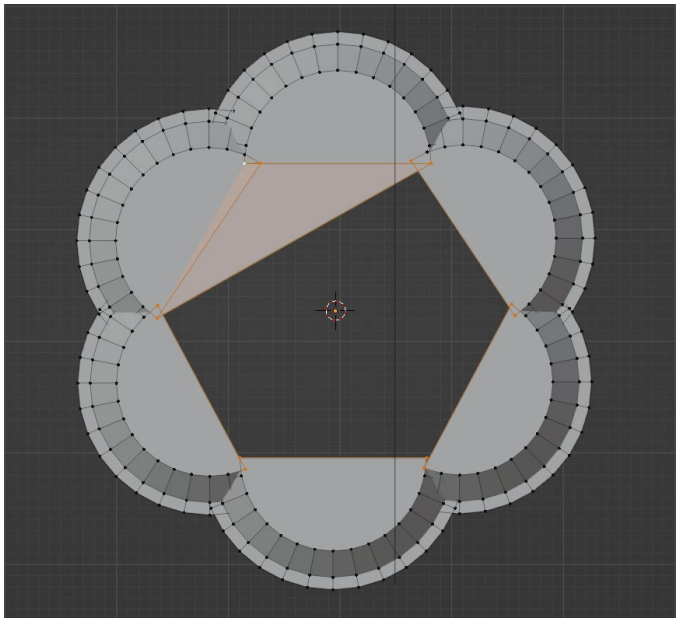


Figure JP.34: Attempting filling the empty hexagon. It can be noted that the vertices around the hexagon are selected.

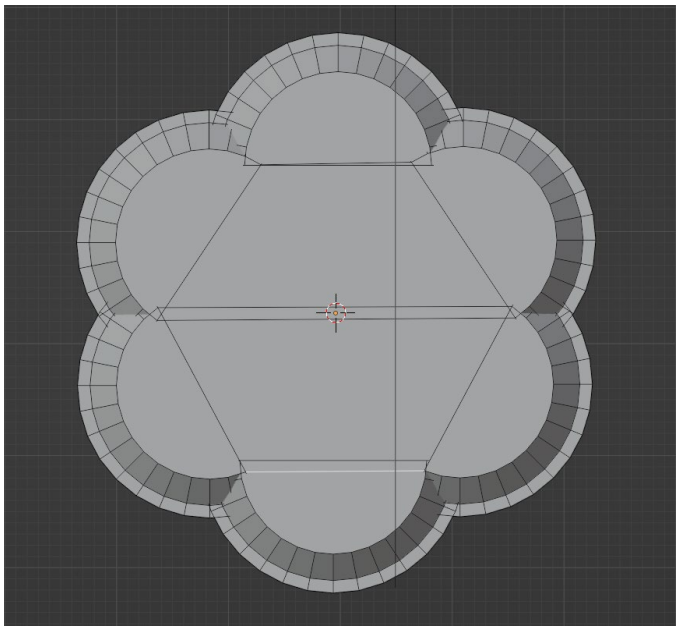


Figure JP.35: Two trapezoids cover the hexagon

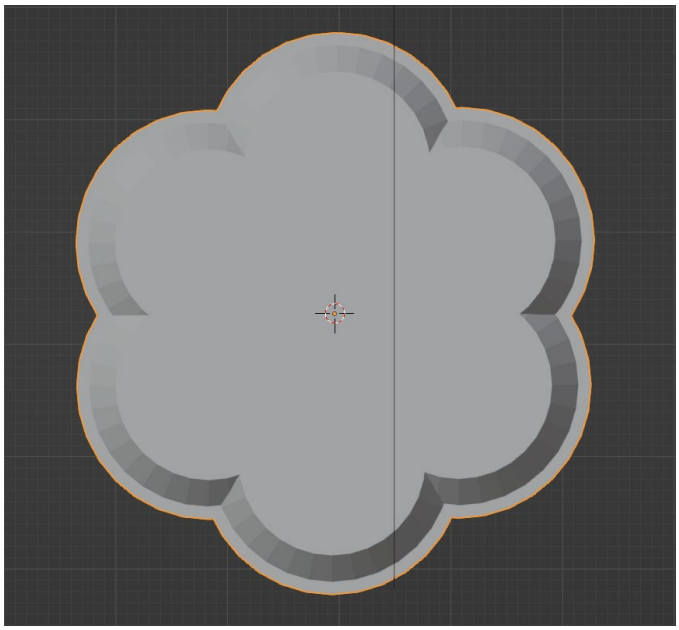


Figure JP.36: The model in object mode

Although the model looks clean, more poly could have been removed so additional edges were removed and the faces were refilled selecting the most optimal vertices (Figure JP.37).

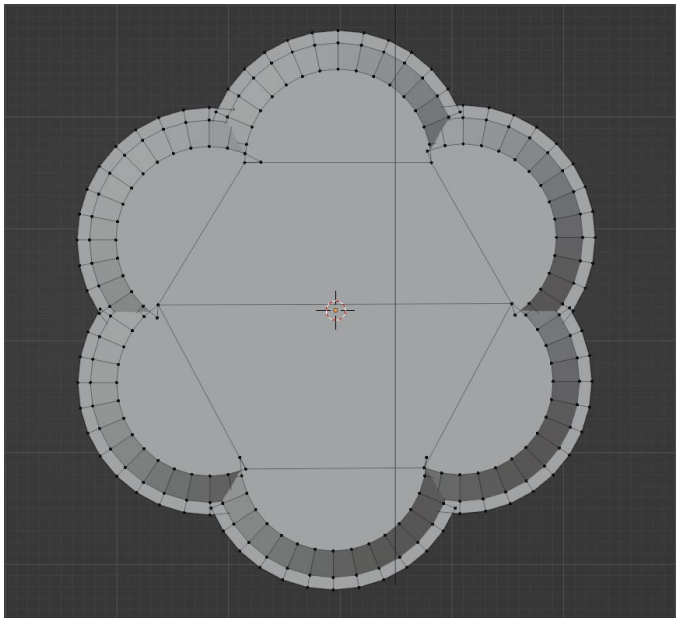


Figure JP.37: it can be noted that the additional edges have been removed successfully

The chapel door will be completed in the next few weeks.

Spanish Treasury Door Trim

Looking back at the first modeled door, it can be noted that while the blueprint did not have a handle in the back, the image taken did (Figure JP.38).

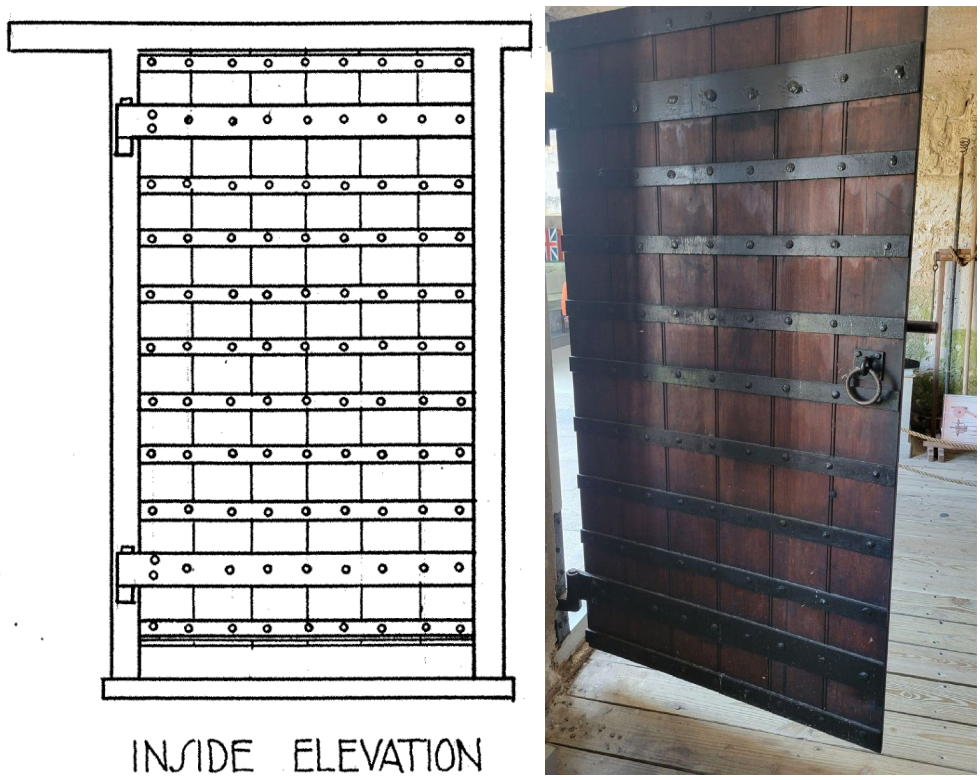


Figure JP.38: Blueprint vs image taken of the back of the door

So using fspy, the handle was created (Figure JP.39) and the two strips attaching to the trim were extended.

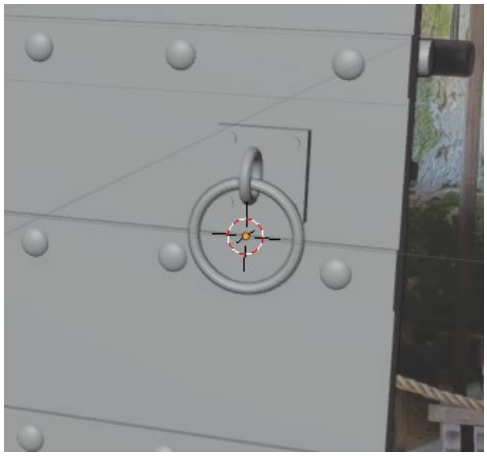


Figure JP.39: handle on the back of the door

Once the handle was complete, the area where the door relies on its swing on was focused (Figure JP.40).

Figure JP.40: the image being modeled

A cylinder was created with adequate thickness and joined together with the strip (Figure JP.41, 42).

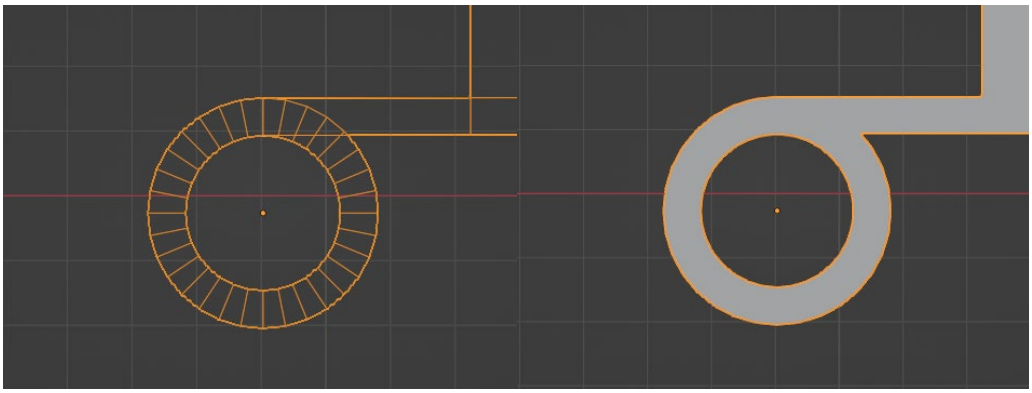


Figure JP.41: Wireframe vs solid view of the joined area. Note that the center of the door is in the center of the cylinder

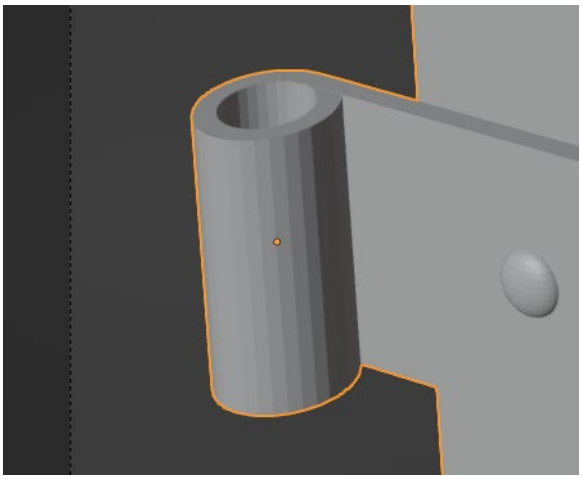


Figure JP.42: User perspective of the joined area

The trim of the door was designed next and an additional cylinder was created to fit the previous mesh (Figure 44). Because this cylinder attaches to the trim, the cylinder was loop cut then the bottom of the cylinder was halved. The four vertices that were exposed from the removal were then extended till it reached the trim (Figure JP.43) and scaled to match the image.

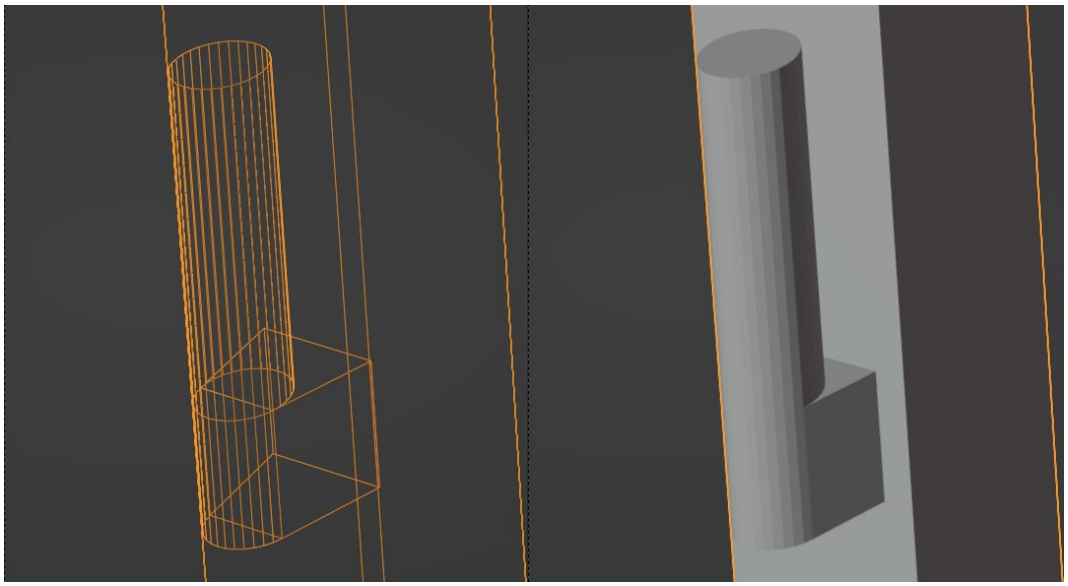


Figure JP.43: Wireframe vs solid view of the cylinder

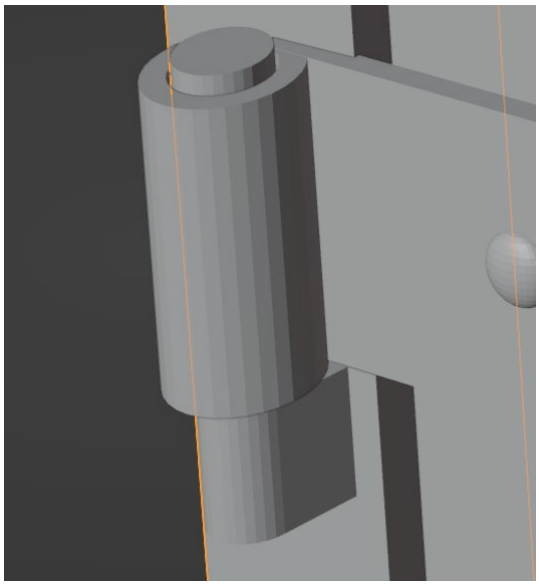


Figure JP.44: The designed cylinder fits the door

Although the opening animation will be done in Unity, the center of the door makes it easy to swing the door as well (Figure JP.45).

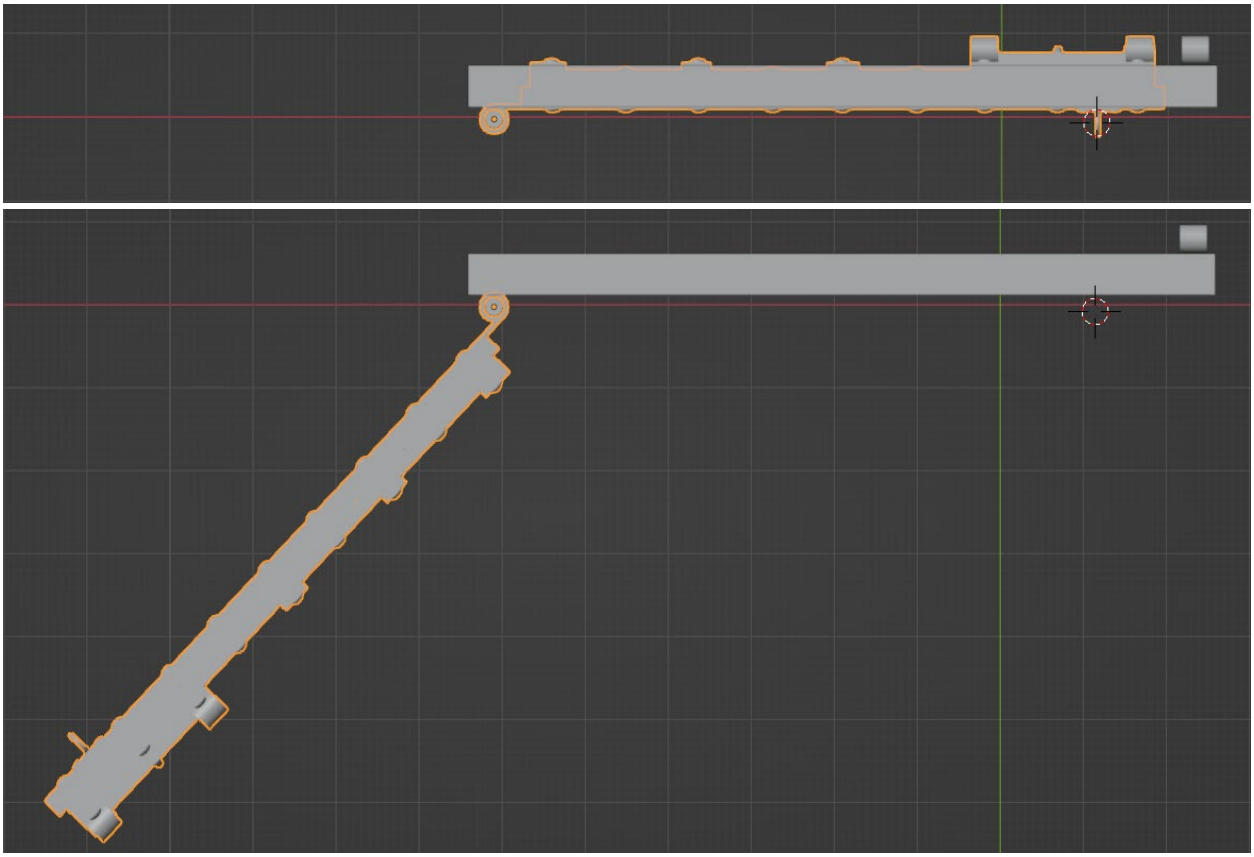


Figure JP.45: rotating the door with the center on the cylinder

9.5. Enrique Rodriguez

I was designated as the project manager during our early group meetings and was tasked with creating a set weekly meeting schedule and place to discuss. We settled on meeting in person after class each day and online on Friday. I created a Discord server where we could talk, save assets/references, tutorials, and record meeting logs. Afterwards, I got in contact with the sponsor to get them on the server and set up a bi-weekly meeting time with them to update them on our progress.

As for system work, I set up the original design document formatting and early sections we used for our early assignment submissions. As project manager, the sections I focused on were related to the user experience and interactions to ensure we deliver a sufficient quality project at the end of Fall 2022. This included setting a set list of goals and objectives and complete as well as setting formal constraints to mandate the guidelines we need to follow and complete. Connected to this, I created early block diagrams for our project that showcased how the user can experience the fort and what classifications each interactable object in the fort had in relation to designing, researching, and prototyping. Lastly, I set up our Gantt chart moving forwards that outlined what we initially planned to complete by Fall 2022.

As for my technical labor, I was tasked with modeling and learning about interactable objects in Unity, both for desktop and VR. I created prototypes for the doors and drawbridges the fort contains and worked on creating their various VR counterparts. Afterwards, I worked on various smaller assets such as display boards of varying sizes and infographics. These infographics were made using constraint based designing and are easily modifiable (Figure ER.1). This was done to ensure easy modeling for future sized displays.

Notice how with constraint based design, simply by making the design longer, adjustments in the legs positioning are made automatically. Although not necessary, this saves a lot of time on the side of the modeler (Figure ER.2).

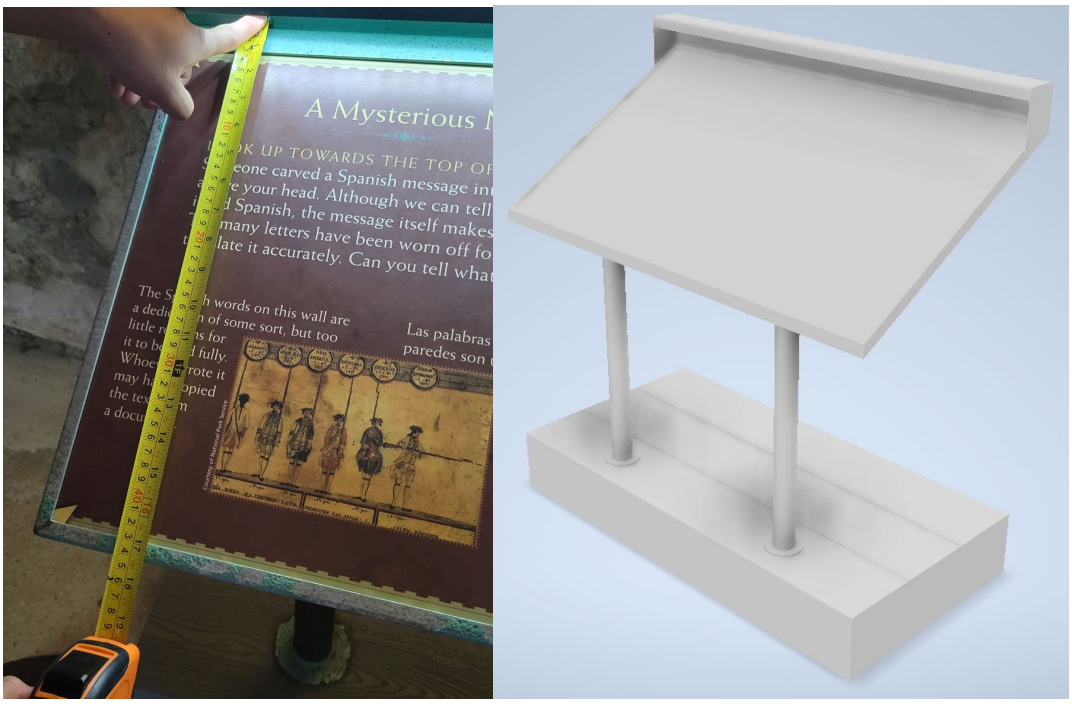


Figure ER.1: A comparison between the infographics in real life and their digital recreations.

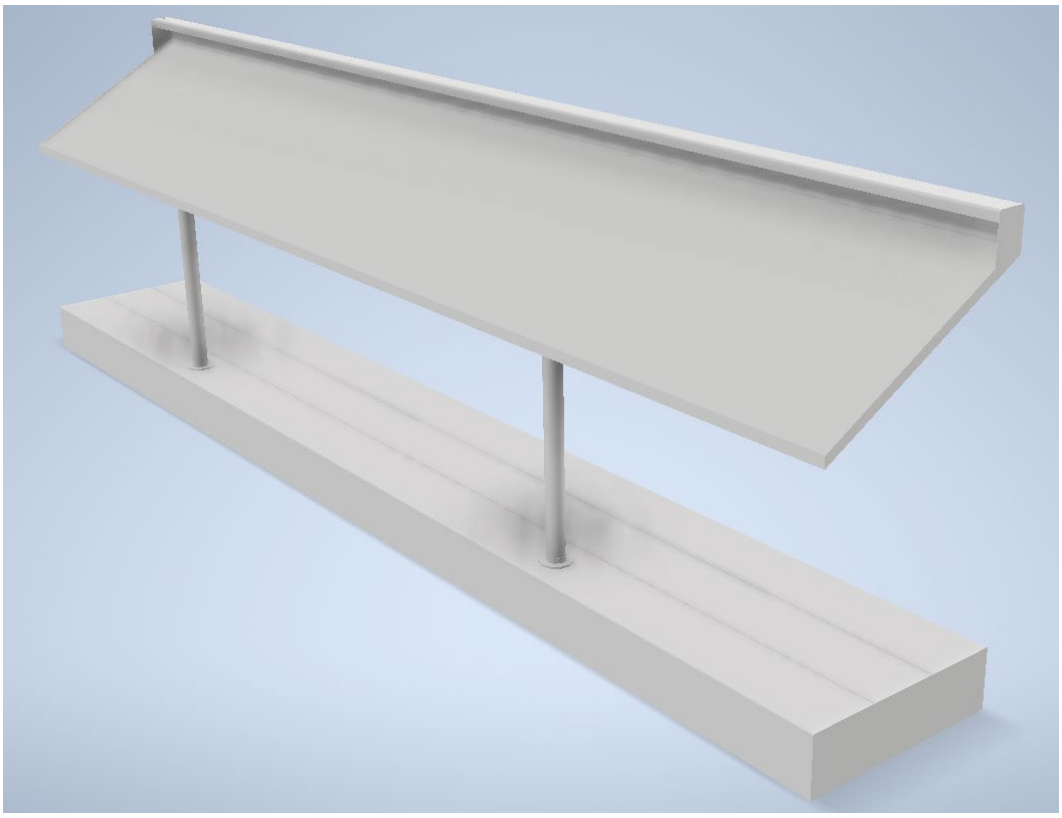


Figure ER.2: An example of a longer infographic.

These infographic pieces would be displayed around the fort and present the user with historical information after a user walks within a certain proximity to it using a trigger (Figure ER.3).

Figure ER.3: A comparison between the visual infographics in real life and their digital recreations.

Doors are also opened and closed via triggers for ease of access, though this can be disabled and enabled if the user wants to open them manually (Figure ER.4 & Figure ER.5). Triggers are specific areas that cause an action to occur when something happens within its area, in this case, that occurrence would be the player walking into its proximity. Once the player enters the trigger, the door is opened, some doors open in 1 direction while others can swing in the direction the door is approached in. The drawbridge to the fort works in the same way, as it is just the door system rotated 90 degrees in the Y direction.

As for their VR counterparts, the doors will work based on a hinge system. The further the user places their hands from the hinge on the door, the easier the door is to open. The only problem with this approach is that all doors have the same

force to open them which doesn't make sense as some doors are heavier than others to open. Testing will need to be done to see if this affects the user experience.

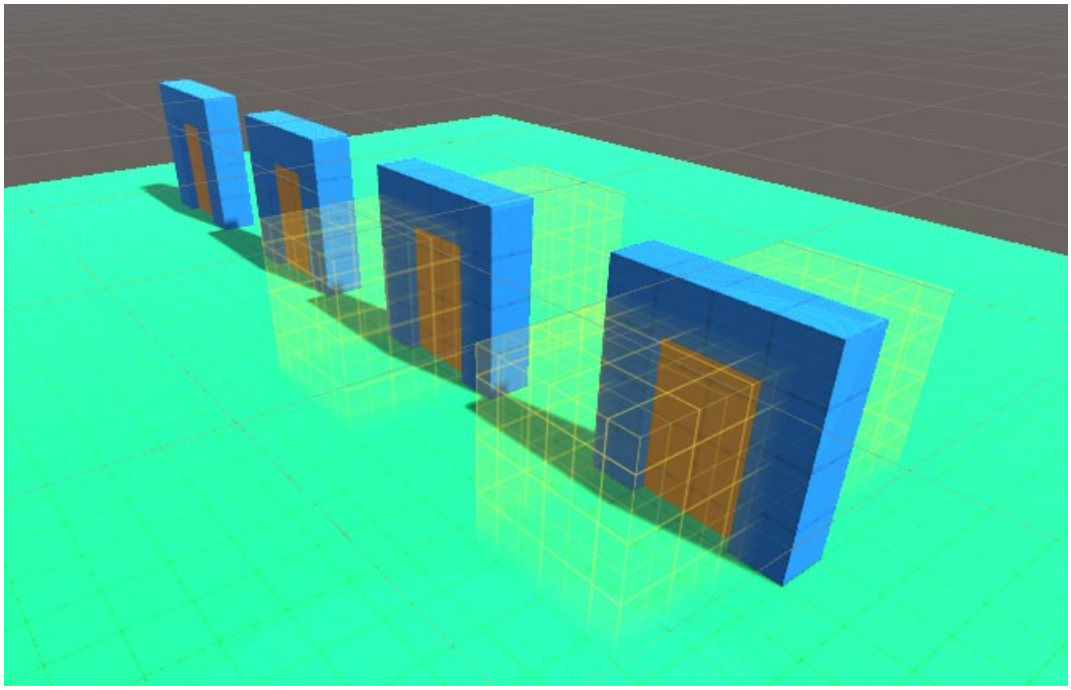


Figure ER.4: The yellow box throughout the two right doors indicates the trigger.

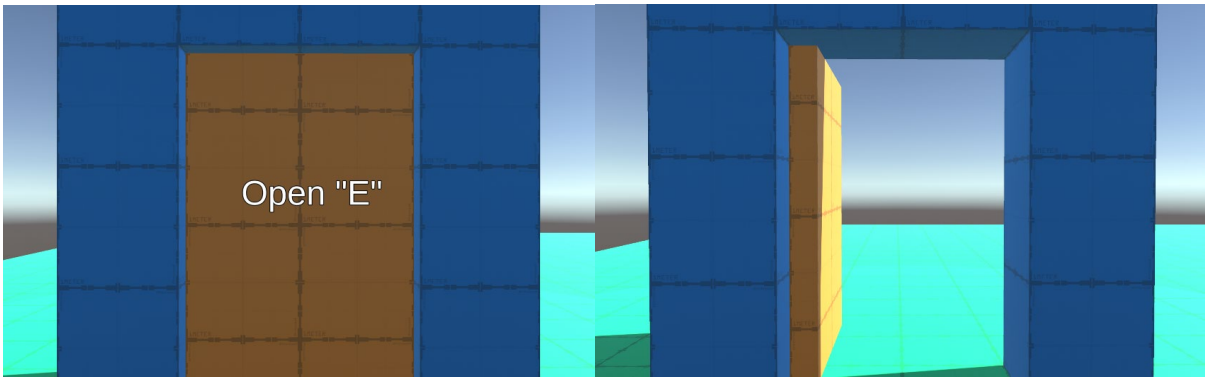


Figure ER.5: The manual version works without triggers and requires user input to open/interact.

These triggers can also be applied to other assets. A fort tour is planned to be implemented and these triggers could allow for progression in the tour. Once the

player walks into the trigger, the tour guide would progress in their explanations and tour of the fort. Triggers are also applied to the various display boards and infographics around the fort to pop up the information displayed. This change also draws in the users attention to the display boards so they spend more time learning and exploring. Various triggers can also be placed around the map explaining broader features like the surrounding area and other locations off in the distance (Figure ER.6).

Figure ER.6: Displays such as this would have information prompts appear upon entering a trigger. (Not Final)

One of the locations we plan on including triggers regardless of user options would be the drawbridge. As the drawbridge has to be lowered at all times to stop the player from falling off the bridge, it's crucial that this bridge is always available to the player while still showing the player some sort of interactivity. The animation trigger that the drawbridge interacts with also has multiple parts of the fort that get

interacted with including the chains that connect to it as well as the mechanism that lowers it (Figure ER.7).

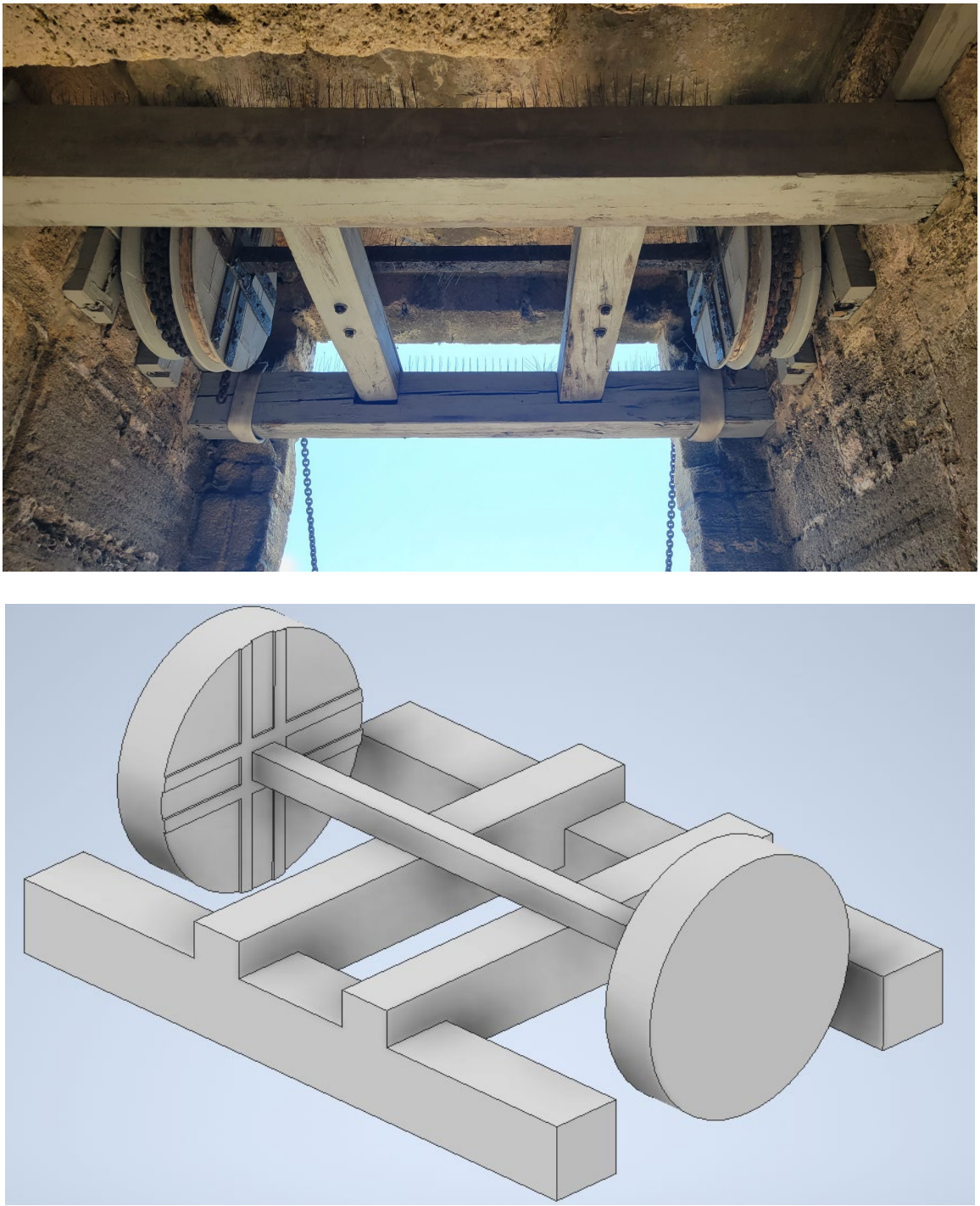


Figure ER.7: Initial status of Drawbridge windup mechanism. (Not Final)

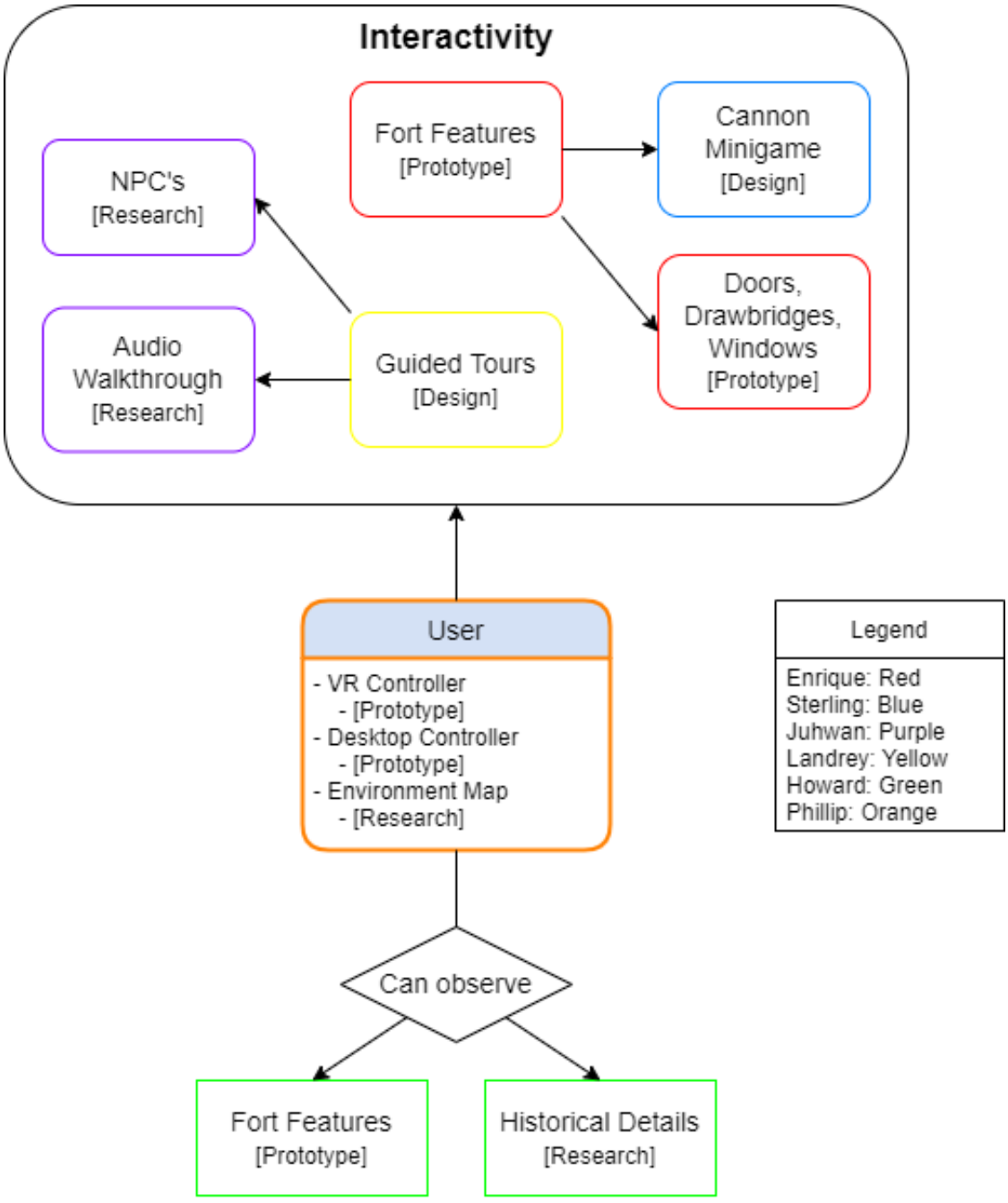
9.6. Phillip Zou

Initially, I researched the broader impacts of the project as well as VR implementation. One of our project requirements was to ensure support for the VR headsets: Oculus Quest 2, Oculus Rift, Valve Index, and HTC VIVE. Our initial concerns were whether we could develop for all of the headsets at once or develop for each headset individually. Furthermore, can we develop a normal 3D game alongside the VR for broader appeal and accessibility, as VR headsets are very expensive. I looked into setting up the environment in Unity for VR development as well as installing the necessary plugins and frameworks.

I set up the VR framework for the project and developed a sample scene in VR with a few interactable objects for testing. Since I do not own a VR headset, I also researched testing VR in Unity without a headset and scripted the basic testing interactions to use within the Unity editor. I then went on to figure out if we had to develop separate scenes to account for the different input or if we could simply add the framework for mouse and keyboard and game controllers to our current VR scene. From there I developed a prototype script that would detect the input device and provide the correct controls and camera for the player in the same scene so we would not have to port our models to a different scene and redo everything for mouse and keyboard.

We then came together as a group to visit the Fort and gather data and pictures for reference. Afterwards, I took primarily a research role focusing on VR implementation, creating textures, and optimization to meet our performance requirements.

10. Block Diagram



11. Project Budget

Building Castillo de San Marcos for a Virtual Reality experience should not incur many expenses, if any. We are working alongside the U.S. National Park Service, so we will have access to be able to visit the fort as needed, free of entry charge. As students, all six of us working on this project have been able to download and use Unity for free. We will be using two different 3D modeling software: Blender and AutoDesk Inventor. Blender is free to access and use. AutoDesk allows students to utilize their software for free if they provide proof of being enrolled. This allows the team to model using Inventor under a student account and will not require us to pay.

The team will not have to purchase any VR headsets, as we have plenty at our disposal for testing purposes. Two team members have an Oculus Quest 2, and we have access to rent/borrow HTC VIVEs, an Oculus Rift, and a Valve Index if needed.

The only foreseeable costs we may incur would be assets on Unity for textures and props for the 3D modeling of the fort, as well as costs we could incur while physically visiting the fort. We do have access to some funding through our sponsor in the case that we need to use some assets that cost money. However, we will be attempting to create the assets we need and cannot find for free on the asset store. The team will be taking a trip to visit the fort for research, as well as taking pictures and scans for textures that we will not be able to find. Although touring the fort would not cost the team any money, we would have to pay for the gas to drive there, as well as pay for parking. Altogether, we are aiming to spend no more than $299 on this project.

12. Research

12.1. Castillo de San Marcos History

The city of St. Augustine goes all the way back to the early 16th century and has played a part in many pivotal parts in history. There have been a total of 5 flags flown over the city: Spain, France, Great Britain, the United States, and the Confederate States of America. These flags were flown under the different forces that had control over the city, and therefore, the fort at the time. The history behind Castillo de San Marcos follows closely with the entire city/province of St.

Augustine, so the only way to truly know the history is to know the entire story behind the City as well.

12.1.1. The First Spanish Period

The first flag flown over Castillo de San Marcos belonged to the Spanish Forces. This flag, shown in Figure 7.1.1.1, is known as the Andrew flag, or the Cross of Burgundy.

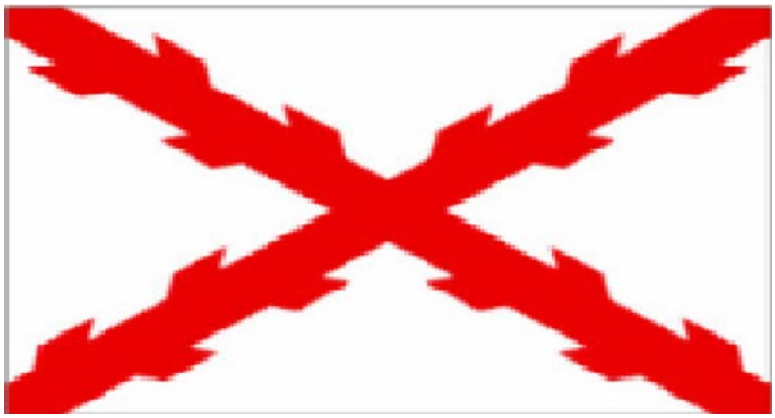


Figure 12.1.1.1: The Spanish Cross of Burgundy flag

In the year of 1513, Juan Ponce de León, a Spanish conquistador, accidentally discovered new territory, which he later named Florida. He claims the land for Spain, however, all attempts to colonize the land over the following 51 years fail. Finally in 1565, Don Pedro Menéndez de Avilés and his fleet of 600 people successfully landed in Florida. Under the rule of King Phillip II in Spain, Menéndez and his crew successfully got rid of a French settlement nearby and began to establish a military outpost for Spain [1b]. Menéndez officially names the new settlement San Augustín, and the city has hopes for further expansion of Spain to the new area.

Between the years of 1665 and 1672, the Spaniards colonizing San Augustín were under attack several times. Between Native American tribes, American pirates, and more, the city was under constant destruction and repair. In 1672, the governor at the time, Manuel Cendoya, decided that a more permanent and lasting fort was necessary to retain control over the area. Prior to this, they had been utilizing wooden structures, which were

either burnt down or in horrible condition at this point. This new fortress took an entire 20 years to build.

This new and improved Castillo de San Marcos is the exact fortress that stands in St. Augustine today. Over time, it has been worn down, and minor repairs have been made by Parks Services, who currently have ownership of the structure, to retain the integrity of the structure as a place people can visit. What cannot be seen today is the white plastered walls and red painted trim around the entire structure, except minor remnants from place to place. In Figure 7.1.1.2, there is a computerized model of what Castillo de San Marcos would have looked like in the late 1600s.

Figure 12.1.1.2: Model of what Castillo de San Marcos originally looked like

Since this structure was completed in 1692, no armed forces were able to take the fort by force despite many attempts. In the centuries following, however, Florida has been signed over to different forces peacefully. This is the reasoning behind so many forces having control over the area despite the fort never being overtaken in battle.

12.1.2. The British Period

The second flag flown over Castillo de San Marcos belonged to the British Forces. This flag, shown in Figure 7.1.2.1, is known by three different names: the flag of Great Britain, King's Colors, and the Union Jack.

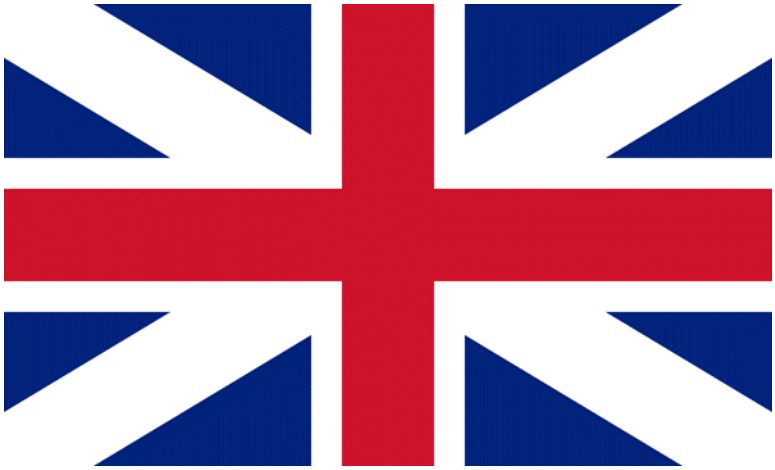


Figure 12.1.2.1: The flag of Great Britain

The Seven Years' War (also known as the French and Indian War), which took place between 1756 and 1763 was the first instance of Florida being signed over to new rule. Since Castillo de San Marcos and the city of St. Augustine were in Florida, they were no longer under Spanish rule. The First Treaty of Paris was signed in 1763 to end the war. Comprised within the treaty was granting the entire province of Florida to be under the rule of Great Britain [2b].

The Spanish settlers in St. Augustine did not like the new change in power, so they packed everything and left for Cuba. This hit to the population did not last long, as the new population soon doubled what was there prior to when the Spanish settlers left. This rise in population led to prosperity in the area. The city was built up for commerce and trade [2b] and renamed to Fort St. Marks.

Trouble began to rise between the colonies and Great Britain. This all started when Great Britain began to impose taxes upon American colonists. The Stamp Act of 1765 was the first tax placed by the British Parliament. The Stamp Act taxed the colonists on all paper documents within the colonies. This soon led to rebellion from the settlers in America, who went to great lengths, even violence, to protest it [5].

These troubles between the colonies and Great Britain were only the beginning of the American Revolutionary War. This lasted from 1775 to 1783 [6]. Tensions rose over the years from several more taxes imposed upon the colonists from Parliament, as well as protests that turned violent from British soldiers. This led to the colonies wanting full independence from their motherland of Britain.

St. Augustine became the British capital of East Florida and quickly became a command post and military stronghold for the British. The town was quickly filled with militarized troops to aid during these times of rebellion from colonists. The city was a haven for British loyalists and the fort itself became a camp to contain prisoners of war during the American Revolution.

Finally, in 1783, the Second Treaty of Paris was signed. This declared independence for all colonies north of Florida. Florida itself was signed back over to the control of Spain due to their assistance during the war.

12.1.3. The Second Spanish Period

The third flag flown over Castillo de San Marcos belonged to Spanish Forces yet again. This flag, shown in Figure 7.1.3.1, is known as the War Ensign Spanish National flag.

Figure 12.1.3.1: The War Ensign national flag of Spain

Due to the many years that had passed since the last time Spain had control over Florida, a lot had changed. They were in the beginning of a massive war, which would be called the Peninsular War. Spain could not dedicate much towards Florida or any other colonies in the West.

The now independent Americans noticed Spain's lack of interest in Florida and decided they wanted to expand their control. Through peaceful negotiation, Spain signed Florida over to the United States through the Adams-Onîs treaty in 1821 [7].

12.1.4. The American Period

The fourth flag flown over Castillo de San Marcos belonged to Confederate Forces. This flag, shown in Figure 12.1.4.1, is known as the Stars and Bars.

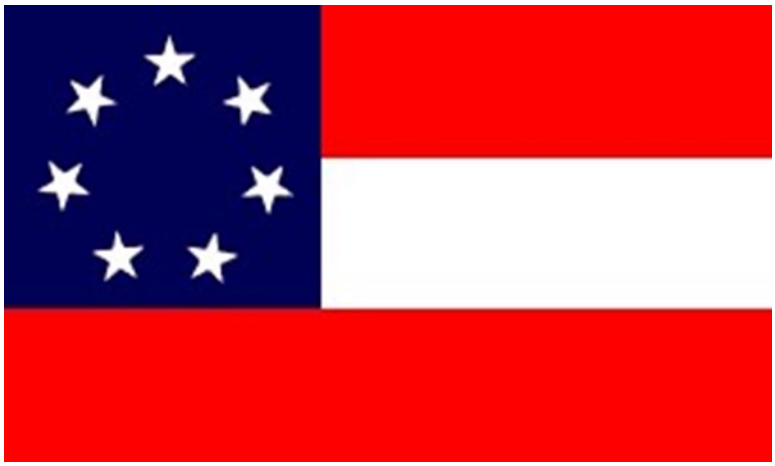


Figure 12.1.4.1: Confederate Stars and bars flag

The fort was renamed to Fort Marion during the Second Seminole War, which occurred during the years from 1835 and 1842. This war was preceded by the Indian Removal Act in 1830 and some of the Seminole Indians in Florida refusing to relocate to Indian Territory. Yet again, the fort was militarized, this time by the United States Army. The war was concluded through both sides backing off. Many remaining Seminoles fled out of the area, but remained in Florida, and the United States Army gave up on removing them from the territory [8].

Shortly after the Second Seminole War concluded, Florida was officially named a state within the United States. It had been a United States territory from the acquisition in 1821 until 1845.

The election of Abraham Lincoln in 1860 led to the succession of several states from the Union, including Florida. Thus began the Civil War. At the time, Fort Marion was not used much aside as a warehouse for old weapons but was immediately reoccupied. The location was utilized as a transit line and method of getting essential materials in, since the harbor was easily protected.

By 1862, Federal forces were able to reclaim St. Augustine with ease due to lack of military resistance in the area. The remaining Confederate supporters in the city were quickly dealt with threats of arrest. The Union was able to defend and retain control over the area through the rest of the Civil war, until 1865 [9].

The fifth and final flag flown over Castillo de San Marcos belonged to American Forces. This flag, shown in Figure 7.1.4.2, is known as the American Flag or the Flag of the United States.

Figure 12.1.4.2: The United States National Flag

Post civil war, there were few more incidents in which Fort Marion played a part in. Throughout the 1870s, it was used to confine Plains Indians within its prison [10]. Throughout the 1880s, it was used as a prison for the Apache Indians. It was also utilized as a military prison during the Spanish-American War in 1898 for war deserters.

Fort Marion was officially removed from the Army's active duty roles in 1900. It was named a National Monument and soon transferred into the ownership of the National Park Service, who still retains control over the structure. The NPS renamed the fort to its original name, Castillo de San Marcos, and has done everything possible to maintain the structure over the years, as well as inform the public of its importance in Florida history.

12.2. Virtual Reality

12.2.1. What is XR/Virtual Reality?

Broadly speaking, "XR" is a term used to describe all forms of visual extension of reality, such as augmented reality or virtual reality. [11] Virtual Reality (VR) is, as the name suggests, a simulated environment that the user is placed into. The goal of VR is to be an immersive experience for the user. Instead of viewing the environment on a screen, the user will don a head-mounted display (HMD) which will display the artificial environment whilst covering the entire peripheral vision of the user. [12] Then the player will interact with the virtual world through hand controllers. Currently, most home set-ups will have the user stand still or sit down and control their character through joysticks on a controller. However, there are also setups that track user movement in a predefined room allowing for deeper immersion. There are even setups with omni-directional treadmills to give players even more immersion with movement within VR.

At this time, VR is primarily used for entertainment, most notably for video games. However, it also has seen use in education and business. For example, VR is used for military and medical training to simulate high-pressure scenarios in a more immersive environment to give better training. Our project is also intended for educational use, albeit in a much less stressful simulation.

12.2.2. Why VR?

VR provides a highly immersive and novel experience for a user. This serves to heighten engagement and interest with the simulated experience from the user. While video games are an interactive and engaging experience, VR takes it a step further. VR interaction has a much wider variety and freedom of movement and interaction for the player than a standard keyboard and mouse or game controller.

The main appeal of going to a national park or historic monument is to immerse yourself in the surrounding environment and take in the sights. Visiting a simulated park viewed on a simple computer screen is much less impressive than walking around the park and experiencing it first hand. Since VR is more immersive than a screen, it will hopefully bridge that gap and provide an experience much closer to actually visiting the park.

It should be noted that while a VR experience is preferable, not everyone has the capability to gain access or use a VR headset. Many people cannot afford a VR headset. VR has also been known to give users motion sickness. Some may not have the technical hardware to use a VR headset. Since this is a project with the end goal of being educational, accessibility and ease of use is key. Therefore, we have opted to add a normal PC version as well.

12.3. Unity

12.3.1. What is Unity?

A game engine is a software development environment geared towards developing video games. [13] The game engine provides functionality to a game developer to build off of, so they don't have to start from scratch. Some of these functions include graphics rendering, physics engine, animation, artificial intelligence, sound, etc. Essentially, a game engine provides a framework and the accompanying documentation for a team of developers to produce a video game. Unity is one such game engine.

12.3.2. Why Unity?

Unity is a widely popular, cross-platform game engine developed by Unity Technologies. Cross-platform is key here, as that means that the developer can code within the Unity editor without the need to switch between platforms. Unity is a widely popular game engine that is free to use if the developer generates less than $200,000 annually. This makes it very appealing for developers who are starting out or have a small budget to get their project off the ground.

Unity boasts an intuitive interface in its integrated development environment (IDE) that streamlines the development process. [14] This IDE lets you visually build the scenes in the game and connect all the parts as well as test the game build within

the editor. Furthermore, Unity has an asset store made available to all developers so they can find and purchase premade assets for use in their own projects.

12.3.2.1. Unity VR Development

Unity has a built-in plugin called “Open XR” that attempts to standardize interactions between unity and virtual reality headsets to simplify the development process for VR projects. An example is shown in Figure 12.3.2.1.1. Open XR standardizes VR headset input to Unity, the developer must still select which VR headsets are expected for the project under “Interaction Profiles” in the XR Plug-in Management setting. The major benefit of Open XR is that the developer is able to create generic XR objects and Unity will handle the individual headsets inputs to interact with those objects. Under the Open XR framework, the developer is able to make a single scene for all VR headsets instead of having to make individual controller objects and a script that will detect which headset and activate the correct camera and controller objects.

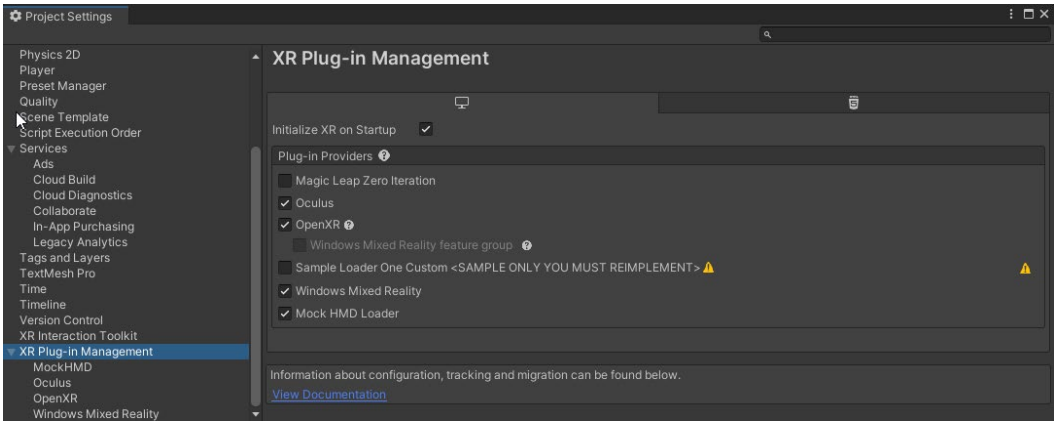


Figure 12.3.2.1.1: Example XR Plug-in Management Setting

While Open XR greatly simplifies development for VR in Unity, there are several issues we must investigate:

- Testing VR component with and without a headset
- The Open XR-plugin simply ensures that the headsets are compatible and that Unity can read the inputs of the headsets. It does not provide any framework for using the inputs.
- VR objects require specific VR interaction which does not translate to a mouse and keyboard input scheme.

12.3.2.2. XR Interaction Toolkit

There is a free Unity package in the asset store called “XR Interaction Toolkit” that is commonly used by VR developers. This package provides a framework that has default scripts for VR headset inputs and interactions for developers to use and edit. This toolkit can only be accessed if the “Enable Preview Packages” setting is enabled under Project Settings > Package Manager > Advanced Settings, shown in Figure 12.3.2.2.1. A preview package is a package that is still in early development and testing and is not ready to be shipped. Many developers use this toolkit despite this, but it is noteworthy as there may be unforeseen bugs due to the nature of the package.

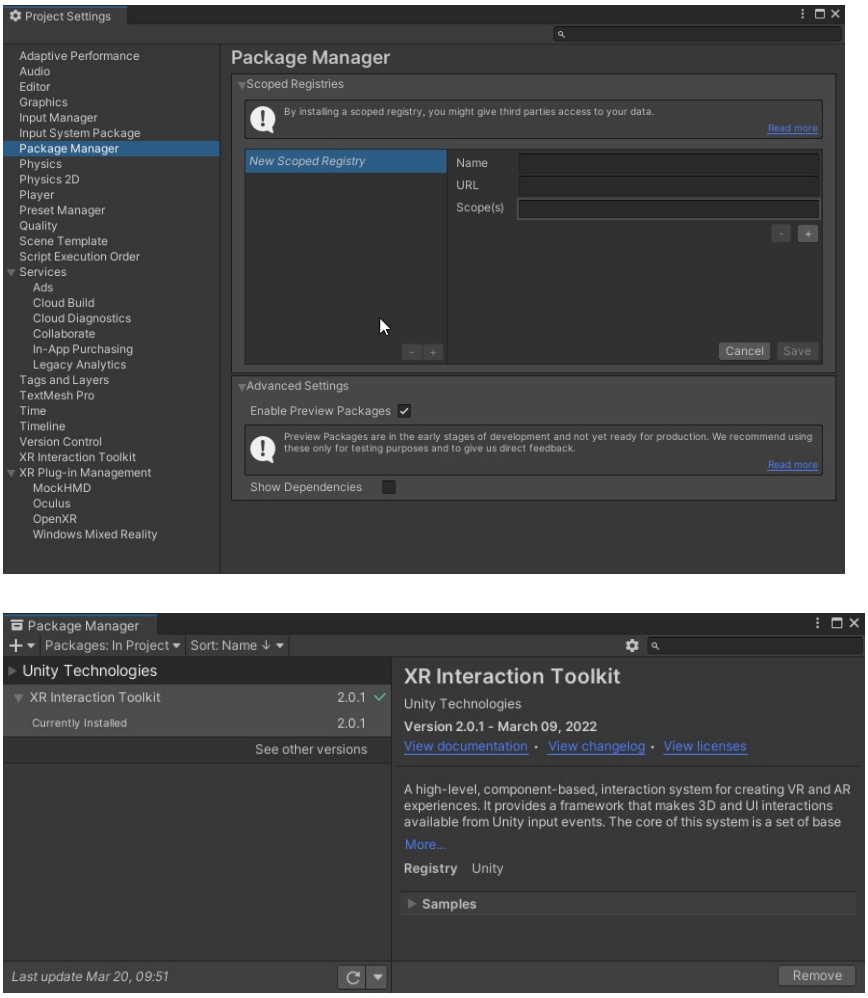


Figure 12.3.2.2.1: Example Installation of XR Interaction Toolkit

The XR Interaction Toolkit provides several presets for VR development.

Specifically it provides default input actions that map to the Open XR input system in Unity. Since VR provides a very different control scheme for locomotion within the game, we need to handle the inputs from the VR headset differently than we would a standard keyboard and mouse. The presets given to use by the XR Interaction Toolkit provides the scripts and mapping for these inputs to be attached to an XR Rig game object in Unity. These presets are broken out in Table 12.3.2.2.2.

Continuous Move	Smoothly moves the in-game camera over time
Continuous Turn	Smoothly rotates the in-game camera over time
Left Controller	Defines input actions for the left controller
Right Controller	Defines input actions for the right controller
Snap Turn	Rotates the in-game camera in fixed degrees

Table 12.3.2.2.2: XR Interaction Toolkit Presets

Continuous Move and Continuous Turn are used to simulate walking the user turning their head, respectively. Left Controller and Right Controller simulate the user's hands and the available actions for each hand. Snap Turn is an option for users who get motion sick from the VR headset. Snap Turn instead allows the user to change their view with a button press on the controller that will rotate the camera at a set angle each press.

The XR Interaction Toolkit also provides an XR Device Simulator, which allows the developer to test their VR scene without donning a headset.

12.3.2.3. Testing in VR

While the Unity Open XR plugin provides a standardized format for object creation in development, testing with each individual headset is still required to ensure parameters and individual bugs are handled. This places a large burden on the developer to have all the headsets on hand to test. Additionally, they would have to don the headsets to see and interact with new objects and scripts. The solution for initial testing is to test the VR components in the Unity editor with mouse and keyboard and later do quality assurance checks with the individual headsets at major breakpoints in development.

At the time of this project, Unity has an option under the XR Plugin Management setting called “Mock HMD Loader” (refer to Figure 6.1) that will give a visual representation of what the VR headset will see in the Unity editor. However, the developer cannot interact and the Mock HMD Loader will not take inputs by itself. With the XR Device Simulator, the developer can use the Mock HMD Loader with the inputs listed above to simulate a VR headset.

12.3.2.4. VR Controls

Movement in VR is handled by the joysticks on the controllers. Typically, one hand’s joystick will control player movement, while the other hand will control the camera or the player’s view. There are several buttons on both controllers that allow the user to interact with the game world, open menus, select menus, and other actions typical of a video game controller.

The left joystick will control player movement and the right joystick will control player camera, however it should be simple to allow the user to switch this control scheme to their preference.

12.3.2.5. Mouse and Keyboard Controls

The mouse and keyboard controls we will be implementing in our project will use the industry standard WASD movement and mouse movement to control the camera.

- W moves forward
- A moves backward
- S moves to the left
- D moves to the right

Due to how VR objects are created and scripted within Unity, it is not possible to port those interactions to a simple 3D mode with mouse and keyboard or controller input. This requires us to create all objects with a “normal” interactable script concurrently with the VR interactable script. Similarly, the player controls and player camera must also be separately scripted depending on VR or non-VR input.

The player movement and interaction scripts are not the same scripts that will be used to simulate VR. It would be undesirable to simulate VR for the user experience as the rendering for VR is different from the rendering for a standard 3D game. Furthermore, interacting with objects in VR will differ in input and effect. A 3D interaction may not have the same complexity in motion and player input compared to the VR version. For example, opening a door in VR would have the player actually push the door with input and motion from the VR wand controls. Conversely, attempting to control a door with a button press and WASD movement or a game controller button with joystick movement in 3D would not be intuitive or immersive.

Unity’s XR camera object is specific to VR. As stated before, this camera object will render the image for VR headsets, which looks odd when displayed on a normal monitor. While simulating VR controls without a headset is possible, it is not desirable for user experience. Another camera object must be made for normal input from a mouse and keyboard or game controller. However, only one camera can be active as the “main camera” as that is the camera that will render the scene for the player. This necessitates a script to detect which device the player has, if any, and activate/deactivate all necessary game objects and scripts based on that

device to ensure a smooth game experience. Furthermore, this will also require multiple player movement scripts, camera objects, and interactable scripts.

12.4. Computer Graphics

Computer graphics is the process of creating computer generated images. It has a variety of uses and meanings depending on the context. Typically it refers to the representation and manipulation of image data, technologies used to create and manipulate images, or methods for digitally synthesizing and manipulating visual content. [15] In the context of this project, it primarily refers to the modeling and rendering of computer generated images as well as animations relating to those images. It can be broadly broken up into two categories: 3D and 2D, with the primary distinction being the coordinate system used for either. For 2D computer graphics, there are only the X and Y planes, creating a “flat” model of which to render an image from. In 3D modeling, the model contains the mathematical information to simulate a 3D space giving us the X, Y, and Z planes, which simulates depth. This allows the 3D model to be viewed from different angles.

3D modeling is the mathematical representation of the surface of any object in a simulated 3D space. [15a] The model is created by manipulating the edges, vertices, and polygons of the object to create the desired shape. 3D models can be divided into two categories: solid and shell, however for our project we will focus only on shell. Shell 3D models represent only the surface of the object and not the volume, hence the name. Almost all 3D models in games and films will be of this type.

12.4.1. Performance

The 3D models must then be rendered to display on a computer screen. There is the option to do 2D rendering or 3D rendering. 2D rendering produces a static image that can only be viewed from one viewpoint. 3D rendering produces a simulated 3D environment that allows multiple perspectives and angles while replicating real life visual effects, such as lighting and shadows. [16]

2D rendering typically was used during the early days of computer graphics and was primarily intended for creating graphics that could be printable for use in the

real world. Today, 2D rendering is primarily used for vector graphics and raster artwork. [17] Vector graphics is the set of mechanisms for creating images directly from geometric shapes on a Cartesian plane. Raster artwork is the representation of a 2D image on a rectangular grid of pixels.

3D rendering is mostly used in digital mediums, since the output must be interpreted and displayed through software. As stated above, 3D rendering simulates the model in 3D space allowing for changing camera angles and perspectives.

Obviously, from a performance standpoint, 3D rendering requires more computational power due to the additional dimension and mathematical computations required for it.

12.4.1.1. Graphical Importance

Since this project is a VR game, it necessitates 3D modeling and a 3D rendering of the game world. Given our performance constraints, it may serve to render the surrounding scenery of the fort that is not reachable as a 2D rendering. Hopefully this will reduce our graphical requirements as it should reduce the number of polygons the game must render at any point in time. However this would also reduce immersion and could be difficult given the tools at our disposal. For example, if the backdrop is rendered in 2D we would lose the ability for it to be affected by lighting and produce shadows which could arguably affect immersion. Obviously, this may not be worth the extra computational cost.

Our background options are:

1. **Create a polygon, such as a square or hexagon, that encases the fort some distance away.** Each face of the polygon will be a 2D sprite that contains the backdrop image of our choosing, shown in Figure 12.4.1.

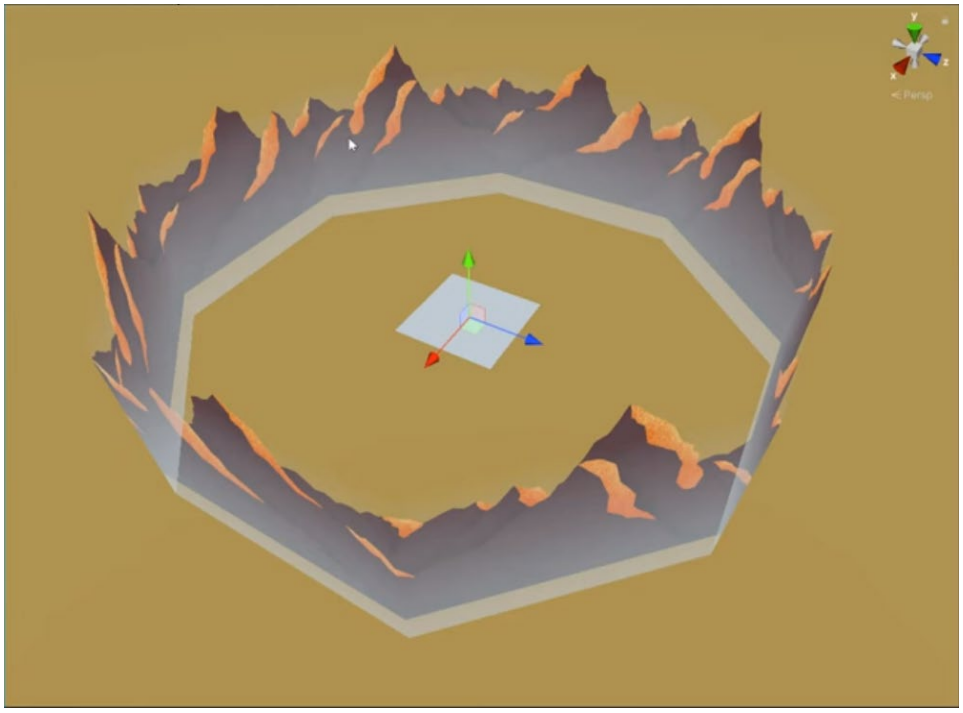


Figure 12.4.1: Example of sprite backdrop. [18]

To make the sprites, we have several options:

- Model the surroundings in 3D and make the fort transparent. Then from the center of the fort, get 2D renders of the surrounding landscape at set angles, for example, 0°, 90°, 180°, and 270° for a square shape. Then apply the corresponding images to the correct face of the polygon. The challenge would be finding the correct distances and perspectives which may take multiple trials, as well as the time requirement to additionally model the surrounding buildings and scenery.

- Take photos of the surroundings and edit the photos to be usable as sprites, with the above method. This is likely unfeasible due to the popularity of the fort and getting uniform angles and perspectives.

- Create and apply a generic scenic backdrop with an applicable image to each face. The main drawback to this is that we would lose some realism and immersion.

Of our options, the most simple and feasible would be the 3rd, applying a generic backdrop. One could argue that the background simply needs to be there to help the player feel that the world is not simply an island with nothing surrounding it, but does not need to go further than that.

The next most feasible would be the first, in which we model the surroundings as well and then take renders at specific angles to get correct and realistic scenery whilst also maintaining low polygon count. Unfortunately this comes at the cost of man hours spent on additional modeling.

The middle option is likely unfeasible due to the constraints of requiring camera work and taking photos of the surroundings and piecing them together which would also require a lot of manpower and man hours. Furthermore, the background would be photorealistic while the actual fort may not be. This would produce a jarring effect on the player.

2. **Adding the surrounding scenery to the skybox**. A skybox encloses a scene in a cube to make the scene seem larger than it is. The concept is similar, however a skybox requires a texture map as it is a 3D object. This adds a complication as we would have to generate a texture map of the surrounding buildings and the sky. This still presents the problem of how to produce the image of the surrounding scenery. The options for producing the image for the texture map is the same, with the added problem of converting the images into a skybox texture map that looks correct. Unity has the option of making the skybox a cube or other polygon, or it can be a panoramic image.

For the cube, we would make a texture map specific to the cube. For the panoramic image, we would need to create a 360 image to be used as the texture for the panoramic skybox. Blender does afford us this option and the approach would be the same as mentioned above. We would make the fort transparent and do a panoramic render of the surroundings model.

While this is also feasible it could be more difficult to create a texture map for the skybox, rather than use a generic one.

3. **Simply model it in 3D.** With some of the solutions shown above, this would have already been done anyway. This solution is to simply port that 3D model over to Unity and use it as the background. The obvious advantage over some of the solutions above is the simplicity as well as it being the first step to a few of them. Furthermore, it would provide a more realistic background that would also be more obviously affected by day/night cycles and weather effects. The clear disadvantage is, again, the requirement for more 3D modeling and the man hours associated with that task.

12.4.2. Textures

A texture map is an image that is applied to the surface of a shape or polygon in computer graphics. [19] The goal of a texture map is to represent the surface and texture of the object it is being applied to, for example a rough rocky texture for a rock object. Every vertex in a polygon is assigned a texture coordinate known as a UV coordinate, which is defined as the coordinates in texture space. This is necessary as the maps are still 2D and it needs to be mapped to the faces of the polygon. The coordinates tell the program how to wrap the map around the object correctly.

Without textures, 3D models lack depth and variety. Textures provide visual clarity, contrast, and color information to make the game world interesting and distinctive. This is especially important for our project as we would like to capture the unique characteristics of the fort, such as the Coquina walls and their distinctive texture and look.

12.4.2.1. Creating a texture from a photo [20]

The following steps were done in Adobe Photoshop, but any image editing software will suffice.

1. **Equalize the image.** Photos taken with a camera will typically have a lighter side and a darker side. It is important to equalize the image so that this is not represented in the final texture.
    a. Duplicate the image twice
    b. Select the first duplicate and select Filter > Blur > Average
    c. Select the second duplicate and select Filter > Other > 200 pixel High Pass and then change the blending mode to Linear Light
    d. Change the Opacity and Contrast of the second duplicate to better match the original image
    e. Merge the duplicate layers to create your equalized version of the image
2. **Crop the image into a square.**
    a. If the edge of the photograph is blurry, scale the crop towards the center to avoid the blurred edges.
        i. Image > Adjustments > Image Size
3. **(Optional) Enlarge the texture swatch**
        i. Filter > Other > Offset by half the crop size
        ii. Patch the lines in the middle to blend it and make the pattern uniform

12.4.2.2. Mapping Textures to a Model

Blender provides a texture mapping tool in the UV Editing tab. To map a specific image texture the user must go to the material properties tab and change the Base Color field to “Image Texture”. Then the user can open the specific texture map to map in the UV Editing tab.

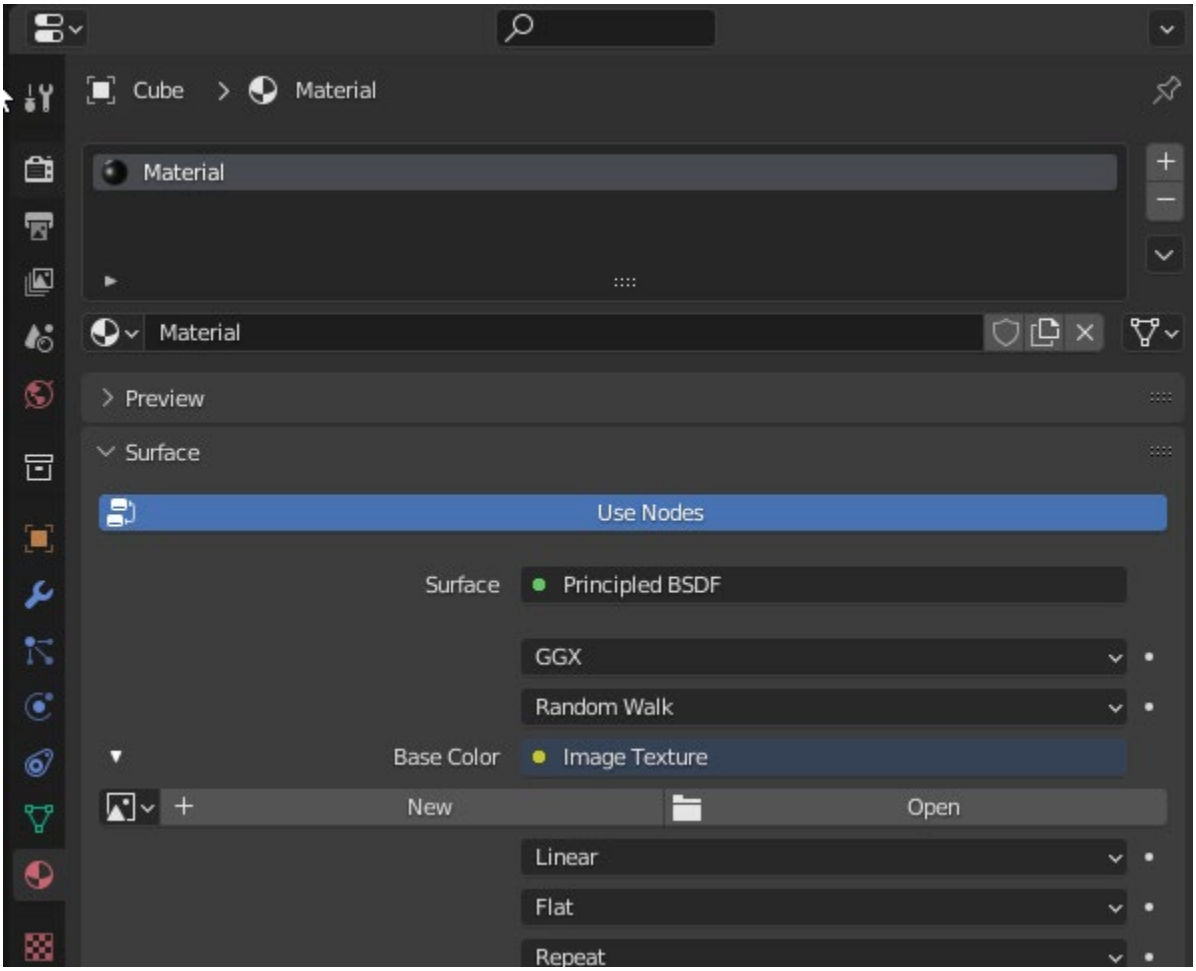


Figure 12.4.2.3.1: Example Material Properties Tab in Blender

In the UV Editing tab, the image is displayed along with the UV mapping of the model being textured.

The faces can then be manually adjusted by the user so that the final model with the texture looks good.

12.5. Optimization

Given that one of the goals of this project is accessibility, optimization is very important in order to provide a quality product to a variety of people with differing hardware capabilities. Furthermore, from a technical standpoint, we also need to meet the minimums for each VR headset in order to have a functioning game.

There are many techniques to optimize a game both outside and within the Unity game engine. One of the first steps towards optimization is ensuring the models are created with optimization in mind. Some of the ways to do that include: minimizing polygon count, using as few materials as possible per model, and using as few bones as possible.

It is also necessary to have Level of Detail (LOD) variants of all our models. LOD renderings reduce the number of triangles rendered as the distance from the camera increases, saving on processing power. [21] The drawback for LOD is that the variants must be made and loaded into Unity so there is a larger memory requirement.

12.5.1. Making LODs

1. Go to Object Mode and select the object that you will be making LODs for
2. Make as many copies of that object as you plan to have LODs
3. For each copy:
    a. Select the modifier Decimate and check the box Triangulate. This is necessary as Unity triangulates everything that is imported so it's best to get an idea of what it looks like in Blender.
    b. Adjust the ratio slider until you are satisfied

4. Select the objects and go to File and select “export as FBX”, make sure to give them identifying names to denote where they belong in the LOD hierarchy.

5. Import textures to Unity and make them materials

6. Place FBXs into the project

7. Apply materials to FBXs in the Materials tab

8. Select most detailed object and add a LOD Group component to it

9. Drag the FBX objects to their hierarchy spots

12.5.2. Optimization in Unity

Unity has a stats window that provides some basic information about the performance of the game when you enter Play mode. Figure 7.5.2.1 is an example of that.

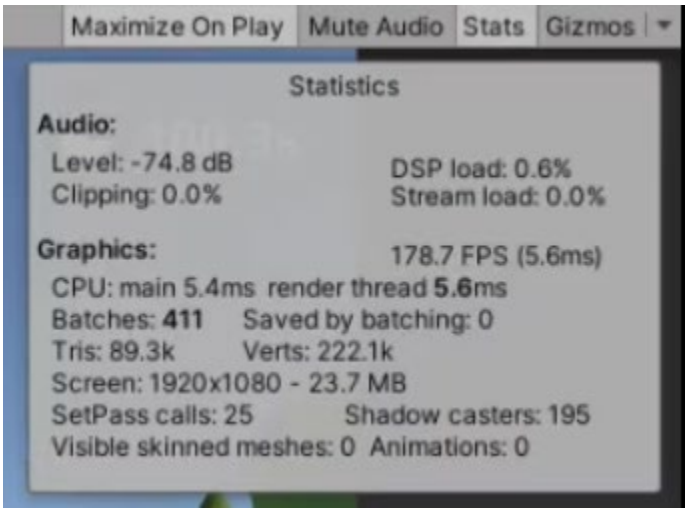


Figure 12.5.2.1: Example Stats Window in Unity

Unity also provides a Profiler tool which gets performance information about the game. It provides much more detailed information than the stats window so the developer can monitor the game performance and identify problem areas to optimize. This menu is accessed through Window > Analysis > Profiler.

12.5.3. Baked Lights

Baked Lights is a lighting mode within Unity that performs the calculations for lighting of all static objects and saves it to the disk as a lightmaps and Light Probes. [23] An example of setting an object to static is shown in Figure 7.5.3.1. A lightmap is essentially a texture map that specifically handles the lighting of an object whereas a Light Probe stores information about how light passes through empty space in the scene. When the scene is loaded Unity will light the scene with this lighting data reducing both the shading cost and rendering cost of shadows at runtime.

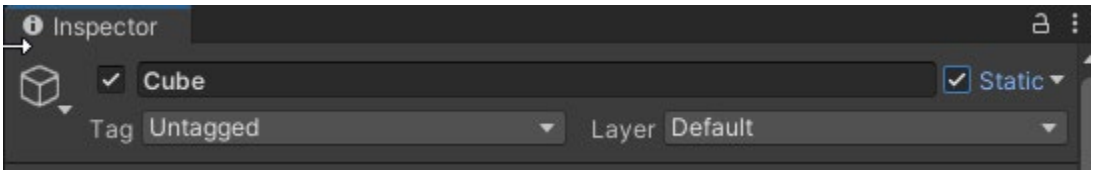


Figure 12.5.3.1: Example of Setting an Object to Static

The Lightmap itself can be further optimized by reducing the lightmap size and resolution as well as compressing the lightmap. Furthermore, you can change the Lightmapper to Progressive GPU which will offload some of the CPU cost to the GPU providing better runtime performance. These options are available in the Lighting tab. [24]

If Progressive GPU is used, we need to generate Lightmap UVs for all of our models. We can simply access our models and in the inspector, enable that setting and hit apply, as shown in Figure 7.5.3.1.

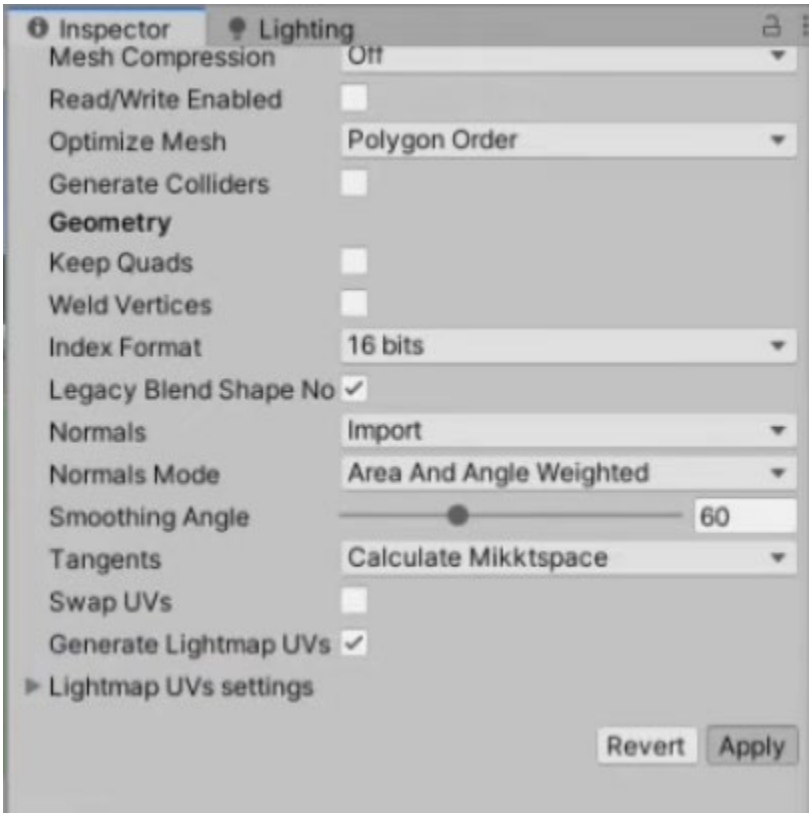


Figure 12.5.3.2: Example Generate Lightmap UVs

It is important to note that the shadows will be static and if the object is moved the shadow will not move with it. This is why it is best used for objects that will not change during runtime. Furthermore, Baked Lights do not contribute to specular lighting (reflective lighting) and dynamic objects do not receive light or shadow from Baked Lights. However, you can change the lighting mode to Subtractive and it will only bake static objects and dynamic objects will still have dynamic lighting.

12.5.4. Shadows

Shadows are famously a large contributor to performance drops in games. Unity has three shadow types: Soft Shadows, Hard Shadows, No Shadows. [25] With Soft Shadows being the most intensive and No Shadows being the most resource efficient. Unfortunately, in a 3D space shadows are key to giving the illusion of depth, although there should be an option to turn off shadows if performance is very low. Soft Shadows are the most realistic shadows but are much more intensive. The happy medium is dark shadows and that is what this project will use. The Shadow Type can be accessed in the lighting object through the inspector.

12.5.5. Anisotropic Filtering

Anisotropic filtering is a texture filtering that increases the draw distance of textures. [26] It has the GPU make multiple passes over surfaces to increase the texture detail which can be a complex process. Obviously, this is resource intensive and we are not making a AAA title so we do not need this functionality. We can turn this off in project settings.

12.5.6. Textures and Materials

Similar to LOD, a mip is a version of a texture at a specific resolution. [27] Mips exist in a set called mipmaps and when mipmaps are activated it will lower the resolution of the texture as the location of that texture gets further away from the camera. Unity will automatically generate mips for a texture map if the option is selected for each texture, shown in Figure 7.5.6.1.

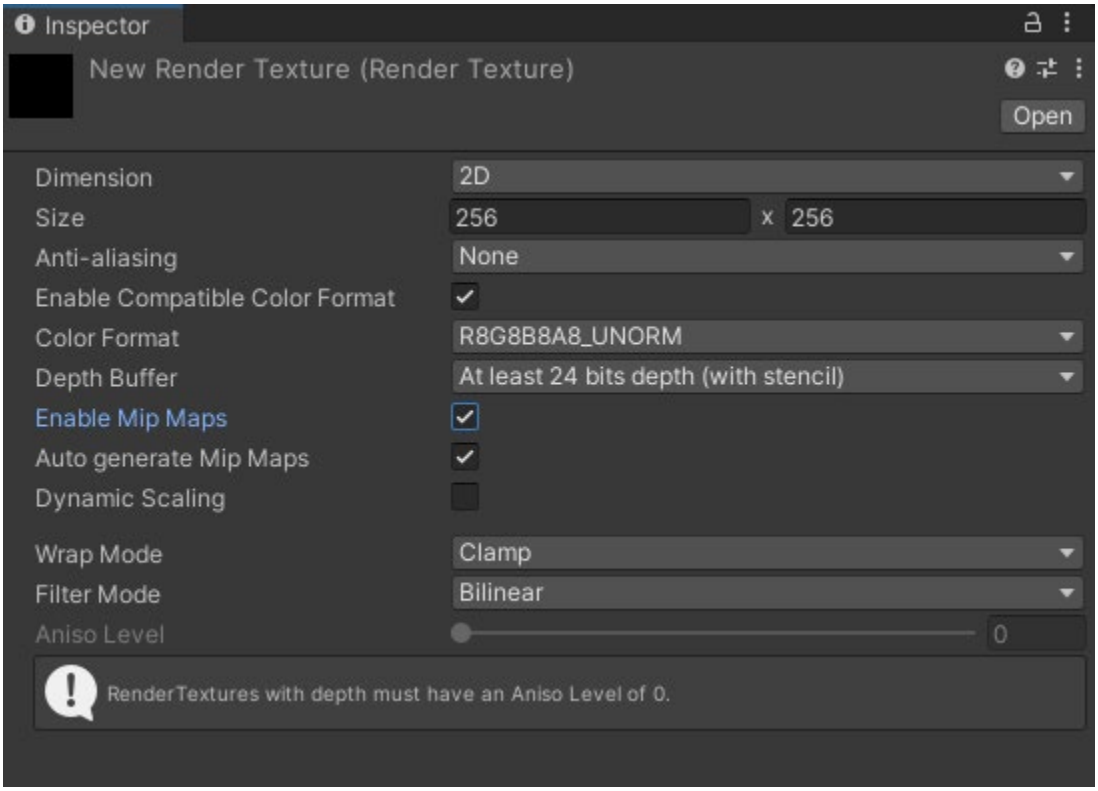


Figure 12.5.6.1: Example of Enabling Mip Maps

You can also lower the map size and use crunch compress to further optimize the texture as shown in Figure 7.5.6.2.

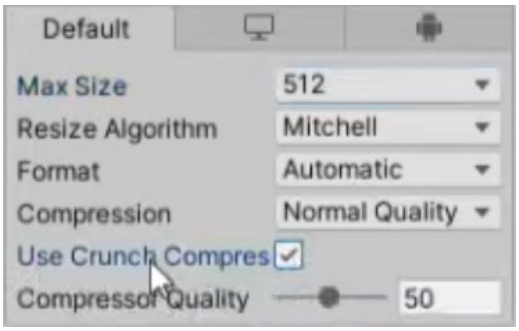


Figure 12.5.6.2: Example of Crunch Compress

For each material, you can reduce smoothness and remove specular highlights as well as reflections. [24b] Smoothness obviously requires the game to apply processing power to smooth out the edges of the material. Specular highlights handle the reflective lighting of the material. While this adds some nice detail, it does come at the cost of performance and is not necessary. You can further optimize the materials by turning on the Enable GPU Instancing setting which reduces drawcalls by rendering similar meshes in a single GPU instance.

12.5.7. Other Optimizations

UI

- Disable Picture Perfect option in the Canvas.
- Turn off Raycast Target for UI elements that don't require player interaction.
    - Raycast targets handle line of sight calculations. For a UI element, once you click it will cast a ray forward and hit and return the first raycast target it hits. If you don't need a UI element to be clicked it does not need to be a raycast target and turning this off will save on computation.

Sprites

- Turn on generate mipmaps options

Main Camera

- Lower Far Clipping Planes
    - Reduces the maximum draw distance
- Ensure Occlusion Culling is on
    - This is normally on by default. Occlusion Culling is when the game does not render any object that is not currently in view of the camera.

12.5.8. VR Optimization

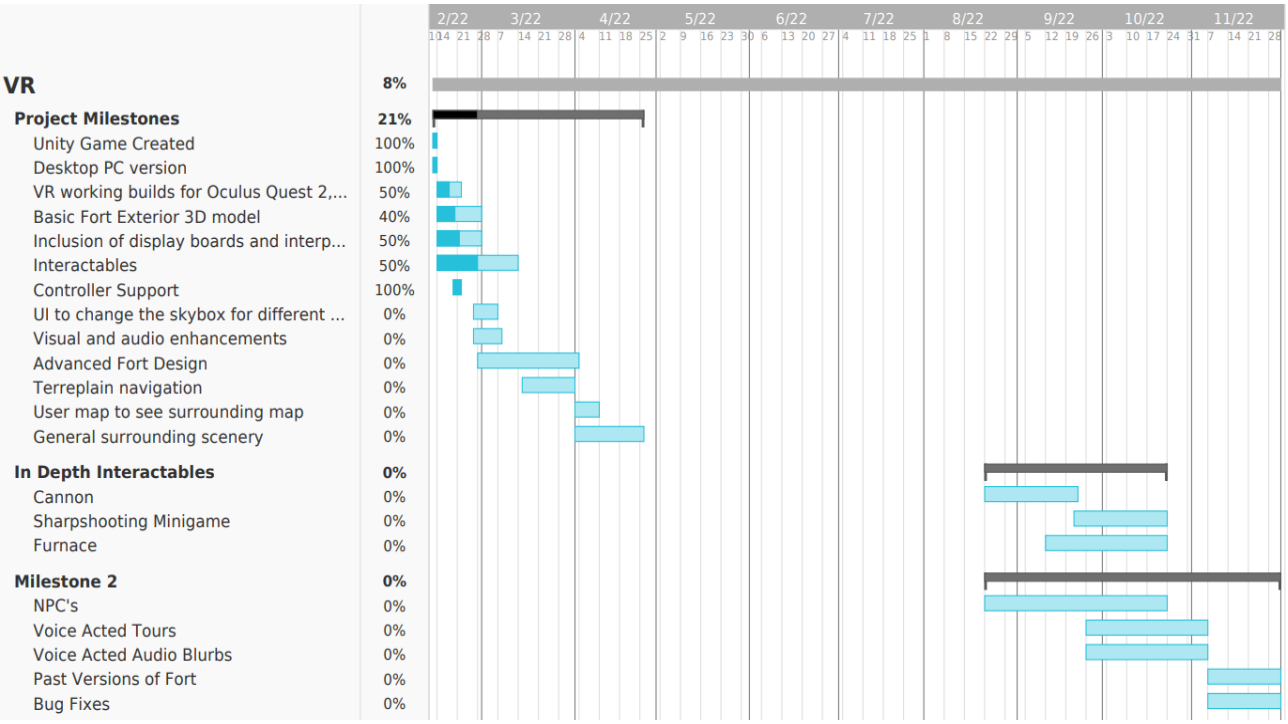
Besides the optimizations mentioned above, VR has its own set of optimization options:

- Rendering Pass Type [28]
    - Multi-pass Rendering: Since VR headsets have a screen for each eye, sometimes Unity will need to render a scene twice. Multi-pass rendering attempts to address this and avoid duplicating things that don’t need to be duplicated. Unfortunately, it will still render most objects twice. This is the most compatible with devices, though.
    - Single-pass Rendering: combines the two textures for each eye into one big texture, called a double-wide texture. This is much more intensive on the GPU but much less intensive on the CPU.
    - Single-pass Instancing: Works like Single-pass rendering however it also reduces the GPU cost as well. It performs a single render pass and replaces each draw call with an instanced draw call. This is the option for best performance.
- Texture Compression
    - ASTC texture compression is a compression algorithm used for Android, but textures can override to Android and use this compression algorithm. [29]
- Texture Mapping
    - Normal texture mapping does not work as well in VR because of the constantly changing viewing angle as well as rendering it for both eyes which does not translate as well. Instead, for VR objects it is better to use a height map. The main difference is that a height map factors in the view angle of the player. This can be computationally expensive and should be used only on important objects in a VR setting. [30]

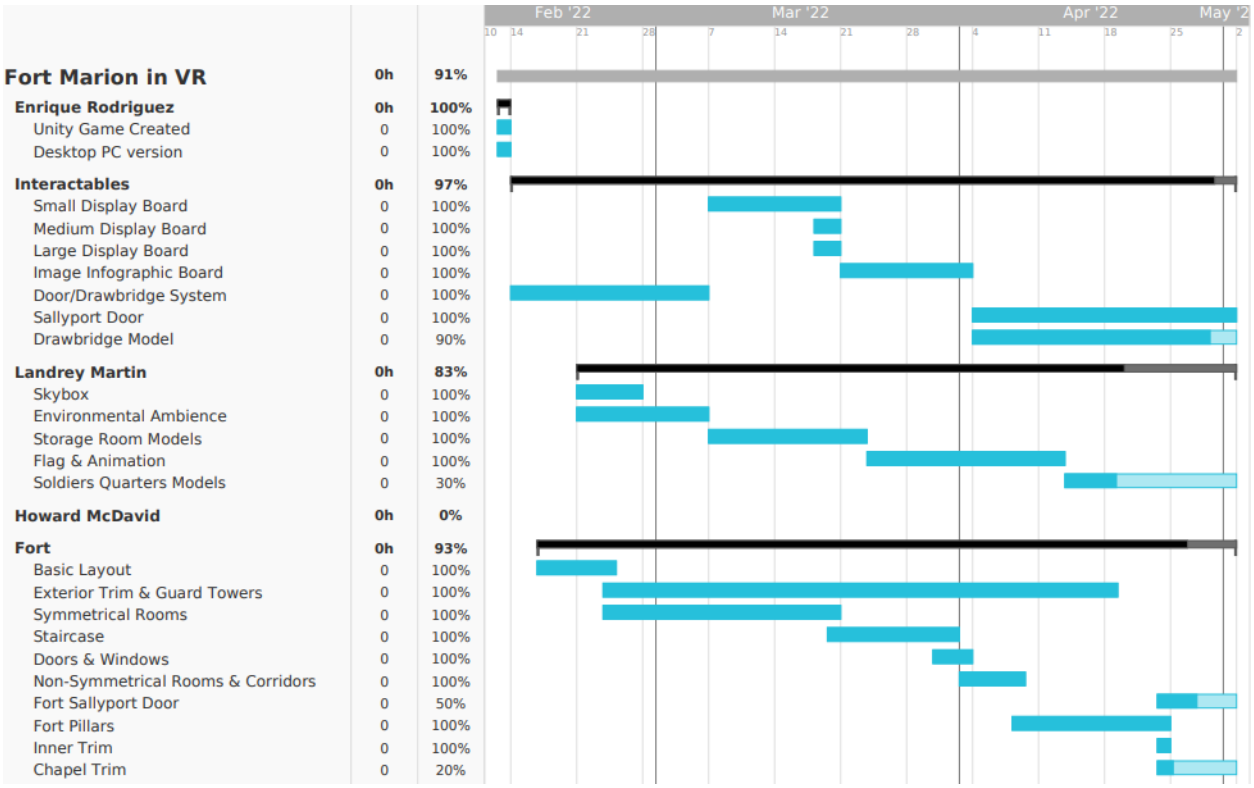
13. Project Milestones

Before beginning our project, we created a Gantt Chart to help track our progress on this project. As we began assigning more work and specifying the workloads, we quickly outscaled our original goals we had set and had to recreate our Gantt Chart. Below is our original Gantt Chart which covered both semesters and our

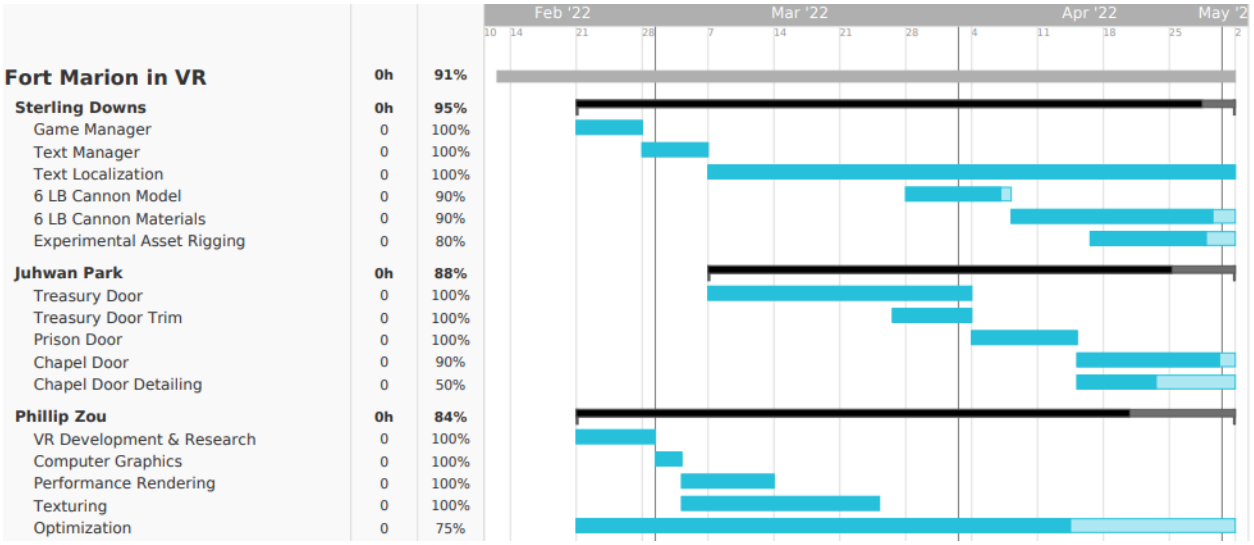
updated Gantt Chart which only covers Spring 2022. An updated Project Milestones page will be included next semester that showcases Fall 2022 progress.



Original Gantt Chart



Updated Gantt Chart Page 1

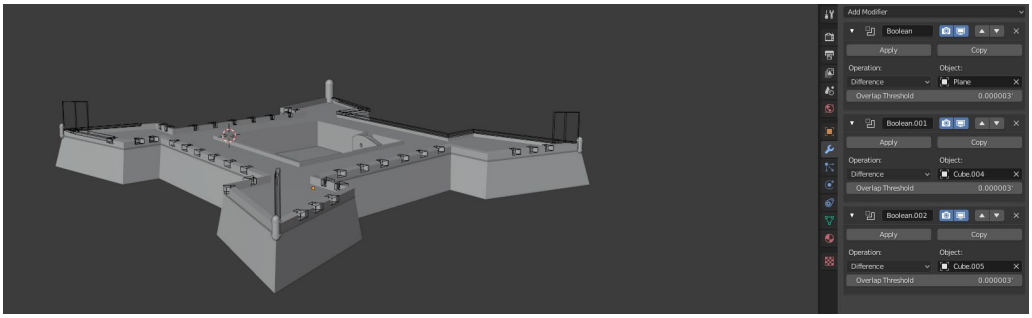


Updated Gantt Chart Page 2

14. Terminology and Definitions

- Assembly - A collection of parts and sub-assemblies that are created in the same system.

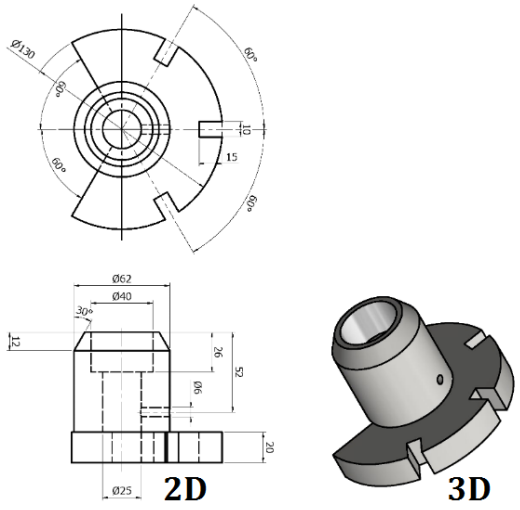
- Boolean Logic - The use of 3D objects to create a 3D boolean matrix which is used to determine new vertices on another 3D object. Depending on the boolean function used, the matrix is used to either remove, add, or keep the mesh in the places where the boolean matrix is true and, when false, keep, keep, or remove respectively.



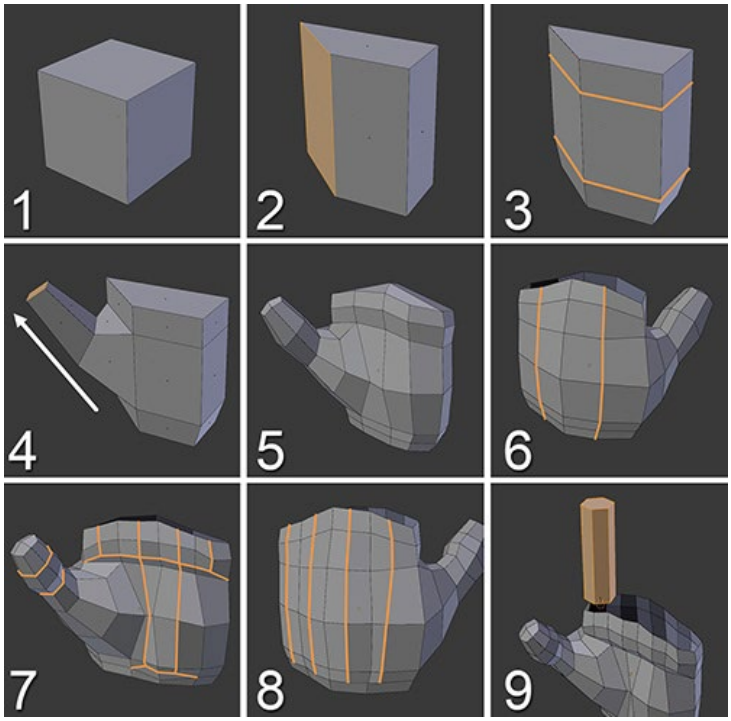
Screenshot from Blender showcasing the boolean tool's use on Fort Marion

- Creation Suite - A collection of multimedia and editing tools.

- Dimension Driven Design - Modeling design based on solid-modeling capabilities involving parametrics. A sketch is drawn using dimensions and 3D models are created by extruding shapes from that sketch.



- Geometry Driven Design - Modeling design based on curves, surfaces, and volumes. 3D models are made with basic shapes combined with boolean functions and shape modifiers.



15. Technologies

15.1. Source Control

The two most commonly chosen source controls for Unity projects are Unity Collaborate and Git. Although Unity Collaborate is a built-in source control for Unity, we decided to use Git through GitHub for several reasons. Each source control has various reasons to use one or the other, as listed in their pros and cons below:

1. **Unity Collaborate** is a simple to understand, built-in source control through Unity itself. It requires very little set-up from the user and is easy to understand. It's best used for projects that consist of very small groups and generally small file sizes.

    Pros:

    - Seamless integration through Unity itself
    - Easier to use for beginners, very straightforward
    - Free for Students

    Cons:

    - Limited in group size
    - No branching
    - Reverting to previous versions is unreliable, errors will persist through rollbacks and lock several Unity features
    - Very slow for updating and opening projects

2. **Git** is a more developer focused external source control through GitHub. Git is generally a more industry focused source control and offers more connectivity between other web services like Trello and Jira. Git also allows for easier presentation of your project through integrated documentation and web hosting through websites like GitHub.

Pros:

- Industry standard
- Supports multiple branches
- Easy to track changes in code across different versions
- Numerous bonus features with extensive documentation

Cons

- Requires an external application for pushing and pulling
- Additional setup for maintaining large files (LFS)
- Harder to arrange a standard workflow

15.2. Asset Creation

Assets are any object that is used within a project. For this project, we intend to create all the assets ourselves whenever possible and obtain the remaining assets from the Unity Asset Store, either free or paid assets using our $300 budget. The programs used to create assets most commonly are Blender and various Autodesk products. All of these programs have their respective place in the industry and each software has their pros and cons below:

1. **Blender** is a free, open source 3D creation suite. Blender is geometry driven which means 3D shapes are created and modified into the desired shapes. It is cross platform so it works on Linux, Mac, and Windows. The main benefit to blender is that it contains significantly more features tied into one program as it contains modeling, as well as, rigging, simulation, rendering, compositing, and much more. It’s available for download from their website and on Steam. As it’s open source, there are no problems using any of your created assets for use on personal and public projects.

Pros:

- Free with no caveats, no school account or work account needed
- Endless tutorials online, very large community that updates these tutorials as Blender gets updated
- Several softwares combined into one, no need to download several softwares and transfer files between them.
- Free addons, Blender supports many community-made addons that allow for quality of life improvements to entirely new features.
- Compatible with the largest range of file extensions, finding a compatible file type for other programs is fairly easy

Cons:

- Not beginner friendly without help, UI/UX is very confusing on launch compared to other modeling programs as its focus isn't 100% modeling.
- Modeling doesn't feel as fluid compared to other softwares, easily changing sketches and already existing shapes is complicated due to their scale system.
- Several menus open when certain actions are inputted and close with subsequent clicks with no way to re-obtain those menus later without shortcuts or exploring through lists.
- No simplified version, most people won't use the other features Blender provides so it doesn't make sense to give all users all systems available.

2. **Autodesk** is a company that makes several software products for modeling, manufacturing, and construction. The most commonly used of which are Inventor, Maya, and 3DS Max. All 3 of these are used in the workplace and

contain similar features so understanding which one to use depends on the features you are looking for.

2.1. **Inventor** is a 3D creation suite focused on mechanical designs and visualization. Inventor is dimension driven which means 3D models are created based on diagrams you've created. Inventor is not cross platform as it only works on Windows. Inventor is solely focused on modeling with a fairly low-depth animator included as well. Inventor allows for a 30 day free trial, a monthly/yearly subscription, and work/student accounts for free that contain paid features. Models created on Inventor can not be sold for profit as its intention is internal use.

Pros:

- Inventor is considered as the most widely used computer-aided design software and as such, learning Inventor is a good skill to have in the industry.
- User interface is fairly intuitive and it is easy to modify already existing sketches to change objects.
- Dimension driven design is generally easier to understand for beginners compared to geometry driven design.

Cons:

- Not free, while it does have a trial, payment will be required for any long term project without the use of a student or work license.
- While seamless between other autodesk products, less file extensions are available for exportation which makes cross software compatibility an issue.
- Heavier workload on your PC compared to other similar softwares.

2.2. **Maya** is a 3D creation suite focused on animation and natural 3D design. Maya is a Non-Uniform Rational B-Splines (NURBS) modeling system which means models are based on primitive outlines of your final design, using circles, squares, and other shapes to trim away towards your final design. Maya is cross platform so it works on Linux, Mac, and Windows. Maya allows for a 30 day free trial, a monthly/yearly subscription, and work/student accounts for free that contain paid features. Models created on Maya can be used and sold for profit.

Pros:

- Maya is the industry standard for animation based softwares. NURBS modeling is also the standard for animation modeling and learning Maya is a very helpful skill to learn for animation based computer graphics.
- Modeling is able to be done using constraints and freehand, whereas most programs are solely done using constraints.
- Organic model creation is much easier on Maya than geometry and dimension driven design based softwares.

Cons:

- Learning curve to NURBS modeling is quite high and tutorials for Maya are not as updated compared to other softwares.
- Not free, while it does have a trial, payment will be required for any long term project without the use of a student or work license.
- While seamless between other autodesk products, less file extensions are available for exportation which makes cross software compatibility an issue.

2.3. **3DS Max** is a 3D creation suite more similar to Blender as it contains many features instead of focusing on one. Although focused in modeling, 3DS Max supports user made plugins and addons to add many features. 3DS Max is not cross platform as it only works on Windows. 3DS Max allows for a 30 day free trial, a monthly/yearly subscription, and work/student accounts for free that contain paid features. Models and scenes created on 3DS Max can be sold for profit.

Pros:

- 3DS Max contains many features combined into one software including modeling, simulations, and animation.
- Used by many companies as they can just pay one subscription price for one piece of software instead of paying for several pieces of software to do different things.
- Integration with several other apps including Unity, Unreal, Fusion 360, and other Autodesk products.

Cons:

- Not free, while it does have a trial, payment will be required for any long term project without the use of a student or work license.
- Slowly becoming outdated, other softwares are catching up or passing the features that 3DS Max provides and with few tutorials for it, its lifespan seems limited in comparison.
- Much heavier workload compared to previous products, a good computer is necessary to work with 3DS Max.

3. **Adobe Photoshop** is a raster graphics editor developed by Adobe Inc. primarily used by digital artists. The software has become the industry standard in digital art and has many features for editing and creating artwork. Photoshop currently operates on a subscription based model and offers a free trial, however UCF libraries have photoshop loaded on their computers for student use.

    Pros:

    - Photoshop is the industry standard for digital art, so most tutorials will focus on photoshop as their tool
    - Powerful tools within the software to edit and create art

    Cons:

    - Either have to pay for a subscription or go to campus to produce textures

15.3. Personal Experience

15.3.1. Sterling Downs

Throughout the course of this project, I will be using a variety of different software to accomplish my tasks. Due to this being a Unity project, the main environment of the actual VR implementation will be in Unity. As for 3D modeling, I will be using Blender. It is the 3D modeling software I am most familiar with and a great community for any support that may be needed. In addition to this, I will be using JetBrains Rider as my C# IDE. While Visual Studio offers many useful features, it can be clunky at times and I have found Rider to be a much better experience overall. I have been developing with C# for several years with experience developing libraries, console applications, and form applications via WPF and WinForms. Having worked on various .NET versions, including .NET Framework, .NET Core, and the .NET standard specification, I am very familiar with its feature set. I feel that I have a lot to offer in terms of building up the codebase of this project and plan to put various design patterns into place to keep the project well-structured and maintainable.

15.3.2. Landrey Martin

My experience is primarily in 3d modeling using AutoDesk's Inventor. During high school, my school offered a series of three classes that covered different AutoDesk programs: AutoCAD, Revit, and Inventor. AutoCAD allows for a lot of 3D modeling options; however, Inventor is more suited towards creating parts and assemblies. Revit was more so in the realm of architecture and building designs. I ended up taking the second and third classes in the same semester. The second class utilized Revit, while the third utilized Inventor. For the entire second half of both classes, we were given a very open-ended project. The Revit class's project was to design our dream house and come up with a video presentation consisting of renders from the project. The Inventor project was simply to model objects of our choosing and create a presentation showing off their parts that make it up, as well as the overall final model.

I chose to combine both projects and model objects for my house that I could not find within Revit prebuilt assets or websites that allow you to download Revit assets. I wanted my dream house to be a beach house with the garage being a dedicated surfboard shaping room, so I ended up surfboard shaping stands, leashes, a surfboard rack, a shortboard, and a longboard.

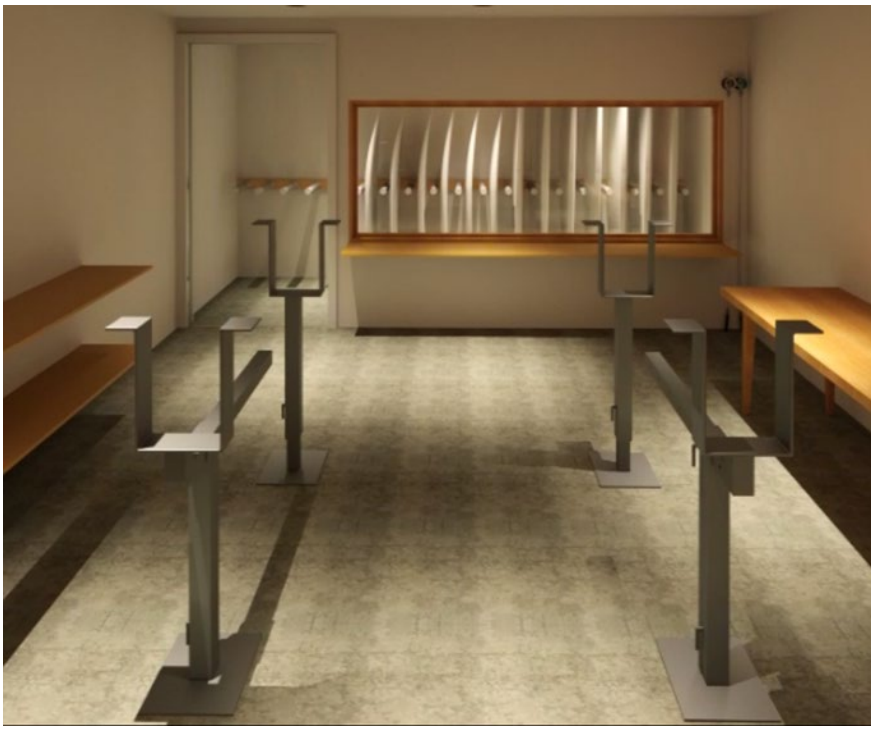


Figure 10.1.1.1: Surfboard shaping stands are in the forefront, while the other items I have modeled are in the back

Figure 10.1.1.2: Surfboards are displayed resting on the rack and the leashes are hanging on the wall on the right

As for my experience with modeling for this project, I aim to use AutoDesk Inventor for the most part. I have to model at least one object in Blender since I require some elements, such as wind, which is not an applicable tool within Inventor. I have had to watch some tutorials for Blender, but it was easy to get a hang of after watching and practicing with it for some time.

15.3.3. Howard McDavid

For my experience with modeling software, most of it comes from 3+ years working with Blender. I started out in Blender version 2.78c. At that time, 2.78c was far outdated, as version 2.8 had already been released. However, my modeling required an extension which, to this day, still has yet to be updated, and only works in 2.79 or lower. In addition, the tutorials I was using used 2.78c, because they were specifically tailored to the game I was creating assets for. As a starting model, I chose to model an Amtrak Sleeper car (Figure HM.28). Following an online

diagram, I created the model following several tutorials that showed the creation of objects, texturing, and exporting.

Figure HM.28: Amtrak Sleeper 2230 in Phase III

From there, I went through and created more models. After creating the passenger car, I started going through some diagrams found on Google and even some purchased through marketplaces such as a website known as RailfanDepot. After a year and a few months, I set to work on my most ambitious project at that time. I was taking a class here at UCF. That class was Harry Potter Studies. Part of the class is to have a film presentation, so I volunteered to create a model of the Hogwarts Express. At that point, I hadn't created a steam locomotive, so it was a first. Using some diagrams of the locomotive featured in the films, I set to work creating a model of it. Based on screenshots of the model on Discord, I believe that was started on March 6/7, 2020 and I had a working locomotive in-game by March 10. I had the final product done by the 14th (Figure HM.29). By the time the assignment came around, COVID happened, so we were unable to really get together and finish the project like we had intended.

Figure HM.29: A model of the Hogwarts Express

During the stay at home period, I learned a few new tricks, one of which being a better understanding of Blender animations. Using that, I undertook a project to recreate the Walt Disney World Railroad locomotives and passenger cars. I was successful with the locomotives (Figure HM.30), but the passenger cars are still only 1/4 done, and likely to remain that way due to Disney's unwillingness to provide style designs from the trains. After that project and a few others, I finally decided to upgrade to Blender 2.83 LTS, which brings me to a basic model of the fort (Figure HM.31).

Figure HM.30: The four locomotives of the Walt Disney World Railroad

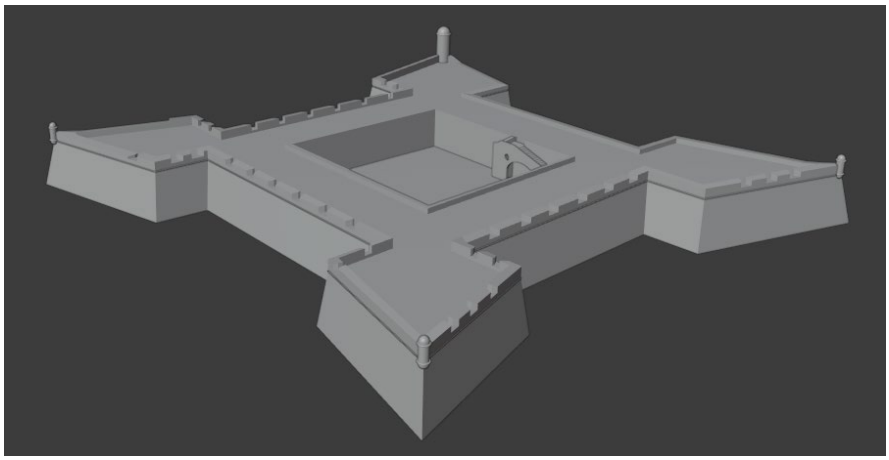


Figure HM.31: A 3D model of the fort

15.3.4. Juhwan Park

With our project I have been solely working on 3D modeling. I learned how to use Blender as I continued modeling, and have learned how much effort it takes to make objects look perfect in terms of dimensions and symmetry. It was also difficult to decide the model between the images that were taken and the blueprints available as sometimes there were disagreements between the two resources and oftentimes one would be less symmetrical than the other. So there were many instances where I had to choose symmetry over the resources and others where the images were followed instead of the symmetry.

I also made many mistakes and sometimes, such as when I have made too many new changes or have closed and reopened the file, undoing my actions were unavailable so this also hindered the modeling process.

The laptop that I was working with also had a rough time keeping track of all the objects that were in the file so I had to join multiple objects into one which also made it hard to make adjustments as specific vertices had to be selected instead of selecting objects.

Regardless, Blender has allowed me to model all the objects modeled so far in detail and through this time I have become more adept in modeling through Blender.

15.3.5. Enrique Rodriguez

For this project, I have chosen to model my assets using Autodesk Inventor. Although most of my group is working in Blender, I found the UI to be very confusing as I am used to working using blueprints and constraints. The UX was also quite weird as menus would disappear and only be found through long settings lists without explanations, even with tutorials I didn't see myself getting work done efficiently using Blender.

I have done several classes that worked using engineering guidelines and switching from constraint/dimension driven modeling to geometry driven modeling was very confusing for myself. I also made dimension based sketches of many of the assets I planned to create using measurements we got from our 1st outing to the fort and trying to recreate those sketches in Blender is a lot of work whereas in Inventor, it's almost a 1-to-1 copy of what I have drawn on paper. Additionally, Inventor still has some compatible file types that can be used in Unity so there aren't any issues with compatibility.

During the 2020 Pandemic, I had the opportunity to work with 3D modeling for an internship I had and was allowed the flexibility to use any modeling software I deemed the best for the job I was given. I was reintroduced to Inventor and Maya here as well as various other products such as Solidworks. I found the content creation pipeline in Inventor to be very straightforward compared to its counterparts and for the work I was doing, I didn't need the additional tools many of its counterparts provided.

15.3.6. Phillip Zou

I have experience with Unity and C# where I developed a small game for a class called *AI for Game Programming* at UCF. This has been my only experience related to any of the core technologies used for developing a game.

16. Assets

Throughout the fort, various door models are not interactable to guests due to possible damage occurring to historical items. The primary drawbridge to the fort cannot be raised and lowered on a daily basis due to the possibility of the original wood breaking down further. However, for this project, the sponsor and fort representatives want all drawbridges and doors interactable by the player. The player will trigger these manually but an automatic option is provided based on a trigger system. This system is explained in further detail in section 4.0.

16.1 Asset Store

The Unity Asset Store is a catalog of self-made assets and libraries created by Unity and various Unity users. Most assets are free but higher quality and larger asset packs can also be sold for a price. A majority of our assets are created by ourselves, however, several assets were used from the Asset Store such as some textures and general objects like barrels. The reasoning for this is to prioritize more important features on the project. Some assets like barrels are free and recreating the barrels from the fort 1-to-1 simply isn't worth the time invested for the output gained. Textures can also be bought here which saves a lot of time on our end as photo scanning the entire fort to gather textures is not feasible. There are far too many surfaces to scan and as the fort is a public place, waiting for the room to clear to take scans isn't feasible either.

17. Platform Future

Our task for this project is to create a virtual model of Castillo de San Marcos. With this task, we must create a new platform specifically for this task. This platform we are creating gives rise to some possible future projects using the same platform. Using our platform, we, or anyone who has the source code, can create either extensions for our VR experience or a whole new experience with another location.

17.1 Extensions

The extensions to our project would be relatively simple or they may be an overhaul. This distinction is decided more by the person, or people, deciding what they want to do. If they wanted to add a feature, like a new minigame, or even some smaller details around the fort, it would be fairly simple. They would only need to add some additional code if it requires it, as well as create the new models.

Should the new team do an overhaul of any features, it may be more difficult. If, for instance, they wanted to create an updated model of the fort itself, the team would need the blueprints of the fort, a plethora of reference photographs from the fort, and a decent length of time to model the fort itself. In addition, the Unity game would need to be updated to accommodate the new model, since the game was created with a specific model, with the hitboxes specifically tailored to said model.

17.2 New Experiences

This platform we are creating gives an excellent opportunity when it comes to experiences not related to Castillo de San Marcos. While it may not be the same for every experience, the guidelines below can give a great background for how another experience can be created using our platform.

17.2.1 Guidelines

1. Research

   The first step in any good project is research. For this type of project specifically, research is key to getting things correct. The very first thing a team should do is research the location they are tasked with creating. This research may include:

   - Official Websites
   - Blueprint Searches
   - Virtual Tours
   - Physical Visits
   - Etc.

Official websites may not always exist because some landmarks have not been properly taken care of. However, in the many instances where websites are available, they would be one of the most helpful resources, especially in the research phase. Websites for historical landmarks typically have a very high standard when it comes to the history of said landmark and what is disseminated through the history section of the website. This makes the website a perfect location to do research on a historical landmark.

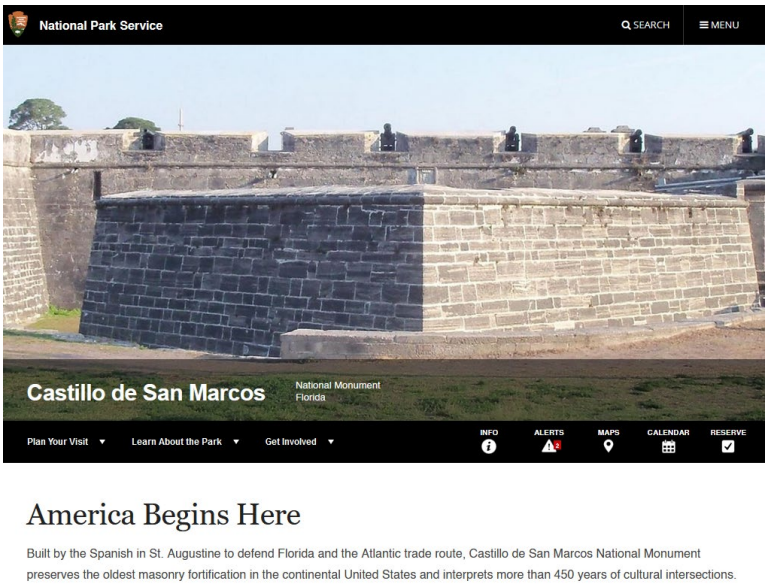


Figure 17.2.1.1: An example of a historical landmark's website

Searching for blueprints can be difficult at times, and, for many things, downright impossible without paying thousands of dollars for a survey. However, many historical landmarks in the United States have blueprints, especially if the location was accessed in the mid 1930s by the Works Progress Administration. Through the WPA and the National Parks Service, many historical landmarks, such as forts, were purchased and subsequently surveyed to give an idea of what the buildings needed for proper care and attention. In the case that the blueprints are not publicly available, the new team may need to either pay to access them, or contact the owners of said blueprints

to get access. Should blueprints not exist of the landmark modeled, it will be significantly more difficult, however not impossible.

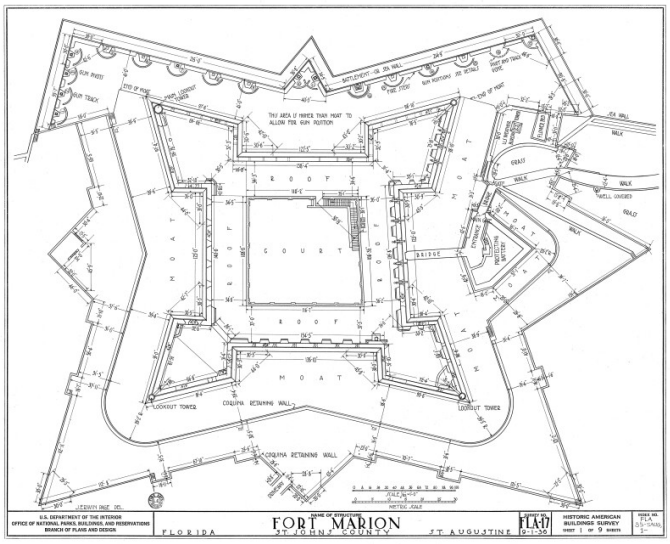


Figure 17.2.1.2: An example of a blueprint

Many historical landmarks, like Castillo de San Marcos, have virtual tours. These virtual tours, whether they be done through Google Street View or through a school project, can be an immense help if physically accessing the location is impossible.

The best case scenario for this is that the landmark the team wishes to create is physically accessible to them. In this case, the group members may visit the location as needed to collect pictures, take measurements, and gain a better understanding of the location they are modeling.

2. Development

a. Modeling

The modeling of a landmark is one of the biggest parts to a project like this. When it comes to modeling there are different philosophies. Some people prefer to model with extremely high poly counts, others like to keep poly count low in an effort to preserve performance. For this kind of project, especially with the VR component, it is best to keep a low poly count when possible, since when a VR system tries to draw too many vertices, the performance drops significantly.

If the team was able to source blueprints, they are well on their way to getting their project done. If not, here's where the difficult part comes in. Without blueprints, modeling becomes more tedious, as the modeler will need to keep proportions correct. Here is where a physical visit to the location comes into play. Using pictures taken at the landmark, it is possible to use an addon to a modeling application, like Blender, to add a camera in the place where the photo was taken, with the proper angles to give an exact line to model along. From there, the modeler can use any measurements taken to get the model scaled properly.

b. Coding

For coding, most of this is done through our platform, likely without need for changes unless something drastic must occur. Most coding the team will need to do is for additional features they want to add, such as minigames to add a fun aspect to the landmark.

3. Testing

Testing is an extremely important step to creating a project like this. Not only does the project need to look good, it needs to have excellent performance. The first round of testing for this kind of project should be the integration of the custom built 3D models and the Unity Game Engine. While it is good to test VR, since the VR implementation would already be done for this project, VR testing would only be for evaluating performance metrics to ensure the industry standard frame rates of 90+ fps are met. Additional testing would include the testing of newly implemented features, like the minigames the team may want to include. This testing will ensure all systems work in the game.

4. Deployment

   If a team has reached this point, it means the development has been completed, testing is done, and the game is polished enough to be given to the public. This step can be done through many different stores. The most common stores developers use are the Steam and Oculus stores, but they are not the only ones. The project a team has created can be uploaded to the store and any user may be able to download and play the experience.

18. Workflow

To keep development organized, it is important to implement a standardized workflow that members are expected to follow. For this project, we use JIRA to keep track of tasks and assign tickets in addition to a feature-branch workflow with Git as our source and version control.

18.1 Definition

The workflow is as follows:

1. Move a **To Do** task assigned to the member in the JIRA board to **In Progress**.
2. Create a new local branch in the git repository named *feature/TASKNAME*.
    - If additional members are required to work on the branch, push to remote.
3. Complete the task, making additional progress commits if needed.
4. Test the changes to ensure they are as intended.
5. If needed, commit any uncommitted changes.
6. Merge feature branch into *main* branch.
    - If a merge conflict occurs, manually resolve each conflict and then continue with the merge.
7. Move **In Progress** task in the JIRA board to **Done**.

By implementing the above workflow, it is easy to keep track of who is working on what task, how long they have been working on a task, which tasks have been completed and which tasks still need to be worked on. In addition, both Git and JIRA provide tools that help to visualize progress over time.

19. Design

19.1. Versions of the fort to be implemented

Our team would like to implement different variations of the fort within this Virtual Reality experience to provide a more in-depth experience for the

users. We have decided that it would be very interesting to provide representations of the fort from when it was built and originally used, as well as present day, so our goal is to allow the users to choose between those two options.

One downside is that we do not have a lot to work off in terms of recreating the structure as closely as possible to what it looked like when the Spanish finished building it in the late 1600s. We will be basing it off computerized models, like figure 7.1.1.2, as well as the knowledge the park rangers have on the history of Castillo de San Marcos and the area at the time.

As for the present-day experience of the fort, the users will be able to view the fort as it would be if they were to visit it in person. The exhibits and displays will be recreated as they appear. We will implement a tour of sorts to show the user around Castillo de San Marcos as if they were being led by a tour guide. However, this tour would be optional for the user. They will also be able to explore the area freely if they choose to do so.

19.2. 2D Graphics

To display 2D graphics in Unity, such as images we can make use of sprites. Sprites are essentially just normal textures that can be used as needed. However, the loading method of sprites is a bit peculiar. Instead of loading each sprite as a separate image file, we load them from a sprite sheet. Each sprite sheet is just a larger image containing multiple other images (which will be used as sprites). The benefit to this is that only a single image needs to be loaded. Unity provides a Sprite Editor tool that will allow us to easily manage sprites for this project.

19.3. Importance of 90FPS Lows

Meeting a performance requirement of 90 FPS lows is essential for the success of this project. VR devices typically render a frame twice, once for each eye. The left eye’s frame renders the left side of the view while the right eye renders the right side of the view, with some overlap.

Some devices rely on what's called asynchronous spacewarp (AWP). When running below 90 FPS for several frames, AWP devices will automatically drop down to 45 FPS and synthetically generates every second frame.

For some people, using VR devices at 45 FPS can cause motion sickness and to others it can simply make gameplay uncomfortable. To prevent such issues from occurring in the first place, we will be closely monitoring performance metrics via monthly testing on several VR devices. [31]

19.4. Device Detection

As the project will support multiple VR devices, we will need a way to detect what way is being used. Luckily, Unity provides a way to detect such devices. [32]

```
var leftHandDevices = new List<UnityEngine.XR.InputDevice>();
UnityEngine.XR.InputDevices.GetDevicesAtXRNode(UnityEngine.XR.XRNode.LeftHand, leftHandDevices);

if(leftHandDevices.Count == 1)
{
    UnityEngine.XR.InputDevice device = leftHandDevices[0];
    Debug.Log(string.Format("Device name '{0}' with role '{1}'", device.name, device.role.ToString()));
}
else if(leftHandDevices.Count > 1)
{
    Debug.Log("Found more than one left hand!");
}
```

Figure 19.4.1: Example code for device detection in Unity

The above code snippet will attempt to find a left-handed controller: Since we can detect the devices, we can write logic to handle each device or even support mouse and keyboard in the event that no VR devices are found. A flowchart for detection is shown in Figure 19.3.2.

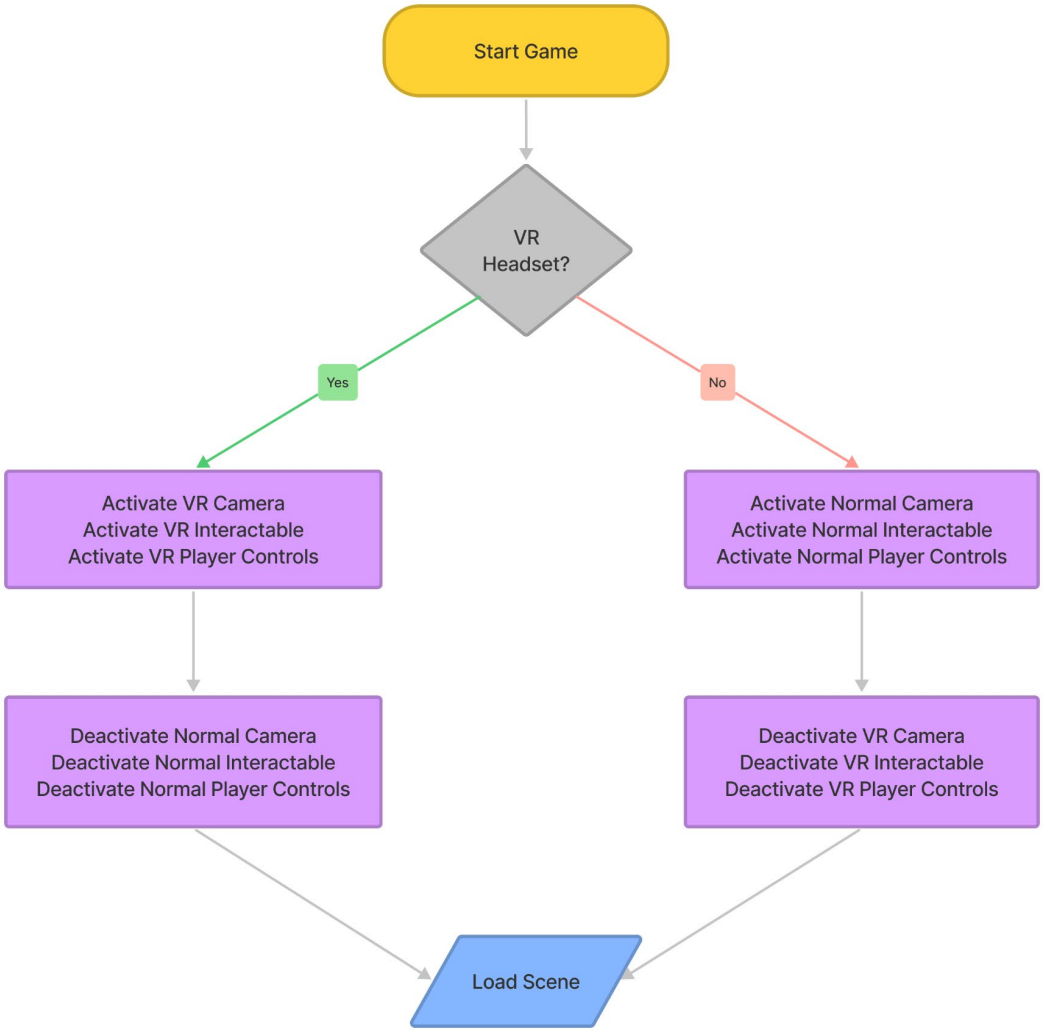


Figure 19.4.2: Flowchart for Device Detection

19.5. Audio

Audio can be implemented directly through Unity's provided feature set. To do this, we define an AudioSource that will play the audio and then

pass it a valid AudioClip. An AudioClip can load data from an audio file with supported formats including .aif, wav, ogg, and mp3.

For VR implementations, we can make use of ambisonic audio which represents a sound field that can be rotated relative to the user's orientation. Ambisonic audio in Unity requires the format to be that of a multi-channel B-format WAV file with ACN component ordering and SN3D normalization. Being able to use ambisonics as well as being a lossless audio file format and the limited number of audio files this project will include makes WAV our preferable audio file format.

AudioClips can be played when certain triggers occur, such as the user firing a cannon, opening a door, or simply walking along a stony path. Such clips can be layered on top of one-another to create dynamic and immersive audio for the scene.

Most of the sound effects needed for this project are generic in subject and commonly used. As a result, we should be able to require most assets for free provided that we give proper credit to their creators.

Sound assets that we intend to implement include:

Category	Sounds
Movement	● Walk on wood surface ● Walk on stone surface ● Walk on dirt surface

Interactions	• Metallic clang • Explosion (cannons) • Water splashing • Fabric rustle • Door open • Door close • Grab object
Ambience	• Bird sounds • Ocean sounds • Wind
Menu	• Menu open • Menu close • Button select

Figure 19.5.1: Example code for device detection in Unity

19.6. Text Areas

Upon entering a specific area, a text description relating to that area will appear if it has one. For example, upon crossing the bridges at the entrance of the fort, a popup display will appear that describes the construction and purpose of that bridge. The same goes with a variety of areas around the fort. Such areas include the chapel, prison, courtyard, and moat along with many others.

This functionality can be achieved by using a transparent GameObject with a Collider component and defining logic for the OnTriggerEnter() method that is invoked when a different collider collides with the previously defined GameObject. To prevent an object that is thrown into this collider from triggering the text area logic, we can specifically limit the logic to the condition that the colliding GameObject is the player agent.

19.7. Displays

In addition to text areas, we will also utilize separate displays. These displays must be interacted with directly by the user. Upon doing so, additional information about a specific feature will be displayed. This is useful for conveying information that may not necessarily be relative to an area. It also allows us to expand on existing text areas by including separate displays inside of them.

These displays will also allow for images to be displayed. Such images might include artwork of an important figure or historic images taken when the fort was in active use by its occupying military.

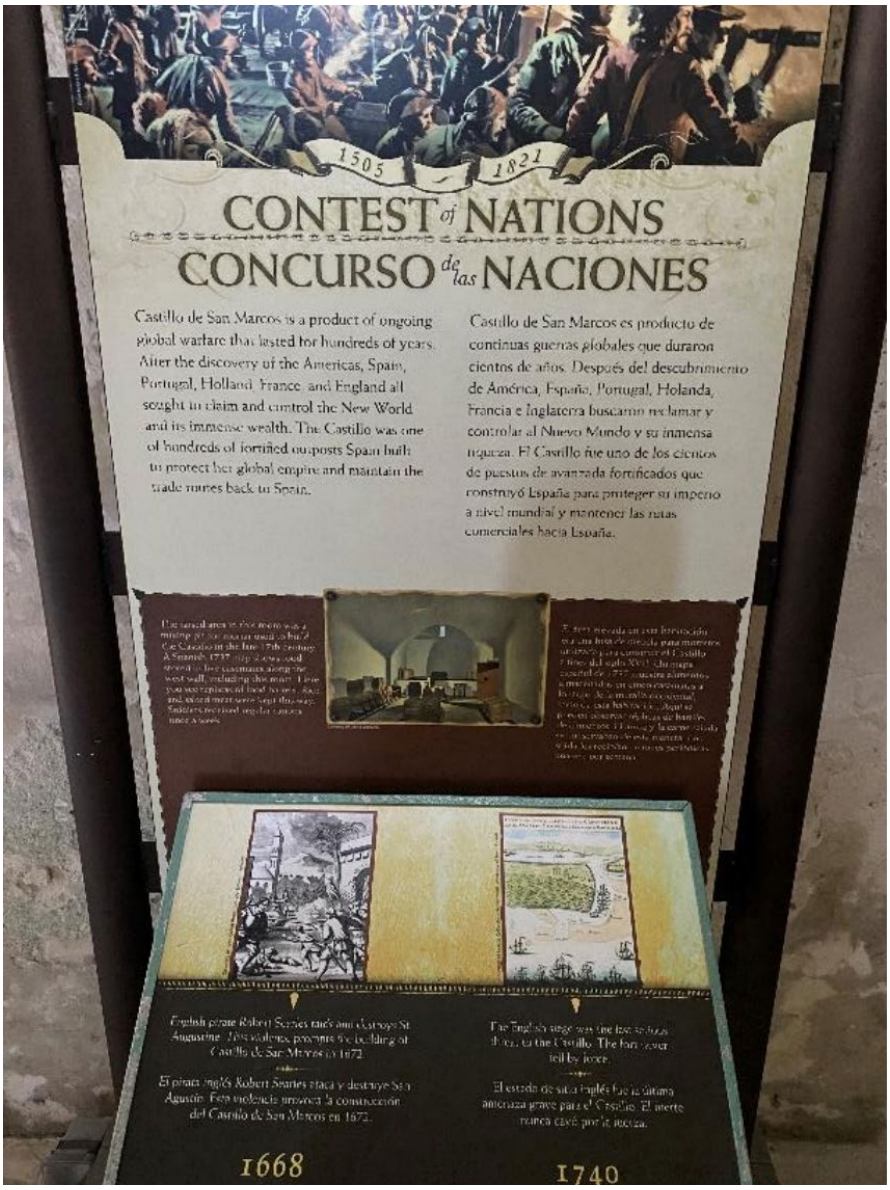


Figure 19.7.1: Example of information that will be displayed, not representative of how it is displayed

Due to the volume of the information that a display could contain, the user will not be able to move or rotate while viewing the display. This is done to improve the user experience as it can be hard to read and keep track of moving text.

19.8. Localization

In order to provide proper localization in an easy-to-maintain manner, this project implements its own resource manager for strings called TextManager. Each string consists of an *id* and a *value*. The *id* is simply an identifier used for referencing the string, while the *value* is the actual contents of the string. Each string can be fetched by its immutable *id*. This allows for the value of the string to be modified at any time without needing to modify any other part of the system. For example, in the main menu user interface, there is a button for “options”. When the main menu is loading, the actual name of this options button is obtained using the TextManager and fetching the string with the *id* “main_menu_btn_options”. While this *id* remains unchanged, the actual value returned from the id could be defined as “options”, “settings”, or even “opciones” (Spanish for options).

For storing the data for these strings, the TextManager uses a dictionary where *id* is the key and *value* is the value. Using this data structure is beneficial for this task as it has an O(1) constant lookup time for values when given a key.

	A	B
1	main_menu_btn_start	Explore Fort
2	main_menu_btn_options	Options
3	main_menu_btn_exit	Exit
4	options_menu_graphics	Graphics
5	options_menu_audio	Audio
6	options_menu_gameplay	Gameplay

Figure 19.8.1: Example strings file for localization viewed in Microsoft Excel

The *id* and *value* of each string is stored in a CSV file as two separate columns. This means that the file could be opened in software such as Microsoft Office Excel, which makes maintaining and updating the file user-friendly to those familiar with Excel. In the above image, each row is a string where column A represents the *id* and column B represents the *value*.

Additionally, there can be multiple string files. The only requirement is that each file follows the format *strings.{LOCALE}.csv*. By default, the system will attempt to load *strings.en-us.csv*. In order to add a new language, one just needs to create a new strings file containing all of the ids and values for that locale. For Spanish, that file might be called *strings.es-es.csv*.

19.9. Gameplay

When the user opens the game they will enter the Main Menu. From there they can play the game, access the tutorial, change settings by accessing the Options Menu, open the Credits menu, or exit the game.

Within the game, the player can roam around the fort and interact with NPCs and placards. The NPCs and placards will provide information and historic details about the fort. Some NPCs will have more in depth interaction trees.

There will be minigames available throughout the fort and denoted on the minimap for the player to play if they want to. The minigames will have instructions on how to play them when the player interacts with them. There will also, hopefully, be an audiovisual tour that a player can opt into.

The audio/visual tour will function similarly to a quest system in a 3D game. There will be a quest marker to indicate where to start the audiovisual tour as well as indicators placed in the game world as well as on the minimap to show where to go to continue the tour.

A flowchart of the gameplay is shown in Figure 19.9.1

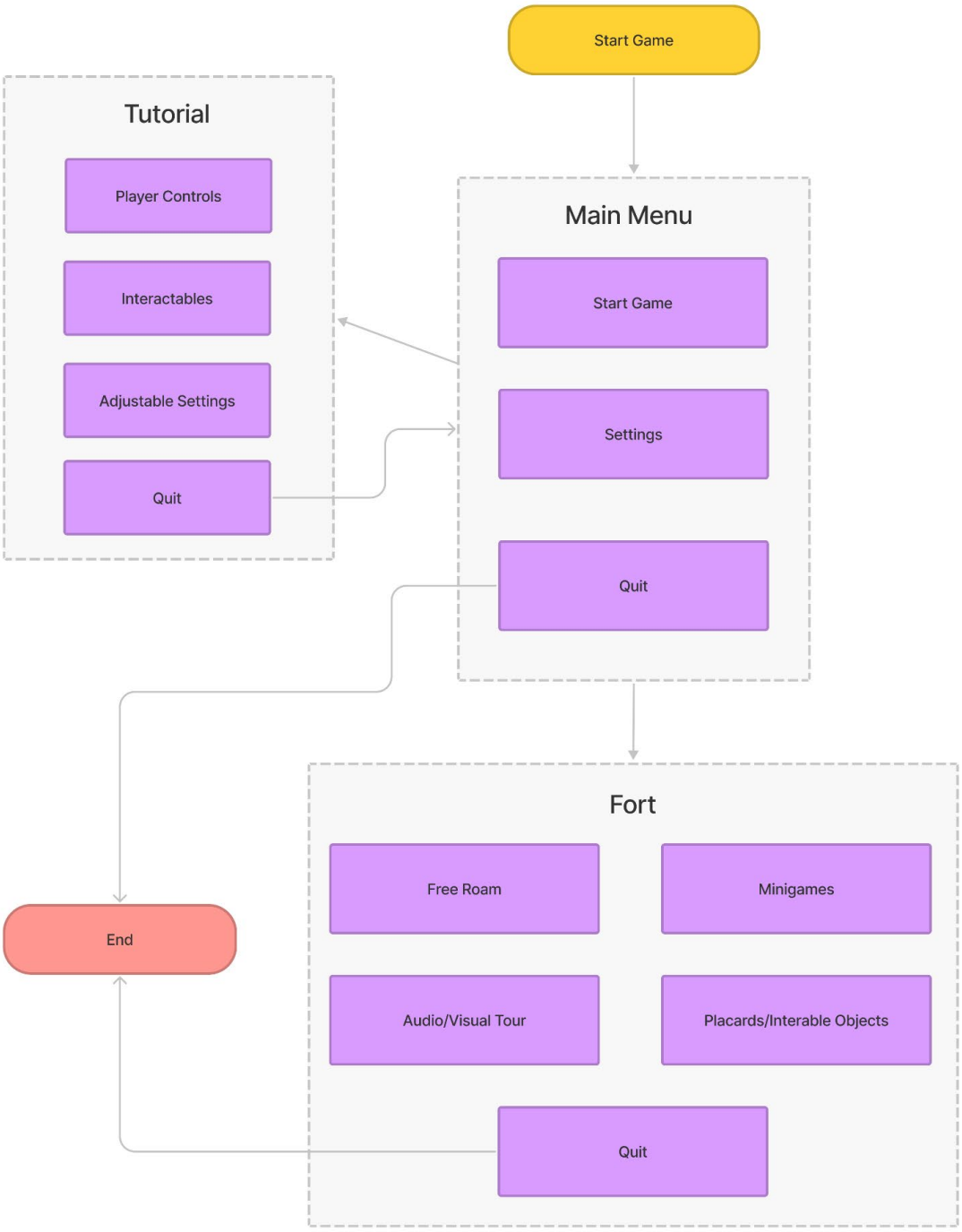


Fig 19.9.1: User Interaction Flow Chart

19.10. Main Menu (Start Screen)

Upon starting the VR game, the user will be brought to the main menu screen. This screen will feature several different options for the user to choose from before they begin exploring the fort.

Option Label	Description
Explore Fort	Loads the scene and all related components for VR interaction.
Options	Opens the **Options Menu**, which allows the user to configure their game settings as needed.
Tutorial	Opens a brief tutorial page that explains player controls, how to tell what objects/NPCs can be interacted with, and some important settings the player can change.
Credits	Opens a new display showing the credits for this project. Includes credits for developers, contributors, and any creators of additional assets used.
Exit	Exits the game.

19.11. Options Menu

This menu allows the user to configure their game settings. It can be accessed from the Main Menu or when pressing the inputs controller's corresponding "Menu" button while exploring.

Graphics

- Quality [Select] – Adjusts the quality levels of assets.
    - *Low* – Half resolution textures, no shadows, no MSAA, low LOD bias.
    - *Medium* – Full resolution textures, medium resolution shadows, medium shadow distance, no MSAA, medium LOD bias.
    - *High* – Full resolution textures, high resolution shadows, high shadows distance, MSAA x2, high LOD bias.
- Foliage [Toggleable] – Turn on/off extra foliage details.
- Birds [Toggleable] – Turn on/off birds flying overhead.

Audio

- Master Volume – Adjusts volume of all sounds.
- Narration Volume – Adjusts volume of narrated text.
- Effects Volume – Adjusts volume of effects such as cannonballs, birds, and water.

Gameplay

- Toggle HUD (enabled by default).
- Adjust mappings of actions by selecting an action and then pressing the new control you want to map to the selected action.
- Reset control scheme with confirmation.

19.12. Navigation

19.12.1. Map

A mini-map displayed through the HUD can be used by the user to navigate the grounds. This map will include an overhead display of the fort along with the current user's location in relation to the fort. Furthermore, an arrow-like icon will be used to indicate the user's location and it will rotate to also show the direction that the user is facing.

The mini-map can also be expanded to that of a full-size map with additional functionality such as the ability to zoom in and out. If a user decides that they do not want the mini-map to be displayed for whatever reason, they can disable it by toggling off the HUD in the options menu.

19.12.2. Movement

Movement around the fort will be free roam. However, there will be some

restrictions in place to prevent the user from performing certain actions. For example, invisible walls will be in place to prevent the user from jumping/falling off of the fort. Furthermore, a boundary within the scene will be in place to

19.13. Audio/Visual Tour

The audio/visual tour will work similarly to the tour at the actual fort. There will be a quest marker, denoted with a "!", that shows where the tour will start. Like the real life tour, the audio/visual tour will have set locations where the player will travel to in order to continue the tour. At each location the prerecorded audio will play. The recordings will be done by a park employee that gives tours and we will coordinate with them to decide on locations and content.

When the audio is playing, the game may take control of the screen to change the display so that it follows along with the content of the audio portion. Depending on the content, UI elements displaying more detailed pictures, text, or videos may appear for the player to read and interact with. For players who cannot access the audio portion, there will be a subtitle option available that will display the transcribed audio at the bottom of the screen.

19.14. Tips

When exploring around the premises of the fort, those unfamiliar with our VR game may have a limited understanding of what they are able to do. In response to this problem, we will be implementing a “Tips” system that will give the user explanations about different interactions. Of course, for returning users or those that want to figure out everything on their own, this feature is completely optional and can be toggled in the Options Menu under Gameplay. It should also be noted that each tip will only show once per explore session. To view a tip again, the user must restart with a new explore session. This is done to prevent the user from being spammed with content that they are already aware of.

These tips may include the following:

Tip Trigger	Tip Text
Approach Drawbridge Mechanism	“You can raise or lower the drawbridge by spinning this mechanism.”
Approach Cannon	“Try speaking to the cannon instructor to learn how to operate this cannon.”
Approach Display	“You can interact with this display to view additional information about the fort.”
Approach Door	“You can open or close a door by interacting with it.”

Grab Cannonball	“This can be loaded into the cannon to fire as a projectile.”
Grab Worm Tool	“This tool can be used to clean any remaining wadding from the barrel of the cannon after each shot.”
Grab Sponge Tool	“This tool can be dipped lightly into a bucket of water and inserted into the barrel of the cannon to remove any remaining sparks after each shot.”
Grab Ladle Tool	“This tool can be used to shovel gunpowder into the barrel of a cannon.”
Grab Rammer Tool	“This tool can be used to pack the contents inside the barrel of the cannon together.”
Grab Powder Horn Tool	“This tool is used to pour gunpowder into the priming hole.”
Grab Linstock Tool	“This tool is used to ignite the priming hole of the cannon to fire it.”

Figure 19.14.1: Chart shows examples of tip triggers (what activates the tip) and tip text (the actual tip that gets displayed).

19.15. Cannon Interaction

There are several tools used in preparing and loading a cannon. These include a worm, swab, sponge, ladle, rammer, linstock, and powder horn.

Figure 19.15.1: Cannon prepping tools

From left to right: worm, swab, sponge, ladle, rammer, linstock

The steps are as follows:

1. If the cannon is up against the wall, pull it back a few feet for easier access.
2. Use the worm tool to remove any wadding that may remain in the barrel from the previous shot.
3. Use the sponge tool dipped lightly into a nearby water bucket and inserted into the barrel of the cannon to remove any remaining sparks from the previous shot.
4. A cartridge consisting of a cannonball and a bag of gunpowder is loaded into the barrel of the cannon.
    - If a cartridge is not being used, a ladle will be used to shovel gunpowder into the barrel, followed by a cannonball.
5. A rammer tool is inserted into the barrel of the cannon to push everything down.
6. Push cannon into firing position (using a piece of wood and leverage).
7. If a cartridge was used, a vent pick is then stuck down the priming hole to poke a hole in the gunpowder bag.
8. Fine powder is then poured from a powder horn into the priming hole.
9. A linstock equipped with a slow burning rope is then lowered on top of the priming hole to light the cannon.

Each cannon will have a state associated with it that specifies where it is in the firing sequence. As many of the rod-based tools require precision in order to get them into the barrel of the cannon, a certain tolerance will be allowed for snapping these tools into place following which the user can use the tool on an individual axis allowing for forwards or backwards (pull/push) movement until the tool is pulled out of the cannon. When shooting the cannon, there will be a loud explosion sound as a smoke

particle effect and a flash of fire spew from the barrel of the cannon. Such visual effects can be achieved with a ParticleSystem component. The cannon will also roll back at a simulated recoil velocity. The cannonball will be fired as a projectile with a velocity of approximately 100 meters per second with some randomized variance. Upon impact, an appropriate effect will be played. For example, upon hitting the water, a large splash effect will occur. When hitting a ship, the ship will break apart before sinking a few seconds later.

19.16. Cannon Tutorial

A separate tutorial will be included to teach users about the proper procedures for firing a cannon. This tutorial is completely optional and can be accessed at any time through a cannon instructor NPC. The tutorial will teach the user step-by-step on how to operate the cannon. These steps are the same as the ones indicated in the description of the cannon interactable. It will highlight each tool when it is needed and direct the user towards where they need to use the tool and tips for how to use it.

19.17. Cannon Minigame

The cannon minigame will consist of the user attempting to shoot the cannons and hit indicated targets. The user will use the cannon to shoot as many targets as possible. For each target they hit, their score goes up by 1. If a user misses 3 times (total), the minigame ends and their resulting score is displayed. If a user chooses to, they can leave the minigame at any point in time and continue exploring the fort.

Given the large number of steps associated with loading and firing the cannon, we plan to implement an NPCs that will help speed up the process and reduce complexity for the user. The goal of the cannon mini-game is to be fun and not cumbersome. To give the user more control of how they want to play the minigame, they can select which actions they want to be responsible for upon starting the minigame. Of course, the user will always be responsible for aiming the cannon. As for firing the cannon, they can either do it themselves or order for one of the NPCs to fire the

cannon. Either way, the user is still responsible for the timing of when the cannon is actually fired.

Figure 19.17.1: Likely candidate model for use as the ship in the cannon minigame.

19.18. Simulating Weight For Interactables

Due to the fact that the user is not interacting with objects directly, but through virtual reality devices, simulating weight proves to be a challenging problem. As there isn't any sensible way to restrict the user directly, we must restrict the user in the virtual environment in a way that is realistic but not an annoyance.

While it would be fun to throw a 12 lb cannonball as if one were throwing a baseball, it is not realistic at all and would only serve to hurt the immersive environment. To solve this without overcomplicating the problem, we can set a max velocity for movements with an object while it is being held by a user. This limit would be different for a user using one hand to pick up an object versus both hands. Additionally, some interactable may only move through other methods. For example, moving a cannon may not be done through normal means. Rather, the user would need to make use of leverage with a piece of wood to move it. This is consistent with how the cannons were moved in and out of firing position when the fort was for military use.

19.19. Simulating Physics For Interactables

Different game objects require different physics systems depending on what they are. For example, the physics of interacting with a musket is different from that of clothing. Luckily, Unity provides solutions to such problems. For the musket, applying a physics system is as simple as adding a Rigidbody component to the game object [34]. As for clothing, Unity provides a Cloth component that can be used for simulating cloth physics [35].

Rigidbody Component

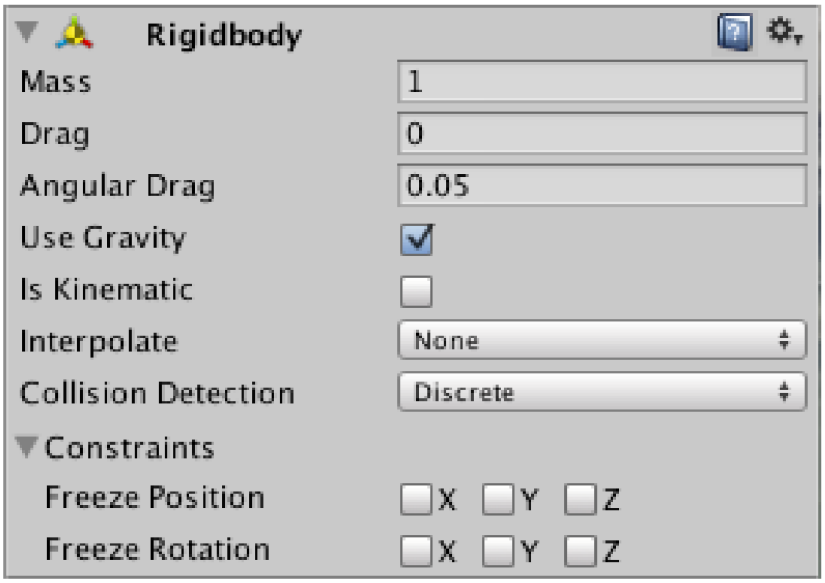


Figure 19.19.1: Example UI of RigidBody component in Unity

Cloth Component

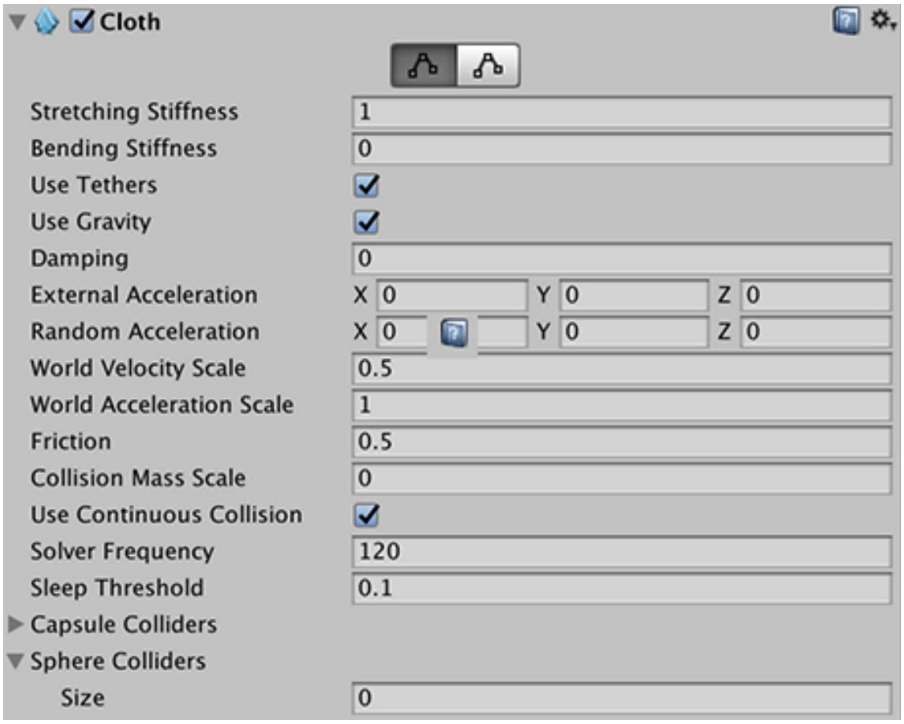


Figure 19.19.2: Example UI of cloth component in Unity

19.20. NPC Dialogue System

19.20.1. How it Works

An NPC system will be included that offers flexible interactions. The design of the system features a window displayed in VR and relies on loading text via the string resource manager as the content text depicting the NPC's side of the conversation. Optionally, buttons can be included with text loaded from the string resource manager along with a separately defined action that is invoked when each button is pressed. An NPC could exist merely for dialogue or could offer buttons to initiate specific actions.

19.20.2. Why it Matters

Populating the scene with non-player characters (NPCs) offers many benefits to the immersive experience of the scene. Not only does it add life to the scene, but it also allows opportunities to provide additional functionality. For example, the user can interact with the cannon instructor NPC to access the cannon tutorial. Such an interaction offers a much better experience than simply entering a designated area.

By providing a system for placing and interacting with NPCs, additional NPCs can be added as needed. For example, if the project were to be expanded in the future, additional NPCs could be added to simulate the routine of those stationed in the fort.

19.20.3. NPCs with Dialogue

- **Fort Guide** – Offers a guided tour of the fort (if tours are implemented in time).
- **Cannon Instructo**r – Allows the player to access the cannon tutorial as well as the cannon minigame.
- **Spanish Garrison (Various)** – A few (fictional) members of the fort's garrison offering a variety of dialogue relating to the fort.

19.21. Skybox

19.21.1. What is a Skybox?

New projects created in Unity all end up with the same default view. It begins with a blue to white gradient going towards the middle of the view, while the bottom half is all gray. This blue to white gradient is the default skybox, as seen in figure 19.21.1.1. It could pass as a very basic sky, but there is no detail or complexities like clouds. Due to how basic it is, most people end up adding a new skybox to their projects.

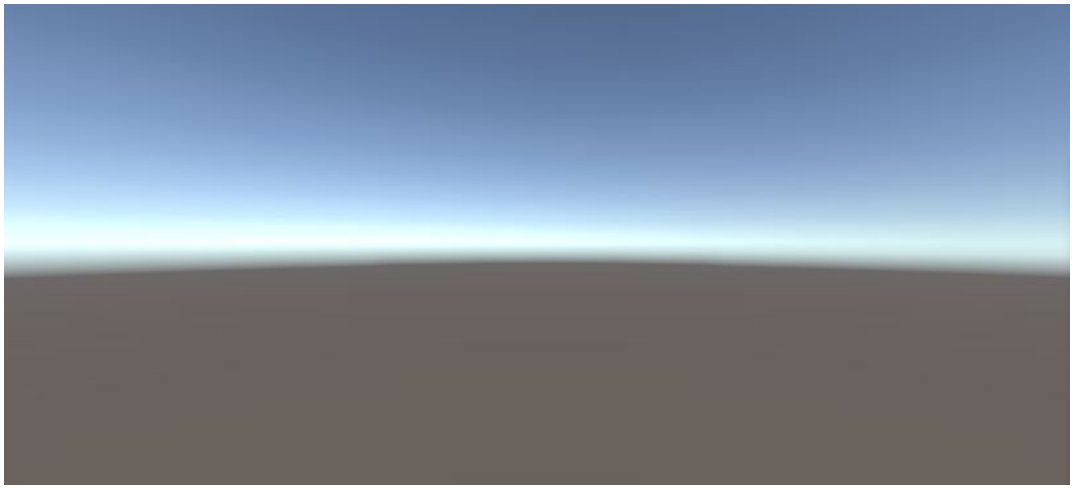


Figure 19.21.1.1: Default skybox on new Unity project

A skybox is essentially a scene that wraps around the entire horizon of the Unity scene. It adds a new level of complexity to the project, giving the impression that there's more detail added to the horizon of the scene, while it's essentially just an image. They are typically created as a cube map. These are composed of 6 images or textures to give you all angles needed to have a realistic environment according to the project. These 6 sides would be front, back, left, right, up, and downwards facing images.

19.21.2. Choosing a Skybox

There are many options to choose from on the asset store for skyboxes. There are realistic looking sky options, as well as some that give off a less

realistic and more of a game feel. There are also asset packs that contain the same cloud pattern but offer different lighting options. This would be very useful in the case of adding a time feature to the project. The light source (sun) appears to come from different directions while using these, which gives the illusion of different times of day.

If we were to add a time aspect to the project, we could implement a script to choose which skybox to use for different time frames. However, it would require a lot more than just a different skybox. We would have to consider shadows and change the light source placement. Our default project is in perfect lighting conditions, which would be around noon since the sun is directly overhead. Due to this, we don't have to worry much about shadows. Figure 19.21.2.1 displays ideal conditions for a skybox on our virtual reality simulation.

Figure 19.21.2.1: Example with skybox and lighting implemented

Another aspect this project could include would be different weather conditions. This would be weather relative to the location, so it could be rainy or overcast. This would be a combination of changing the skybox, as well as implementing rain into the project, but would lead to us needing to decide whether we want the rain to be as realistic as possible and puddle onto the ground, or just have the graphics of the rain as it falls and have it not pooled up on the ground. If the project follows realistic rain physics, it would be interesting to have it utilize the water drainage holes along the top of the fort.

Our goal for this project is to create our own skybox from images we took during our visit to Castillo de San Marcos. This would lead to the most accurate experience of visiting the fort as we can create. We were aware of this while visiting the fort for research and photos, so we made sure to take plenty of pictures of the surrounding environment as options to work with.

19.21.3. Implementing a Skybox

The simplest route to take while adding a skybox to a Unity project is to find one on the Unity Asset Store. Once the developer finds the skybox or skybox pack that suits their desires, they can click “Add to my Assets.” This will then lead to a pop up with two options, “Open in Unity” and “Go to My Assets.” The developer may click on the first option, which will redirect them to the Unity program they are working on with a new window open called the Package Manager. The developer must download the asset before importing to their project.

Once the asset has been imported into the project, it is finally accessible to be applied to the scene. The developer may return to their Unity project, find the Window tab in the top toolbar. Then, they must find the “Rendering” option, follow the arrow to the right, and click on Lighting. This will open a new window within Unity that contains all possibilities for the lighting and surrounding environment for the game scene. The developer may click the “Environment” tab at the top of this new window. At the top of the Environment page, there is an option called “Skybox Material,” which is utilized to change the skybox that surrounds the scene. As this option is clicked, the project will display a folder of all options for skyboxes that are included in the assets, which is called Select Material. The Select Material menu is displayed in figure 19.20.3.1. Any option may be selected, and the project’s skybox will immediately reflect any selections made.

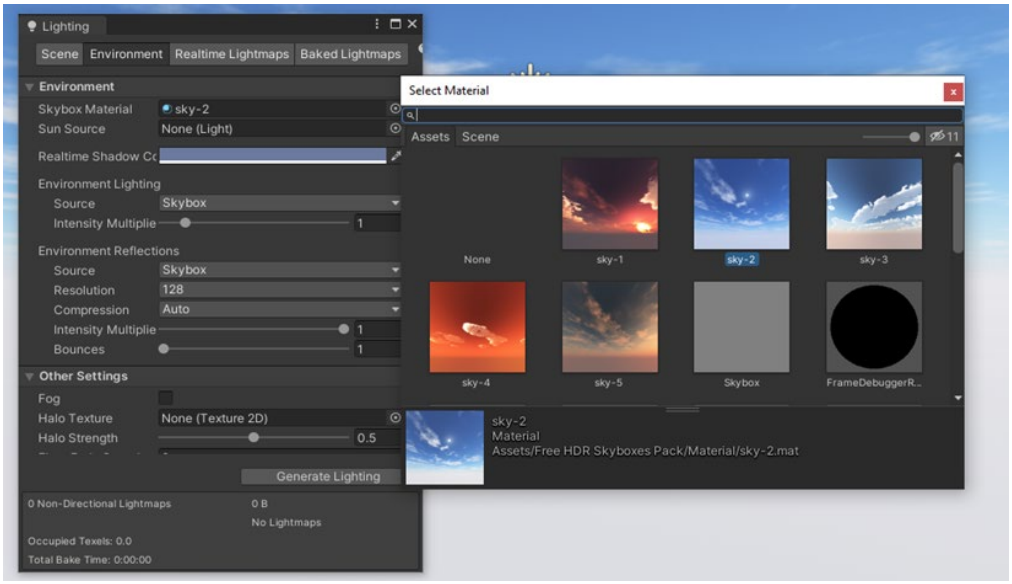


Figure 19.21.3.1: Lighting window and Select Material windows opened

19.22. Water

Since Castillo de San Marcos lies on the shore of the Matanzas River, it is important to implement water in this project. To do this, shaders are a nice option. By finding a good-looking and realistic water shader on the Unity Asset Store. Once the shader has been found and imported into the project, it can be placed as a once we have a riverbed laid out. The water can be positioned up to a certain height to fit within our constructed riverbed to make it look as realistic as possible.

19.22.1. What is a shader

To understand how this works, it's important to understand what a shader is. A shader is a program that allows for an easier method of taking meshes, textures, lighting, etc. and turning it into an output graphic. They can be animated using mathematical algorithms and applying them to the textures to modify the placement and coloration of the object. Reflective attributes can also be added to the texture, which will be essential within a water shader due to the sun reflecting off the water's surface.

19.22.2. Choosing a water shader

For this project, we needed to find a realistic water shader that closely resembles the water you would find in a river. Many of the free water

shaders offered on the Unity Asset Store have very clear water or are a bright blue, however, the Matanzas River is murkier and more difficult to see through. We have an image, shown in figure 19.22.2.1, that we took while visiting the fort to use as a reference to what the water would look like on an average day.

Figure 19.22.2.1: The water is blue, yet still murky and does not have too much motion due to wind

Using this as a reference, we found a shader that fit the overall look of the river. Implementing this shader is very simple once we have a rough riverbed designed. This shader allows for several modifications for almost every aspect of it. Figure 19.22.2.2 displays every modifiable aspect of the water shader applied to the project.

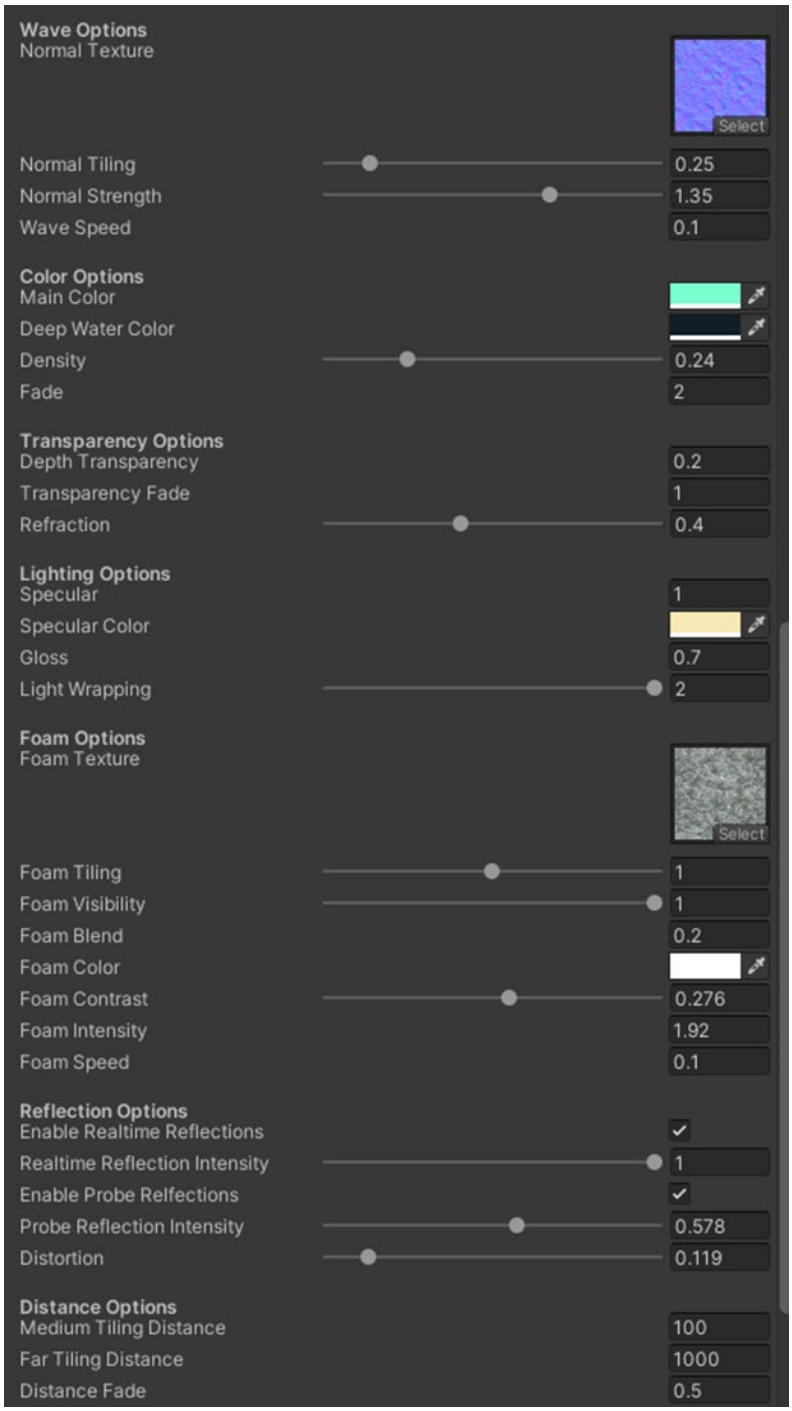


Figure 19.22.2.2: Settings for water shader

The normal tiling setting modifies the amount and sizing of water disturbance. The lower the value set for tiling, the larger the size of each tile is, which creates less water disturbance on the surface. Normal strength also modifies the disturbance, however, it's more so a current and wind

appearance. The speed setting is the period in which it takes for each simulated wave to run.

The density setting modifies the density of the water. The higher the density is, the darker the water will be when the water is shallow. For this project, we will use a higher density to replicate the appearance of river water more accurately. The fade value modifies the area where the water transitions between a shallow color and a deeper water color. Depth transparency deals with the topmost surface area of the water, which will be the lighter (shallow water) color. The larger this value is, the more transparent that light color is. Light wrapping modifies the amount of light that reflects off the water's surface. For this application, we will utilize a low value to have the water appear a darker shade.

The final important settings that are modifiable apply to the foam. Foam blend affects how much foam displays where the water meets the land at the shore. Foam visibility affects the transparency of the foam.

Figure 19.22.2.3 displays an ideal representation of how we would utilize the water shader as the water in the Matanzas River within our project.

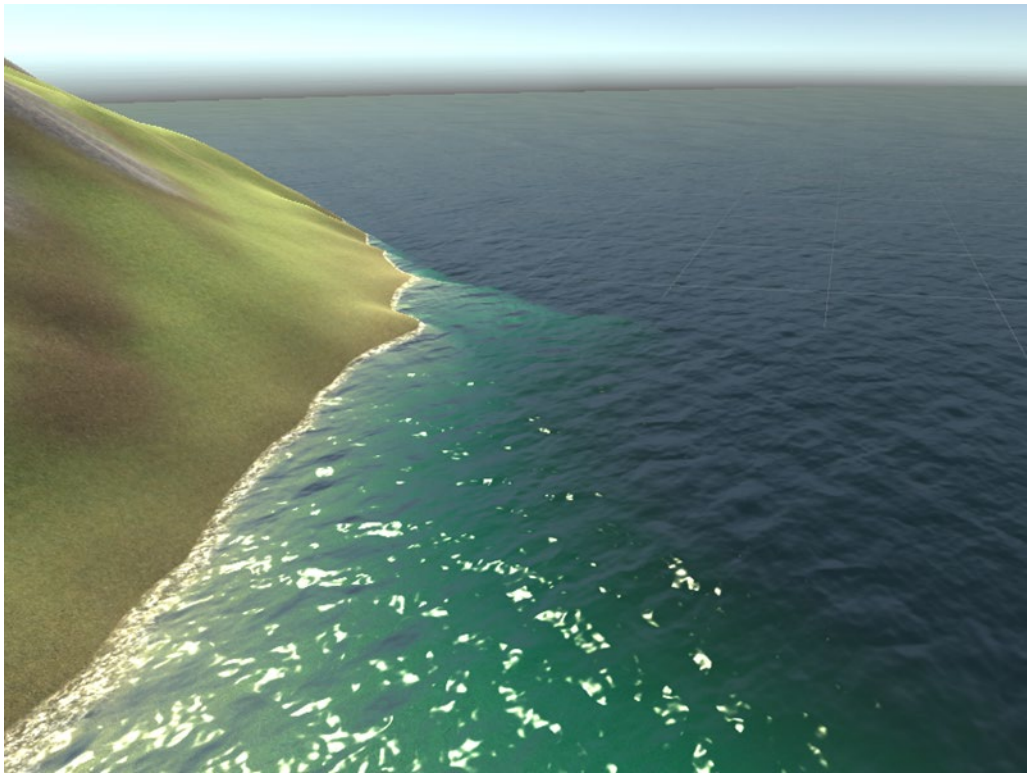


Figure 19.22.2.3: Water shader applied to a demo map to display all the attributes

19.23. Birds

Another prominent feature we want to add to our project's environment is birds. During our visit, we noticed that there were birds everywhere. They were flying overhead, in nooks and crannies inside the fort, in water drains, in trees, perched on the walls, and even more. Since the fort is on the Matanzas River, there were many seagulls around, which was expected. However, we did notice that most of the birds around were pigeons. The seagulls seemed to be constrained to being closer to the river, while the pigeons were the ones found all around and inside the fort itself.

Since the birds were a lot more prominent than we expected, we paid attention to them during the visit and came up with ideas as to how we wanted them implemented into this simulation. They truly felt like a large part of the entire experience. Anywhere the team walked, we could always see some birds sitting within the cracks, within any holes, and even up in some of the higher spots within the rooms throughout the fort. The constant sounds of birds chirping were prominent the entire time.

A very important feature we want is to ensure that the birds had a logical spawn pattern. We do not want the user to see a bird pop up out of thin air, as that takes away from the realistic experience. To ensure this won't happen, we want to make the birds spawn in trees at random.

Another implementation we want to make sure of is that the birds have randomized flight and land patterns. If the birds spawn in the same tree, fly the same path, and land on the same points, it would take away from the experience for any user paying attention.

19.24. Grass and Trees

By introducing grass, trees, and foliage to our scene, we can provide a more lively experience as the user explores the grounds of the fort.

19.24.1 Grass

The grasses of the fort are simplistic in nature with minimal variations in consistency and type. Due to the consistent upkeep of the fort, the grass appears to be of a consistent short length both inside and outside the fort. Grass appears specifically in the fort's courtyard, moat, and outside vicinity. Additionally, the outer perimeter of the fort includes dirt paths as a result of people walking around the fort, especially in areas closer to the Matanzas river.

Figure 19.24.1.1: Dirt path located near the north end of the fort.

19.24.2 Trees

When our team visited the fort in person, we noticed that most of the surrounding trees were palm trees. These palm trees occurred as singular sources around the fort. Furthermore, the number of trees in the immediate surrounding area of the fort were few in number and sparsely populated the grounds, likely to ensure that views from the fort remain unobstructed. However, in the distance, there were some different varieties of trees including oak trees and camphor trees. From a distance, these outer trees appeared to be grouped much more tightly than those immediately surrounding the fort.

It should also be noted that no other foliage was visible from the grounds of the fort other than the leaves on the aforementioned trees. Figure 19.23.2.1 shown below provides a more detailed display of the environmental scenery.

Figure 19.24.2.1: Surrounding view of the environment north-west of the fort.

19.24.3 Scene Implementation

Unity provides several features that make populating a scene's environment with details including grass and trees a straightforward procedure. The terrain will consist of 3 main parts: ground textures, tree meshes, and grass textures.

20. Conclusion

Throughout the planning and prototype development of this project, there have been many challenges that we experienced. Some of these challenges were expected, but others appeared without warning. Each member had their own schedules to adhere to, not just from school but also personal and workplace responsibilities. In order to be successful with this project, managing our time became an important hurdle that we needed to overcome. To solve this problem, we got together and reviewed the consistent aspects of everyone's schedules, such as school and work. From here, we determined that the best times to meet were Mondays and Wednesdays after class in-person as well as 11am on Fridays over Discord. Of course, when forming the group, this was an obstacle we all had expected so we got together quickly in order to solve it as soon as possible.

One of the more unexpected challenges was the workload required for 3D modeling. In the initial phases of planning this project, we knew that we would

have to spend time on 3D modeling but we far underestimated the actual time that would be spent on this area of the project. Not only did 3D modeling pose a harsh learning curve, but the time spent to produce models once familiar with the process still vastly exceeded our expectations. Furthermore, finding time-saving assets that were both within our budget and consistent with our environment's art-style also proved to be a difficult challenge.

Now that our team is more familiar with the technologies involved in our project, we have been able to better adjust our plan of action for design and implementation. Each team member has been actively communicating and sharing the things that they have learned to excel our team as a whole. Ideas are discussed as a team and all tasks are designated in a strategic and structural manner.

References

[1] U.S. Department of the Interior. (2018, October 2). *The founding of Castillo de San Marcos*. National Parks Service. Retrieved March 17, 2022, from https://www.nps.gov/casa/learn/the-founding-of-castillo-de-san-marcos.htm

[2] U.S. Department of the Interior. (2022, April 20). *The British period (1763-1784)*. National Parks Service. Retrieved March 20, 2022, from https://www.nps.gov/casa/learn/historyculture/the-british-period.htm

[3] *Loop cut*. Loop Cut - Blender Manual. (2022, April 27). Retrieved April 27, 2022, from https://docs.blender.org/manual/en/latest/modeling/meshes/tools/loop.html

[4] *Extrude*. Extrude - Blender Manual. (n.d.). Retrieved April 27, 2022, from https://docs.blender.org/manual/en/2.80/modeling/meshes/editing/duplicating/extrude.html

[5] History.com Editors. (2009, November 9). *Stamp act*. History.com. Retrieved March 20, 2022, from https://www.history.com/topics/american-revolution/stamp-act#:~:text=Instead%20of%20levying%20a%20duty,in%20exchange%20for%20the%20stamp

[6] History.com Editors. (2009, October 29). *Revolutionary War*. History.com. Retrieved March 20, 2022, from https://www.history.com/topics/american-revolution/american-revolution-history

[7] *Our history*. Our History | St. Augustine, FL. (n.d.). Retrieved March 19, 2022, from https://www.citystaug.com/693/Our-History#:~:text=In%201861%2C%20the%20Civil%20War,St

[8] U.S. Department of the Interior. (2017, March 21). *Seminole incarceration*. National Parks Service. Retrieved March 20, 2022, from https://www.nps.gov/casa/learn/historyculture/seminole-incarceration.htm

[9] U.S. Department of the Interior. (2018, August 23). *The Civil War in Florida*. National Parks Service. Retrieved March 19, 2022, from https://www.nps.gov/casa/learn/historyculture/the-civil-war-in-florida.htm

[10] U.S. Department of the Interior. (2022, April 20). *Plains Indians*. National Parks Service. Retrieved March 19, 2022, from https://www.nps.gov/casa/learn/historyculture/plains-indians.htm

[11] Goode, L. (2019, January 5). *What is XR and How Do I Get It*? Wired. Retrieved February 21, 2022, from https://www.wired.com/story/what-is-xr/

[12] Bardi, J. (2022, March 29). *What Is Virtual Reality: Definitions, Devices, and Examples*. Marxent. Retrieved February 21, 2022, from https://www.marxentlabs.com/what-is-virtual-reality/

[13] *What is a game engine?* Full Scale. (2022, April 20). Retrieved February 23, 2022, from https://fullscale.io/blog/what-is-game-engine/

[14] Jordan, J. (2021, November 11). *What is Unity Game Engine & Why should you choose unity for game development?* NarraSoft. Retrieved February 23, 2022, from https://narrasoft.com/what-is-unity-game-engine-why-should-you-choose-unity-for-game-development/

[15] Wikimedia Foundation. (2022, April 7). *3D computer graphics*. Wikipedia. Retrieved March 1, 2022, from https://en.wikipedia.org/wiki/3D_computer_graphics

[16] Pratt, D. (2021, October 26). *2D rendering vs 3D rendering: The differences explained*. Cyber Risk. Retrieved March 1, 2022, from https://cyberisk.biz/2d-rendering-vs-3d-rendering/

[17] Wikimedia Foundation. (2022, January 13). *2d Computer Graphics*. Wikipedia. Retrieved March 1, 2022, from https://en.wikipedia.org/wiki/2D_computer_graphics

[18] *2D background in a 3D game! Unity beginner tutorial.* (2021, June 18). [Video]. Youtube. https://www.youtube.com/watch?v=Bv1foWFahM4

[19] Wikimedia Foundation. (2022, April 22). *Texture mapping*. Wikipedia. Retrieved March 20, 2022, from https://en.wikipedia.org/wiki/Texture_mapping

[20] *How To Create Your Own Texture Resources in Photoshop & Illustrator.* (2017, October 17). [Video]. Youtube. https://www.youtube.com/watch?v=VfogUHUWDtU

[21] Technologies, U. (n.d.). *Level of detail (LOD) for Meshes*. Unity. Retrieved March 20, 2022, from https://docs.unity3d.com/Manual/LevelOfDetail.html

[22] *How to make lods (level of detail) using Blender and unity game engine.* SUPER JUICE. (n.d.). Retrieved April 20, 2022, from http://superjuicegames.com/devlog/how-to-lod-using-blender-and-unity

[23] Technologies, U. (n.d.-a). *Unity - Manual: Baked lighting*. Unity. Retrieved April 20, 2022, https://docs.unity3d.com/2018.4/Documentation/Manual/LightMode-Baked.html

[24] *How to Actually optimize your game in Unity - Complete Game Optimization Guide*. (2020, June 19). [Video]. YouTube. https://www.youtube.com/watch?v=ysk7ATmIeOs

[25] Technologies, U. (n.d.-c). *Unity - Manual: Shadows*. Unity. Retrieved April 20, https://docs.unity3d.com/2017.4/Documentation/Manual/ShadowOverview.html

[26] LoProto, M. (2020, September 28). *What Is Anisotropic Filtering? PC Graphics Setting Explained. Cultured Vultures.* Retrieved April 20, https://culturedvultures.com/anisotropic-filtering-graphics-setting/

[27] Technologies, U. (n.d.-c). *Unity - Manual: Mipmaps.* Unity. Retrieved April 20, https://docs.unity3d.com/Manual/texture-mipmaps.html

[28] Unity Learn. (n.d.). *Optimizing your VR/AR Experiences*. Unity. Retrieved April 20, https://learn.unity.com/tutorial/optimizing-your-vr-ar-experiences#5fe2ad02edbc2a08ce472231

[29] *VR Optimization and Performance Tips for Unity*. (2021, March 24). [Video]. YouTube. https://www.youtube.com/watch?v=xqgt9W4Zrjg

[30] *Improve VR Performance in Unity! | EASY Performance Improvements (Beginner)*. (2021, December 3). [Video]. YouTube. https://www.youtube.com/watch?v=i4fPErsEJD4

[31] Oculus Developers. (n.d.). *Performance and Optimization.* Oculus. https://developer.oculus.com/documentation/unity/unity-perf/

[32] Technologies, U. (n.d.). *Unity XR Input*. Unity. https://docs.unity3d.com/Manual/xr_input.html

[33] *Lowpoly Sailing Ships* (2021, October 4). Sketchfab. https://sketchfab.com/3d-models/lowpoly-sailing-ships-babc260ab85044dfa96981725742bc09

[34] Technologies, U. (n.d.). *Rigidbody Component Reference*. Unity. https://docs.unity3d.com/Manual/class-Rigidbody.html

[35] Technologies, U. (n.d.). *Cloth*. Unity. https://docs.unity3d.com/Manual/class-Cloth