

Fort Hays State University

Hays, Kansas

An Interactive Virtual Campus Tour

A thesis submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

in

COMPUTER SCIENCE

by

Levi Smith

Advisor: Dr. Anas Hourani

Spring 2023

Content

Abstract.....	pg. 4
Intro.....	pg. 5
Problem Statement.....	pg. 6
Software Used.....	pg. 9
Proposed Project.....	pg. 13
Conclusion.....	pg. 23

Figures

Figure 1: <i>A screenshot of a room in Albertson Hall within the current virtual tour...</i>	pg. 6
Figure 2: <i>A screenshot of the current virtual tour's building selection screen.....</i>	pg. 8
Figure 3: <i>Polycam's Capture tab, with several thumbnails of scanned floors.....</i>	pg. 10
Figure 4: <i>Blender viewport, with the Art and Design building in edit mode.....</i>	pg. 11
Figure 5: <i>Godot's viewport, showing its Node/Scene structure on the left.....</i>	pg. 12
Figure 6: <i>North-facing screenshot of Albertson Hall.....</i>	pg. 19
Figure 7: <i>East-facing screenshot of Albertson Hall.....</i>	pg. 20
Figure 8: <i>South-facing screenshot of Albertson Hall.....</i>	pg. 20
Figure 9: <i>West-facing screenshot of Albertson Hall.....</i>	pg. 21
Figure 10: <i>Doorway icon in Albertson Hall.....</i>	pg. 22
Figure 11: <i>Closed infocard in Albertson Hall.....</i>	pg. 22
Figure 12: <i>Open infocard for Room 104 in Albertson Hall.....</i>	pg. 23

Abstract

One of the main things that would benefit prospective international students in terms of deciding whether to enroll in Fort Hays State University is to receive a better sense of what the campus is like. Local prospective students are able to take a tour of the campus to inspect the facilities and understand the feel of the buildings. There does happen to exist a Virtual Tour on the FHSU Website [2], though it is insufficient in providing the actual feel and scale of the buildings on campus. Thinking on this problem a bit further, one comes to the realization that not only prospective international students would benefit from introducing an improvement on the virtual tour option currently provided; prospective students within the country without the resources to support an in-person tour would also benefit. Implemented properly, a solution could benefit current FHSU students as well, by serving as a map of the buildings on campus that are easier to interpret and remember than current options of flat, two-dimensional floor layouts. The proposed solution is implementing a system in which a user-controlled agent walks through a 3D representation of the interiors of the buildings on campus. This system allows for traversal between floors and into and out of notable rooms inside of the buildings, along with info cards on standard classrooms denoting room capacity and schedule.

1. Introduction

As stated, a useful tool to help a prospective student determine whether they would like to enroll in FHSU is a tour, as evidenced by the in-person tours, which some sources determine results in around a 60% matriculation rate [4]. In addition, many campuses are adding a virtual tour option, including FHSU. Many of these can be seen at websites like youvisit.com [5], which compile and display virtual campus tours. There are some downsides to the current FHSU virtual tour option, however, which will be elaborated upon in the problem statement section. An improvement upon the current system would increase prospective student's matriculation rate, as evidenced by research from BDHP [4]. This project will attempt to improve the tactile sense of scale and spatial cohesion of each building on campus. It will attempt to improve these metrics by including a controllable agent, which is controlled by the user to explore the buildings "off-rails," as many of the colleges and universities providing a virtual tour option stick to an approach similar to that of Google Street View. This is an implementation in which a set of 360 degree photos are taken at set intervals, and the user is allowed to move between those discrete points, looking around in each given section. This implementation would provide a more continuous exploration of the buildings on campus, only segmented by the doorways and stairs inside of the buildings.

This paper will reiterate the problem statement, expanding upon the issues and benefits of the current option. Then it will outline the software options for each step of the project, including scanning the building interiors, modifying the meshes generated by the scanning software, and implementation of the user-controlled agent in the 3D

space. Finally, it will describe the implementation and methodology of a proposed solution to the shortcomings described in the problem statement, with images of the results.

2. Problem Statement

Here there will be further extrapolation upon the current option for getting a sense of campus, i.e. the virtual tour, and the proposed option to be implemented. The virtual tour simply presents photos of people inside of the buildings alongside a general description of the building's purpose, as shown in figure 1.



Figure 1: A screenshot of a room in Albertson Hall within the current virtual tour

This does not provide the tactile sense of scale and space that a three dimensional tour should provide. One of the main goals of this project is to provide an accurate sense of

the layout of each building, in other words, the project should impose a good sense of local spatial cohesion. In addition, the info cards will give a better sense of the capabilities of each room of each building on campus. This will help it to solidify the image of the campus in virtual students' minds, but also give a good reference to those who are on campus that would like to know the arrangement of rooms in buildings on campus, along with useful information for each room. In the case of the virtual tour, there is no local spatial cohesion between the images. There could be several photos in a row of the same room, followed by an image of a room on a different floor, and on the complete opposite side of the building. This does not give a good sense of the layout of the buildings being presented. One of the benefits of the current design of the virtual tour, however, is that it brings life to the rooms which are photographed. This is mainly due to the improved camera quality, and the photos staged with people. The sense of life is a lacking aspect of this project due to limitations including diminishing returns on scanning time versus texture mesh quality, and the intentional lack of peoples' presence in the scans. The current virtual tour also gives an excellent sense of global spatial cohesion in the scope of the campus and surrounding area as shown in figure 2.



Figure 2: A screenshot of the current virtual tour's building selection screen

The building selection screen does a good job of allowing a user to determine the location and relative distance of buildings to one another. Finally, the advantages of the current virtual tour in regards to storage space and resource requirements are relatively stark. To be clear, the absolute effect of storage space requirements is not an issue. This is due to the project and virtual tour being hosted by FHSU's servers, which are more than adequate for both options. However, the client-side resource requirements are greater in the case of this project than the current virtual tour. Rendering a view of a 3D model from the perspective of the agent, taking input, and determining collision with the floors and walls results in a greater utilization of CPU and GPU than the static slideshow presentation of the virtual tour. This may cause issues for individuals with inadequate laptop or desktop setups. The goal of this project is not to replace the

current virtual tour, but rather to introduce an option to offset the shortcomings of the current option. All in all, combining the adequate parts of the current virtual tour option and the improvements of the suggested solution could result in an application that benefits virtual students, international students, prospective students, and currently enrolled on-campus students more than the current virtual tour can alone.

3. Utilized Software

To create and implement this system, including the 3D modeling, the engine for rendering the perspective of the agent, and exporting it to a browser-runnable executable, there are multiple options. The pros and cons of which will be explained in this section.

Firstly, for 3D modeling, there are a few options available. Three options that stuck out immediately were applications available on mobile devices, including smartphone and tablet. These options were Canvas, Magicplan, and Polycam. These are all software that will take input from the device's camera and translate them into a 3D mesh (from now on referred to as "scanning"). Canvas was the first option tested. This application would take scans from a solitary point, and generate a mesh centered on said point. The meshes that Canvas produces would have to be stitched together in a 3D modeling software, the options of which will be outlined later in this section. Canvas also had a pricing plan per-square-foot, which when considering the size of the campus would add up quickly. This discarded it as an option soon after discovering the pricing. Magicplan had a monthly pricing scheme, which would have been more efficient for this project. Magicplan, however, only produced floor plans at the time of this

project's inception. This could possibly work, extruding the walls from the flat floor plan and applying proper textures programmatically, but would have been less intuitive and less indicative of how the campus feels, which is a main goal of this project. Concluding the section on 3D scanning software is Polycam. This option has the 3D models of Canvas with a monthly pricing plan of Magicplan, and has options for LiDAR and photogrammetry, the differences of which will be explained further on. This seemed like the best option, and is the software that was used for the scanning portion of the project. Polycam's thumbnail preview is shown in figure 3.

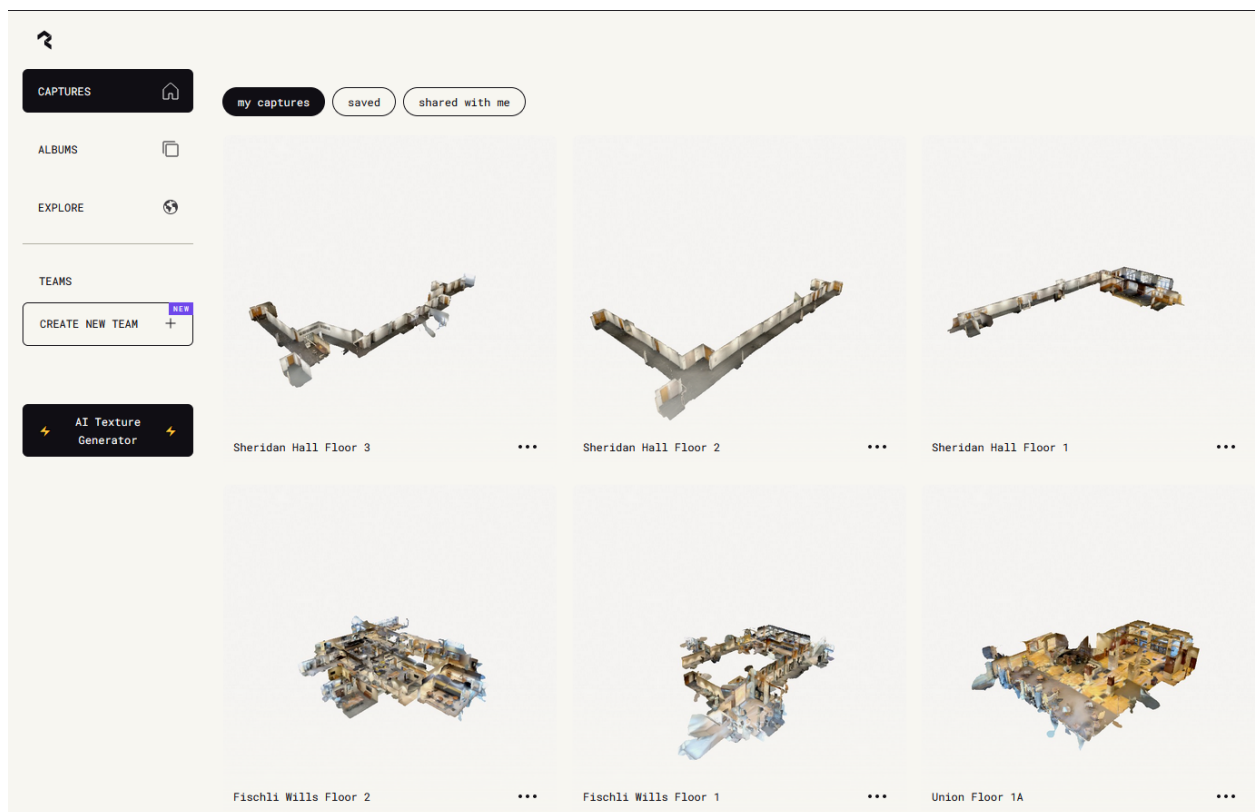


Figure 3: Polycam's Capture tab, with several thumbnails of scanned floors

Next thing discussed will be the 3D modeling software used. Sometimes, the scans generated by Polycam would require touch-ups. This includes fixing certain scanning errors, detailed later, or blotting out areas of the scan that should not show up in the final project. When the in-application functions did not happen to fix these situations, a third-party 3D modeling software was required. In the few times that these were necessary, the open-source 3D modeling software Blender was used. Its viewport is shown in figure 4. It has functionality including mesh manipulation and mesh texture editing.

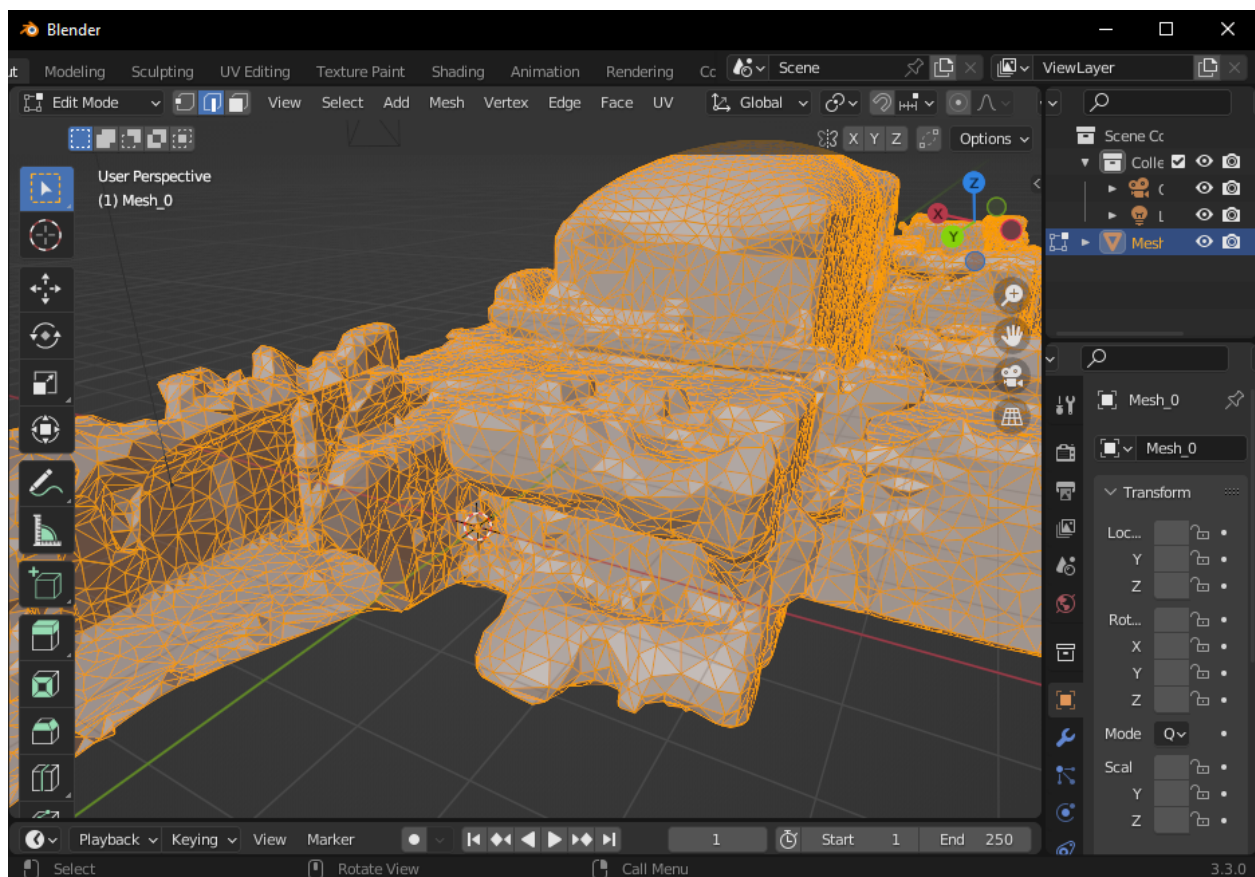


Figure 4: Blender viewport, with the Art and Design building in edit mode

Then, deciding upon which environment to build the controllable agent and infocards, the choice came down to the Unity engine and the Godot engine. These are fairly similar, though the exclusive benefits of Godot did make the difference in determining which one to use. Firstly, Unity engine is a privately-owned piece of software, and their license policy reflects that. Unity's licensing would not provide the level of freedom in creating this project that Godot, an open-source software with the very unrestrictive MIT license, can do just as well. In addition, Godot's Scene/Node project organization scheme works better for this type of project, as opposed to Unity's Object/Property organization. This is shown in figure 5.

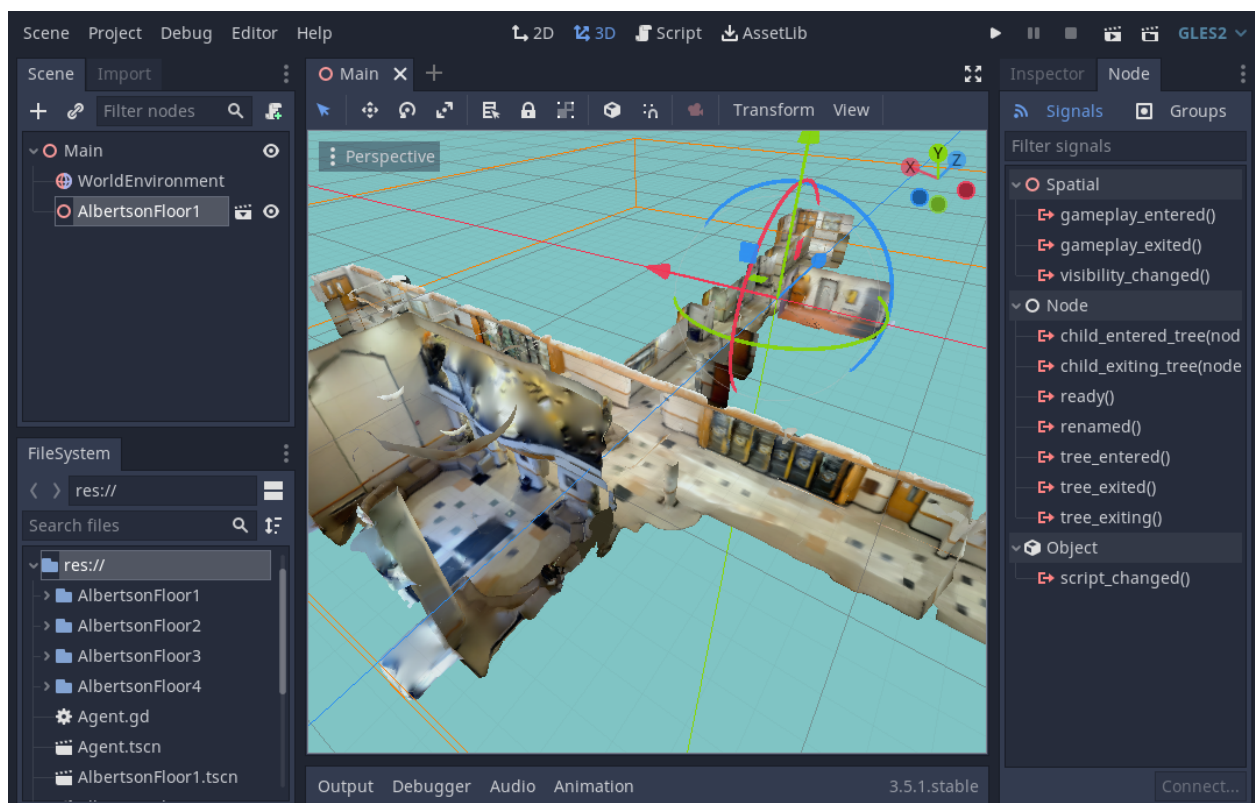


Figure 5: Godot's viewport, showing its Node/Scene structure on the left

Finally, Godot has simple project build and export to HTML5 which is crucial to integrating this project with the FHSU website.

With all other aspects decided upon, the choice between LiDAR and photogrammetry is the only one that remains. Photogrammetry is the process of using photographs to measure the distance between objects. Polycam then uses these sets of measurements, taken by sweeping the device's camera across and around the subject of the scan, to generate a 3D mesh of the object. LiDAR is slightly different. Instead of using photographs, it uses data generated by a LiDAR camera to determine the measurement of objects. LiDAR is most easily explained by this: the difference between LiDAR and radar is, where radar uses reflections of sound to determine distance, LiDAR uses infrared light. This allows the device to save on processing power to calculate distance, as it's a more direct measurement than the calculation of distance between the difference of two photos. The resolution of the LiDAR camera used in this project is less than that of the device's photographic camera, which results in less precisely defined shapes, but Polycam still uses the regular camera to generate textures for the 3D model, which balances out the lower quality shapes. In comparisons of the implementation of photogrammetry and LiDAR scanning in Polycam, the experience of using LiDAR is easier and faster. The dealbreaker between the two choices, though, is the fact that, when scanning with LiDAR, Polycam keeps track of which part of the mesh is already generated, removing uncertainty as to what will be processed and result in the final model. This capability can be lent to the freed up processing power by removing the middleman of photogrammetry distance calculations. To sum up, LiDAR

scanning through Polycam was chosen to be used in the creation of the 3D models for this project.

4. Proposed Project

A broad overview of the methodology for scanning and importing the models into the engine are as follows. Firstly, the building to be scanned was determined based on perceived ease of scanning, leaving the hardest buildings for last. The buildings were scanned floor-by-floor, from the ground floor up, then scanning a basement if one was present. After that, important and distinct rooms were identified and scanned. Then, the scans were renamed for better organization, following the pattern of: the building's name, followed by floor number in the case of floors, or the building ID (RH for Rarick Hall, TH for Tomanek Hall, etc.), followed by the room number in the case of rooms. The scans were then uploaded to the cloud from polycam and exported as a GLTF file on a desktop, to be imported into Godot. The user-controlled agent was placed in a starting point on the ground floor model of the building. Following that, floor transitions were placed in the building's model, then the room transitions, then the info cards. Lastly in Godot, collision was added to each mesh so that the agent would not pass through the walls of the floor or room model. The project was then built and exported as an HTML5 executable.

As previously stated, the tools used for this project included Polycam for creating the meshes of the interiors, Blender for modification of the meshes where necessary, and Godot Engine to import the meshes, create an agent, calculate collision with walls, and export the project to HTML5 for implementation on the FHSU website. Polycam's

usage begins by opening the application on the device that will scan the building, then pressing a plus-shaped icon in the top right of the UI to begin a new scan. The choice between LiDAR, photogrammetry, and an experimental “room” scan which was not used for this project appears. Once LiDAR has been selected, Polycam starts scanning and aligning a wireframe mesh to the features of the subject being scanned. This is not saved into any sort of storage until the record button is pressed. The process of scanning a building is a careful balance of detail and expediency. The more time spent on scanning an area, the better the final scan, however, the more time spent on scanning an area also corresponds with bloated file sizes and an issue called “drifting,” in which the vertices scanned by Polycam become misaligned with the actual details of the real world. In cases where the building loops in on itself, this can result in unacceptably incoherent meshes, ruining the efforts of a 30-minute scan. Drifting can cause floors to have elevation increases or decreases where there are none, or identical walls repeated, separated by a foot or more.

Once finished with the scan, the end recording button is pressed, and the file containing the 3D mesh is inspected for missing sections or drift. If the mesh is acceptable, the file is renamed and saved to the cloud. To finish the portion of the project that utilized Polycam, the scan is then downloaded from the Polycam website to the desktop that has Blender and Godot installed. Blender’s capabilities extend far past its necessary features for this project, but what was used were the simple mesh manipulation functionalities. These included vertex manipulation and texture painting. These were not used often, as many issues were too complex to fix with Blender, or could have been more easily solved upstream with Polycam. Godot engine is an

open-source engine whose structure is based on nodes and scenes. Scenes are composed of nodes, and scenes can act as nodes in other scenes. Nodes are individual elements in a scene that provide some functionality such as collision, displaying meshes, or affecting the lighting of the whole scene, as a few examples. Nodes can also have scripts attached to them, which programmatically provide extra functions. The scripting is based on the Observer design pattern, in which parts of code send signals that other pieces of code can receive and do work on [1]. The scripts are written in gdscript, a language with similar syntax to python, but with faster compile times.

The process of using Godot was as follows. Firstly, a new project was created, each project representing one building on campus. Then, consistent files between projects would be imported. These were files such as doorway assets and scripts, info card assets and scripts, and the player-controlled agent's script. Following that, the scanned building's meshes were exported as a .glTF file type, which is the type that Godot uses most easily. The meshes were then renamed and imported. After that, the meshes were put into a scene, and a collision node was applied to the mesh. This was done using Godot's "create trimesh collision" function, which takes the vertices that define the shape of the mesh, and uses them for collision in addition to displaying a texture. This resulted in a very accurate collision with the geometry of the mesh, very quickly. However, this came at two downsides. One, resource utilization went up, as more collision checks had to be made per frame of movement. Secondly, this required more accurate scans in Polycam, and desync became even more of a problem. These are reasonable tradeoffs, as the collision quality without trimesh collision was not of acceptable quality. Next, the actor and doorways were placed into the scene. If the floor

being worked on was the ground floor, the actor was placed at the most obvious entrance to act as the default spawn point when loading the building. The doorway scene contains a variable that determines which room to transition to when clicked, so that was placed in front of each door and then given the proper name.

This project, of course, had a projected timeline that differed from the actual timeline of work. This project had approximately 14 weeks to complete in the spring 2023 semester. The projected timeline was as follows: Spend 5 weeks scanning buildings, followed by 5 weeks of implementing the scans and developing functional mechanics of the application, such as doors and infocards; the remaining weeks would be spent writing this paper and creating a presentation for the project. However, the timeline of work ended up following a form that combined the categories more than not. The scanning started in isolation for around a week, as Rarick Hall was scanned as a proof of concept. Following that, the main systems, doors and infocards, were implemented in a rudimentary form, which took three days. Scans were then continued as methodological improvements in both the scanning process and functional development of the application occurred. Buildings more difficult to scan than Rarick cropped up, however. The scanning process ended up taking approximately eleven of the fourteen weeks. The writing process began around halfway through the semester, as to give more leeway with the amount of time allowed scanning. The obstacles that occurred that diverted these two timelines are listed in the subsequent section.

Firstly, Rarick Hall was one of the easiest buildings to scan in many regards, in retrospect. The layout of Rarick gave an easy, gridlike scan, with its own problems that did not persist in other buildings. Rarick lacks one of the more obstructive issues found

in most other buildings, though. This issue being windows. As LiDAR uses infrared light to determine the physical distance of an object, windows allowing those beams to pass through cause a large gap in the mesh of the scanned floor. Oftentimes, these gaps would not be stitched together in the processing of the scan, and a large, collisionless, textureless hole would appear in the final model. Methodologically, this could be attempted to be solved upstream by scanning the area in a particular way that would lead Polycam to determine that a window with a texture of the view would result in the best final scan, but this technique could not always be employed, particularly in the case of large windows. This was the second most costly issue on the scanning end of things. A problem that recurred through nearly every building on campus was that of the previously mentioned desync. These were the two most time-consuming obstacles for this project, as they are issues that appear near the end of the most time-consuming necessary aspect of the project, the scan. When these issues did crop up, the most direct solution was to start the scan again, wasting an entire day's scan. A more systemic obstacle was that of the availability of academic campus buildings. They are only open on weekdays from 6:00 a.m. to 6:00 p.m., generally. This proved an issue, as the available times to scan were a smaller window, generally 4:00 p.m. to 6 p.m.; scans had to be done around students moving from class to class, and on a time scale that allows for - at max - 3 buildings a week. There were few issues and obstacles downstream, and those were relatively easy to deal with, as they could be solved without a schedule restriction, and with many online resources.

In operation of the application, the program is booted up, and the agent is controlled with the arrow keys, or W, A, S, and D. In this control scheme, up arrow or W

and down arrow or S applies a constant movement vector in the forward or backward direction relative to the agent, respectively. The remaining keys, left arrow or A and right arrow or D, apply a constant rotation about the Z axis of the agent while the key is held. In order to interact with the doorways or info cards, the cursor is employed, and a single left click within the icon's bounds activates its function. The transformation of the agent's rotation is mainly handled by modifications to Godot's built in rotation property. The forwards-and-backwards movement was defined by a velocity vector that was then called in a function called `move_and_slide`, which applies the velocity of a kinematic body, in this case the agent, to the rest of the collision geometry of the scene. Some coordination between the camera and collision of the agent had to be coordinated by a parent node of both, which the transformations were applied to.

All in all, what was able to be accomplished was a spatially cohesive map of each individual academic and facility building on campus. It serves as a reference for new students and as a tour option for virtual or international students. When implemented, the current virtual tour option would provide a choice between the slideshow currently available and the interactive tour, see Figure 2. Examples below of a hallway in Albertson Hall facing north as seen in figure 6, east as seen in figure 7, west as seen in figure 8, and south, as seen in figure 9.



Figure 6: North-facing screenshot of Albertson Hall



Figure 7: East-facing screenshot of Albertson Hall



Figure 8: South-facing screenshot of Albertson Hall



Figure 9: West-facing screenshot of Albertson Hall

Further examples showing the doorway interactive icon as seen in figure 10, and the info card interactive icon, collapsed and expanded as seen in figures 11 and 12, respectively.



Figure 10: Doorway icon in Albertson Hall

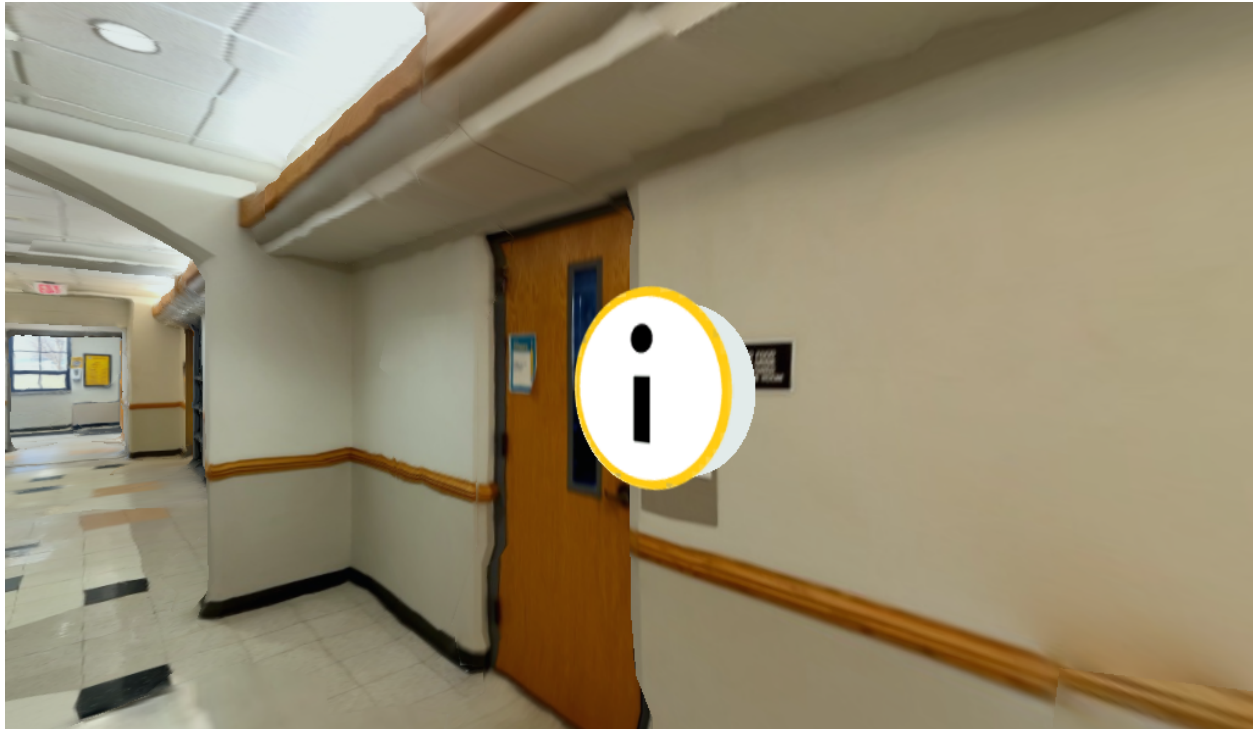


Figure 11: Closed infocard in Albertson Hall



Figure 12: Open infocard for Room 104 in Albertson Hall

5. Conclusion

Fort Hays State University has a large online and international community, all of them joining programs and courses online. Obviously, the online student experience does not include the physical Fort Hays State University Campus. Everyone would benefit from the option to visit the campus virtually, and this project would provide that option. In addition to giving online students a sense of what the campus feels like, new students would be able to use this application as a map, helping to find their classrooms or professors' offices without wasting time looking for them in person. This project will solve these issues and help students to visit classes and future and new students to navigate campus.

There are some ways to expand upon the functionality of this project in the future. For example, it may be beneficial to connect the info cards to a database containing availability schedules for each room, to help determine when rooms are taken. This would take some time, and is outside of the scope of this project for now.

References

- [1] Nystrom, Robert. "Observer." *Observer · Design Patterns Revisited · Game Programming Patterns*, 2009,
<https://gameprogrammingpatterns.com/observer.html>.

- [2] FHSU Virtual Tour, <https://fhsu.university-tour.com/virtual-map#>, April 18, 2023

- [3] Godot Docs - 3.6 Branch - Godot Engine (3.6) documentation in English,
<https://docs.godotengine.org/en/3.6/>, April 20, 2023

- [4] Research on the Prospective College Student Campus Visit and Its Impact on
Matriculation,
<https://www.bhdp.com/insights/research-prospective-college-student-campus-visit-and-its-impact-matriculation>, April 20, 2023

- [5] YouVisit, <https://www.youvisit.com/>, April 20, 2023