Hunter Mast

Dr. Bodenheimer

Augmented Reality

May 2nd, 2024

Adjustable Housing Project

Project images.

Video:

Detailed description of the user interface/techniques.

Emphasis on special design considerations made or required.

In many industries, planning projects beforehand and having an idea of the outcome is mandatory for building a successful product. Project planning can come in many different forms. One such form is of architectural blueprints. Industries such as construction or engineering use blueprints to know what the finished product will look like and what is needed to build such a product. My father works in an RV factory and uses many blueprints to know how to build each RV. A downside is that while we can get an idea of what it will look like, many of these do not include appliances and furniture. With my project, my goal was to create a blueprint creator using image targets in Vuforia to quickly adjust to changes and view what a finished product could look like with or without utilities.

I designed my project focusing on two aspects. One was making it so the house is as adjustable as I could possibly make it. The other was making tracking the main section of augmented reality I focused on and experimented with. I wanted to include four main features in the house. Those being the main outside corners, interior walls, openings such as doors or windows, and finally, furniture and appliances. To do this, I had to create targets to represent each of these aspects. For outside corners, I included unique 10 targets and interior walls I had 9 targets. For openings, I included 3 door targets and 4 window targets. Furniture included a single target for an oven, refrigerator, sofa, and bed. Of course, these could be expanded on to include a bathroom, more kitchen appliances, stairs, etc. To focus more on the overall idea, I just wanted to keep these small amount of features that I considered the basis of this project.

One issue I had right aways was with my targets themselves. Using Vuforia, I could not use multiple of the same images at the same time. My original plan was to use multiple of the same image targets for the same function. During my minimal viable design, I had not fully figured out how I was going to solve this issue. However, I eventually added a specific amount of images for targets to represent the same feature. For example, my project required that I use 10 different images of corners to represent all of my outside corners. I had experimented with QR codes and other ways of handling these targets like numbers, but only different images seemed to work. A few of my images also were too close together to where they conflicted and would show multiple target’s objects onto a single target. Testing through these, I was able to find an amount of targets that did not conflict and allowed for a large number of tests.

Another design aspect I implemented was the addition of colors. This was somewhat simple compared to tracking the images as a canvas object was just added to Unity and I had to run a script on it to change each model’s material color. Experimenting with these buttons and panels also had me rethink my scanning method from the minimal viable design. Originally, I had scanned by tapping the screen to produce each house model to be placed on the house target. However, accidental taps would end with a bad model. To minimize this possibility, I added a scanning button that would trigger each scan of the targets to produce a model. The design of the buttons is very basic as I wanted to focus on usability, but there are a multitude of ways to design the buttons in a more elegant manner.

Going off this, the models also provided a large amount of issues when tracking. Each target has a prefab model assigned to them as a child object. To create the house, I needed to combine all of these models into a single game object on top of the house target. When scanning the house target, all of the models combined along with the calculated walls would appear, showing a complete house. An issue I ran into with this though was rotation and position of these targets. Depending on each tracked target’s location, I had to translate it’s scanned position relative to the house target. Some corners needed rotated in different ways or walls needed positions known to calculate the midpoint of the corners to place a scaled wall model to connect the corners. There were lots of debugging involving these and trying to create a model that looked nice.

Something that I had thought about including, but decided not to was fixed positions. In my current environment, the models can be turned any which way and a wall would be placed between them. This causes a lot of visual errors to where the walls may not connect right with the targets. To solve this, I thought about rounding each of the coordinates to a position relative to another target. This would hopefully align each target so that walls would be placed in a cleaner way. However, this comes with it’s own set of issues. Fixing each position takes away from the flexibility the project was designed for. Some positions would have to round to a completely different number to match the other targets and could drastically change the outlook of the blueprint. For simplicity, I left this out and focused on having as clean of a model as I could without fixing each position.

Checking if each target was in position also came with it’s own issues. What I had implemented was a system where each target would check on it’s X-axis and Z-axis for any other targets above and below within a threshold of 0.5. This would allow for any targets that are not in line to not create a bad model and would result in an error. This did result in many errors to where some checks were not working properly and could sometimes be hard to debug due to the amount of checks that need to be made for each target.

The biggest change from the minimal viable design is how I scanned targets. Above, I mentioned how I included a button. With my final, I also changed how each target was analyzed. Before, I had put each target into a list of available targets. When adding more targets, I needed flexibility to know which targets are available and to change on the fly. For this, I added an array called “allTargets” that would contain all of the possible targets. When scanning, we would sort through this list of possible targets and add then to the correct lists depending on tags. This way, we could easily change which targets are being used without requiring the Unity project to be messed with. Much of my code relied on these arrays, so changing them to lists meant a lot of reworking. Along with this, I added the addition of inner targets to the code. For my minimal viable design, I did not consider the possibility of multiple opening targets next to each other. I realized this when attempting to implement inner targets. To fix this, my code had to be changed from connecting the “openings to the corners” to “searching between each corner to connect each target between together”. This required many changes to my “OpeningWalls()” function. Overall, it did provide for much more flexibility and new ways to design the blueprint.

With inner walls, we also had to include all of the possibilities of their positions. Most of my targets had to just consider being lined up within the corners. For inner targets, there was the possibility of being inside the corners to connect all inner walls. For this, I create a function to connect the inner walls depending on any other targets around. If it was between 2 targets, then it would be a normal connecting wall. If it was between 3 targets, it would become a “T” shape with rotations matching the other target’s positions. For 4 targets, it would become a “plus” shape connecting targets on all 4 sides. Inner walls created the most difficult part of my project as it required me to rework most of my code to fit them in.

For furniture, this section was fairly easy. It just required me to know if the furniture target was inside of the main corners. If so, then it could be placed. Checking between targets did however become an issue when I included a floor. TALK ABOUT FLOOR

Something that I also experimented a bit with, but did not end up including was surface detection. An idea I had was to click a button to display a model on top of a detected surface to place the “houseTarget” model. I realized when creating this, it requires a lot of computation. When trying to create this, TALK ABOUT HOW TO DETECT SURFACE

Development processes I employed included tracking features and accessibility options. TALK ABOUT DEVELOPMENT

What is novel about my approach is the flexibility and simplicity my project includes. There are lots of ways I would love to improve this project. Examples being idea like multiple floors, including a roof, furniture design, more accurate scanning, etc. Focusing on the simplicity though, I was able to create a basic 2D blueprint to quickly gather an idea of what an environment would look like. From what I have gather from my father, any blueprint software comes bloated with many features that does not fully give an idea of the models. It can take multiple hours for someone to create a 3D model of an environment to gather an idea of the project. Quickly being able to see the project saves multiple hours of time. It also allows the flexibility to change the project right away. Moving a target and rescanning, again, saves hours of time compared to recreating a new model.

There are some limitations my project does bring to the table though. The largest limitations came from scanning targets, Vuforia’s tracking algorithms, and lag brought upon with gathering target information. TALK ABOUT LIMITATIONS

I used Vuforia and some models found online. Sources I used were lots of Vuforia’s user manual. I looked into a paper or two with ideas similar to mine, but they did not use any algorithms or code similar to mine. TALK ABOUT SOURCES