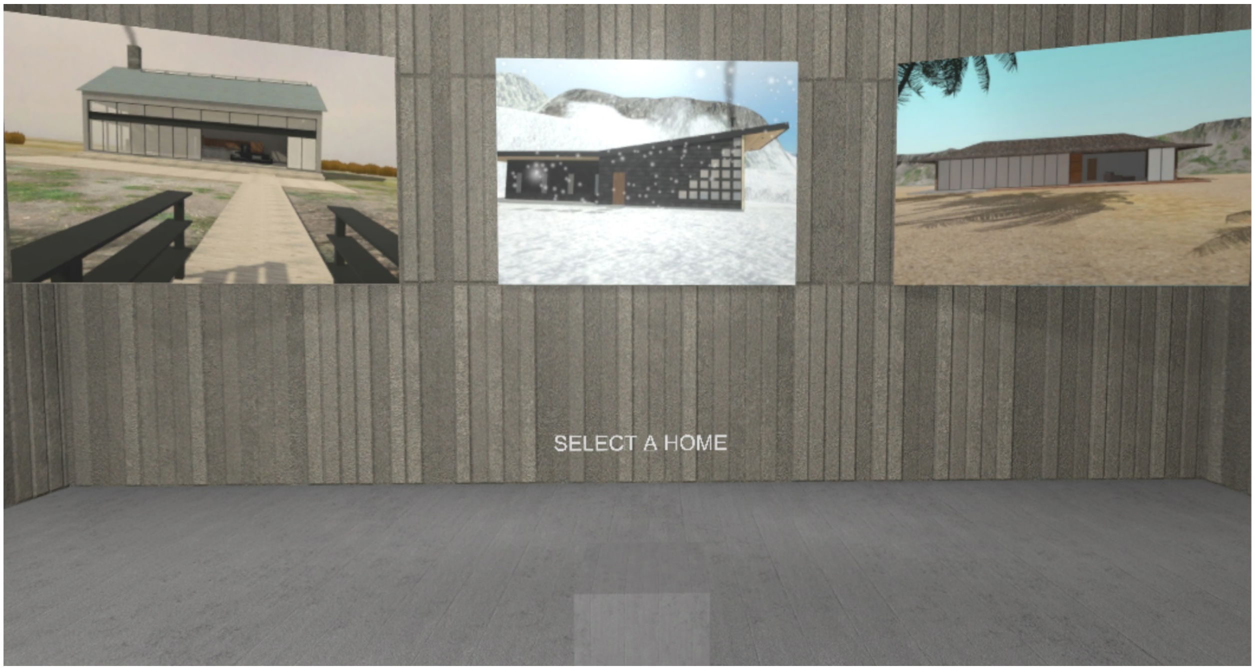
Virtual Real Estate

Description:

For my final project, I wanted to create something that merged my new fledgling hobby of *MR* engineering with my love for Architecture and design - which just happens to be my profession as well. In doing so, I paid special attention to making something of an application that could be more than just a *game*. Something that could be *more* than just a way to show off newly learned skills, something that could serve as a prototype for - or at the very least, a proof of concept for -a market sector that could benefit greatly from the MR revolution. I decided to design and build a *Virtual Real Estate Browser.*

The first thing that came to mind when I decided on my project was Redfin. I might be the exception, but I obsessively look at Redfin - at least 2 to 3 times a day - because I like to know the real estate market and I enjoy looking at homes. Though it may be stressful, *excitement* first comes to mind when thinking of an emotion for home buying. For starters, I knew I wanted a *HUB* like space where one can see an overview of homes on the market, and select one from there to take a virtual tour, much like a landing page in traditional terms.



HUB

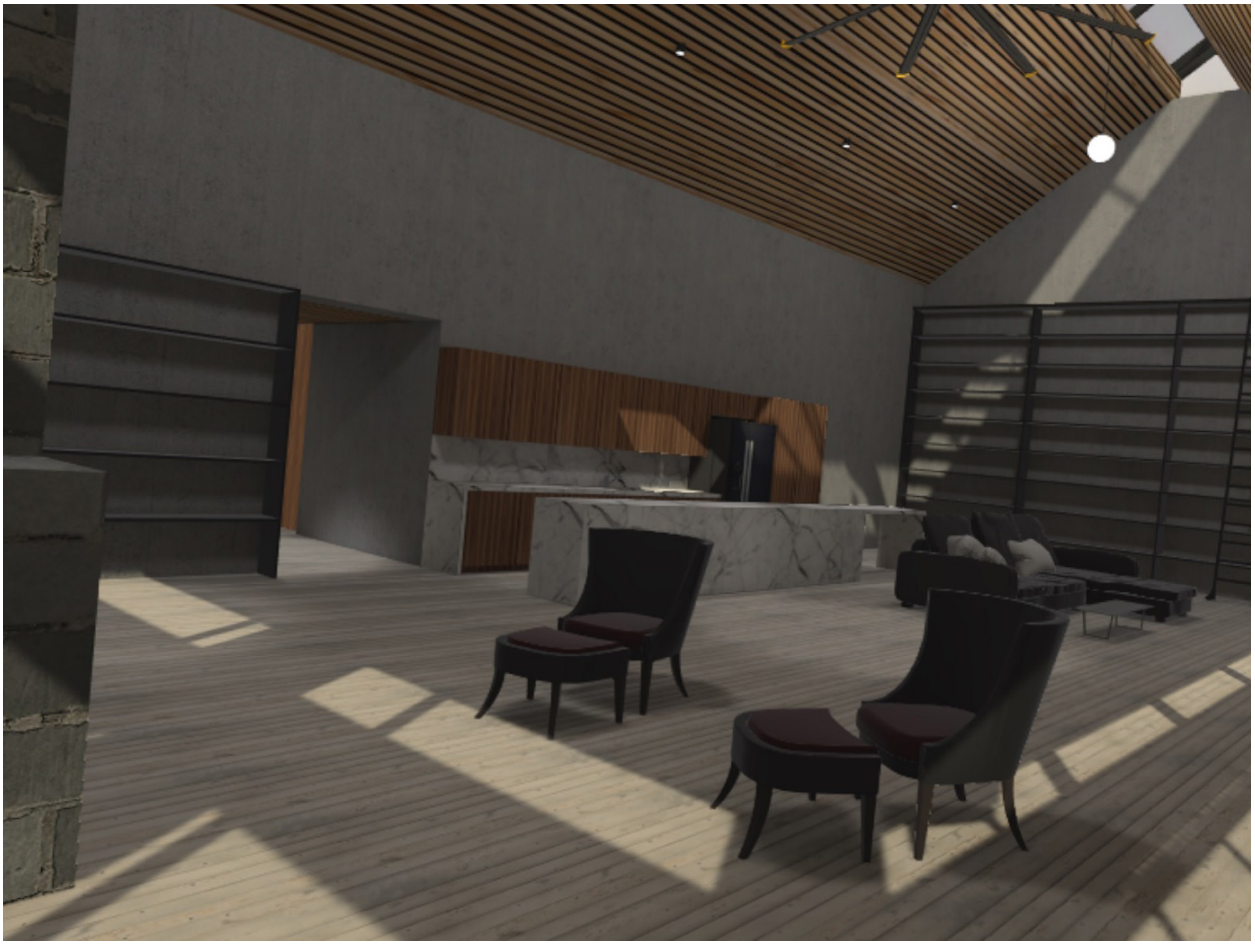
From there, one can take a deep dive into any house on the “market”. Since “gamification” wasn’t where I wanted to take this, I knew I had to lean heavily on 3d modeling and simple and effective UI - including voice commands.

Features And Dependencies:

1. 3D Models

a. This is crucial. For the sake of the PC environment, I decided I wanted to 3d model some custom homes to really utilize the full power of a the machine.

i.

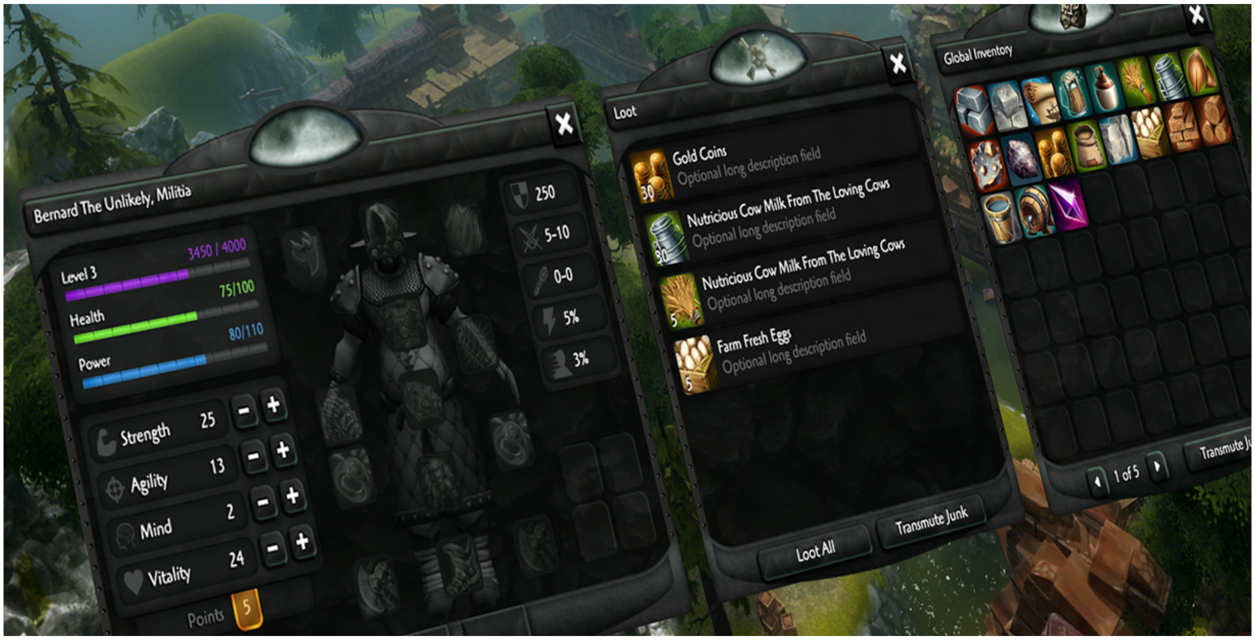


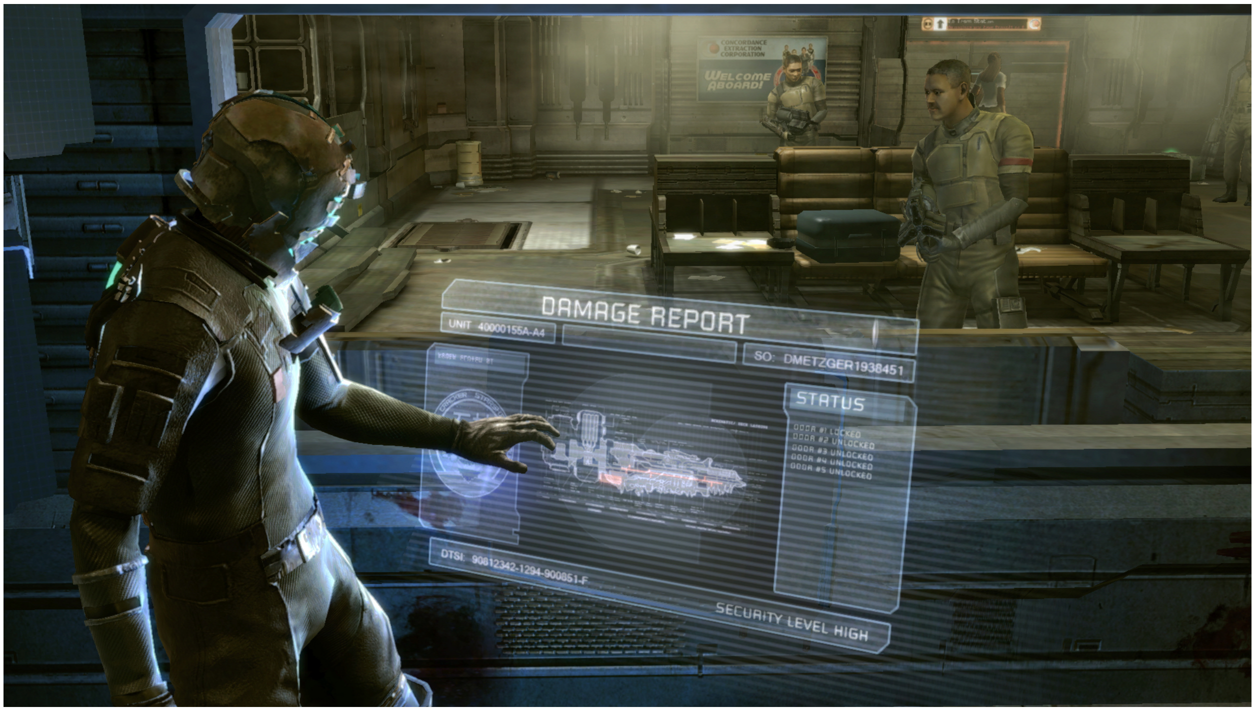
b. What is interesting about this is I can easily swap out a custom 3d model with still equirectangular or cube-map images of existing houses. With something as simple as a boolean check for an object in the scene indicating whether a user is in a model or image based tour, the navigation / locomotion handling can change.

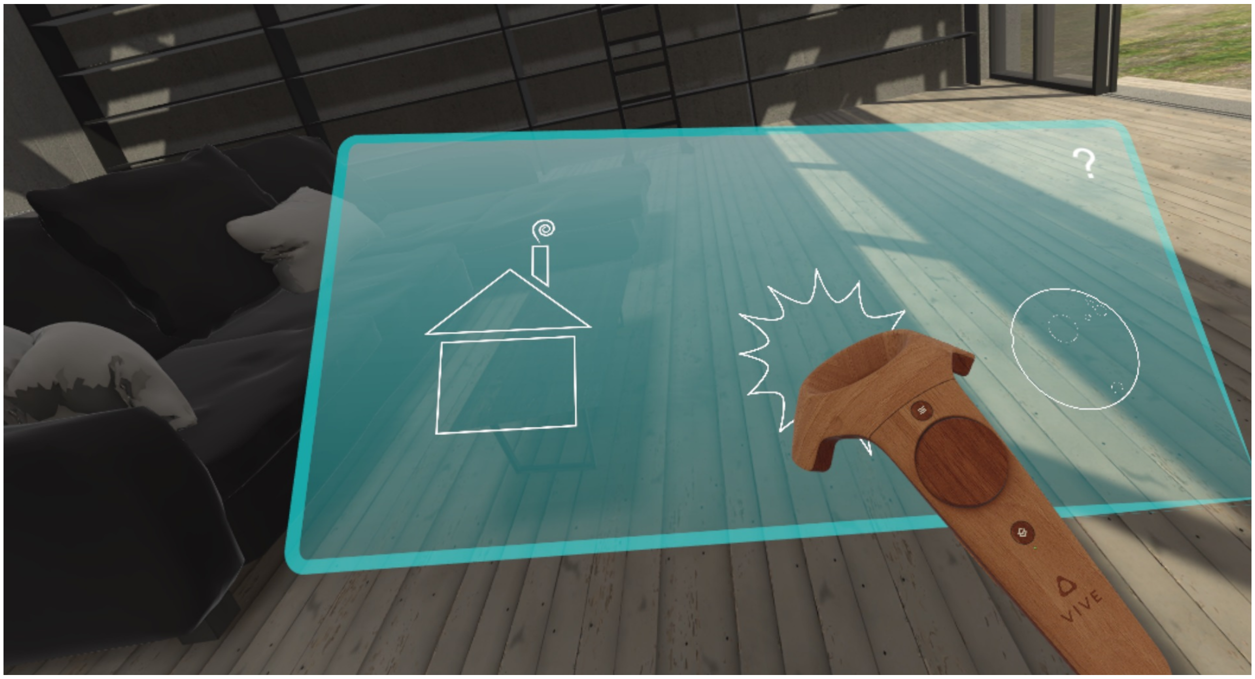
2. Diegetic UI

a. To keep you in the experience, I wanted to develop a simple Diegetic UI. In fact, because I wanted to have as minimal UI as possible, everything one can do in the 'menu', one can do as well with simple, one-word voice commands.

b. See below for (2) Diegetic UI inspirations then the final product of my UI.


Bernard The Unlikely, Militia
Level 3
3450 / 4000
Health
75/100
Power
80/110
Strength 25
Agility 13
Mind 2
Vitality 24
Points 5
250
5-10
0-0
5%
3%
Loot
Gold Coins
Optional long description field
Nutricious Cow Milk From The Loving Cows
Optional long description field
Nutricious Cow Milk From The Loving Cows
Optional long description field
Farm Fresh Eggs
Optional long description field
Loot All
Transmute Junk
Global Inventory
1 of 5
Transmute Ju





3. Application Loop

a. Interaction with the “market” content in the *HUB*. Selecting one house will *setActive(true)* a ‘billboard’ of preview photos for the user to look at.
b. User gets to select which house to tour by pointing and clicking on the appropriate ‘market’ image. Once clicked, the scale model for that image appears in front of the User. The user can then grab the scale model to tour.
c. The User enacts “the Lab” type warping to a specific scene (bringing something close to the users head to initiate a scene change. I just love this feature)
d. Speech Recognition for UI functions. User can use simple speech to initiate UI items like changing from night to day or warping back to the HUB.
e. Locomotion mechanic - using base steamVR teleport. Why try to make something that someone has already made very very well?
f. User can infinitely cycle through the (3) pre-made homes and he HUB.

4. Special Items

a. A script on the “Billboard large” objects utilizes a *for loop* to assign billboard textures in runtime. This would help with scalability if I needed to add an unknown amount of small preview “billboards”.
b. I spent some time on what to do if the user grabs the scale model and lets it go. I ended on having the model translate back to its starting vector in a floating manner. To prevent this model from doing anything weird or clip though the pedestal it spins above, I made a separate function to trigger if the model is let go *BELOW* the pedestal. In this case, the model moves out and up until it reaches its start y value, then it moves back towards its beginning vector.
c. Implemented a simple script with windows.speech to use keywords for speech recognition.
d. Mixed lighting baked into every scene.
e. Implemented spacial audio to reenforce immersion.
f. *Day* and *Night* scenes have unique audio and skyboxes.

g. Animated operable doors that open when the user is a certain distance away from them. In addition, a Boolean was created to define an “open once” door and a door that opens and then closes depending on the user’s distance. This was implemented because I originally had all doors open and close, however, this was quite distracting when moving through the interiors with doors opening and shutting all willy nilly. Therefor, all interior doors only open once.

h. Custom skyboxes appropriate for each scene.

i. Custom Terrain modeled natively in Unity.

Project Scoping

Features And Dependencies

5. Features

a. (6) home 3d models along with a HUB space

- i. **Revised: cut scope down to (3) models and a Home Space.**

b. Custom materials

- i. **Revised: Used custom materials I made along with some substance materials I had access to.**

c. Custom Skybox

- i. **Completely these.**

d. Custom modeled Terrain

- i. **Revised: natively modeled terrain and texture through unity. Remarkable simple.**

e. Spacial audio

- i. **Revised: cut down amount of audio to just a few ambient sounds.**

6. Game Loop

a. Interaction with market.

- i. **Revised: initial aspirations was a more fluid interaction of cycling through images. Final was revised down to a simpler way of cycling using a point-and-click method.**

b. Laser pointer to interact with “market”

- i. **This turned out essentially the way I was hoping for.**

c. Scene warping

- i. **Revised: Tracking the object’s distance to the User’s head was being troublesome (something about local space and setting the model’s parent to controller), I found it easier to track the controllers distance to the user’s head while a boolean *isGrabbingModel* was**

true. What this means was if the the User was grabbing the model, it knows to warp when that controller is brought close to the head.

d. Speech Recognition
   - **i. Revised: there are only (3) keywords now the User can activate. “Day”, “Night”, and “Home.”**

e. Locomotion
   - i. **Revised: Decided to use the pre-built SteamVR locomotion. One thing I’m learning about software engineering is that there is no shame in using something that someone else has built. In fact, its not only the smart thing to do, its efficient as well.**