Virtual Reality Logbook

By: Owen Connell

Problem Statement:

For the longest time I've wanted to have hands on experience with Virtual Reality (VR) that wasn't just playing games. Then I had remembered the first night I had VR I was sick to my stomach with nausea from motion sickness. This was after 2 hours of continuous play on my first day. It took me until about 20 hours of use time before I had gained my 'VR legs.' This got me thinking of possible solutions to the severe initial motion sickness. As Virtual Reality becomes more available, having motion sickness gate the user's experience will become a big nuisance.

Background Information: Motion Sickness in Virtual Reality

- Motion sickness is mainly due to poor latency in the headset
- The disconnect between your perceived environment and your actual environment causes confusion in the brain and can cause nausea
- Interacting with objects that aren’t bound in space throws of people's sense of balance
- The lack of engagement of every sense can cause a disconnection in experiences and lead to motion sickness
- Moving in VR, whist stationary in reality, confuses the brain and causes bad motion sickness (Similar to car sickness or the nausea you’d feel on a rollercoaster)

Source: Stanford, Mitigating Visually-Induced Motion Sickness in Virtual Reality

Brainstorm

Brainstorm Title	Description
Example: Notebook of Secrets	When you open a notebook, a light will shine on the micro:bit. It will flash a diamond four times and scroll, “Intruder!”
Idea Name: VR Motion Sickness Program (VRSP)	An application that helps train the user to get used to VR and some of the movement types they’ll use to ‘numb the brain’
Idea Name: Improved Haptic Feedback	Through simulating touch by means of gloves, shoes or a vest you can trick the brain into being unable to tell the difference between the two realities. This will greatly decrease long term motion sickness
Idea Name: Improved Hardware/Software	Lowering latency and greatly increasing the ppd (Pixels Per Degree) and ppi (Pixels Per Inch) of the headset (Possible with foveated rendering), this will decrease strain on eyes, and the fluidity will reduce motion sickness
Idea Name: Increased Field of Vision	There's a lot of ‘dead space’ in the peripheral of most modern VR headsets. Very few headsets that both have high performance, high resolution, and a high FoV are available to the normal consumer

Background Information: Foveated Rendering

Foveated Rendering uses both eye tracking and deep learning technology to pinpoint the location the eye is looking at in VR and upscale the quality at that point immensely while lowering the quality everywhere else. Your eyes can't tell the difference because they aren’t focussing on this non-priority space. The amount of processing power this saves a GPU allows for better looking upscaling of the focal point and all around performance improvements.

Source: Facebook, Deep Fovea: Neural Reconstruction for Foveated Rendering and Video Compression using Learned Statistics of Natural Videos (November 2019)

Design Matrix

Brainstorm Title	Criteria and Constraints				Total
	Solves the Problem 1-4	User Accessibility 1-4	Time to make 1-4	Uniqueness 1-4	
Idea Name: Virtual Reality Training Program (*VRTP)*	**3**	**3**	**3**	**2**	**11**
Idea Name: *Haptic Feedback*	**3**	**1**	**1**	**3**	**8**
Idea Name: *Improved Hardware/Software*	**4**	**2**	**1**	**2**	**9**
Idea Name: *Increased FoV*	**2**	**2**	**1**	**1**	**6**

As the Matrix shows, I ended up with having to create a program to train people to become accustomed to movement and interaction in VR to reduce long term motion sickness. From looking at the Steam and Oculus Store I haven't found any applications with this similar goal.

Project Ideas:

3 Separate Environments for Three Different Virtual Reality Movement Types

- 1:1 (All motion captured it directly correlated to motion in VR but the person is anchored by their real playspace in VR as well. All other movement types mentioned make us of 1:1) Game Example: Beat Saber
- Point and Warp / Teleportation (The user can teleport by pointing to a location. Depending on the application, this may either be to preordained locations or anywhere. Not as commonly used in recent years due to it being immersive breaking) Game Example: Budget Cuts
- Freemotion / Joycon (A person can move any which direction they choose by using the joy con on their controller. Has the most freedom but causes immense motion sickness) Game Example: Boneworks

A good example of a game that accommodates all three motion methods is: Half-Life Alyx

There exists other movement methods however these are the most common

Project Ideas Cont.

The user will load into a hub area where they are presented with multiple projections that each display text and an image to represent the type of motion that option will use. Upon selected one of the options by pointing your controller at it, the user will be teleported into the environment that was specifically designed for that type of movement. There will be an option from each environment to travel back to the main hub and select a different movement method. To engage the user and provide useful simulation, there should be some simple interactables available to the user to mess with. Along with helping the user get accustomed to VR's physics this will also help to prepare the user for any inconsistencies that may occur in VR. i.e objects not properly colliding with each other, objects snapping into position when grabbed, objects that have simulated weight and objects that dont, walls objects collide with but the user can clip through by physically moving themselves, and many other situations.

Design Process

I decided that I would attempt to create a basic framework of how I think the application would look. I attempted to create this prototype in Unity, due to its ease in making 3D environments and it's VR support, however I ran into multiple issues.

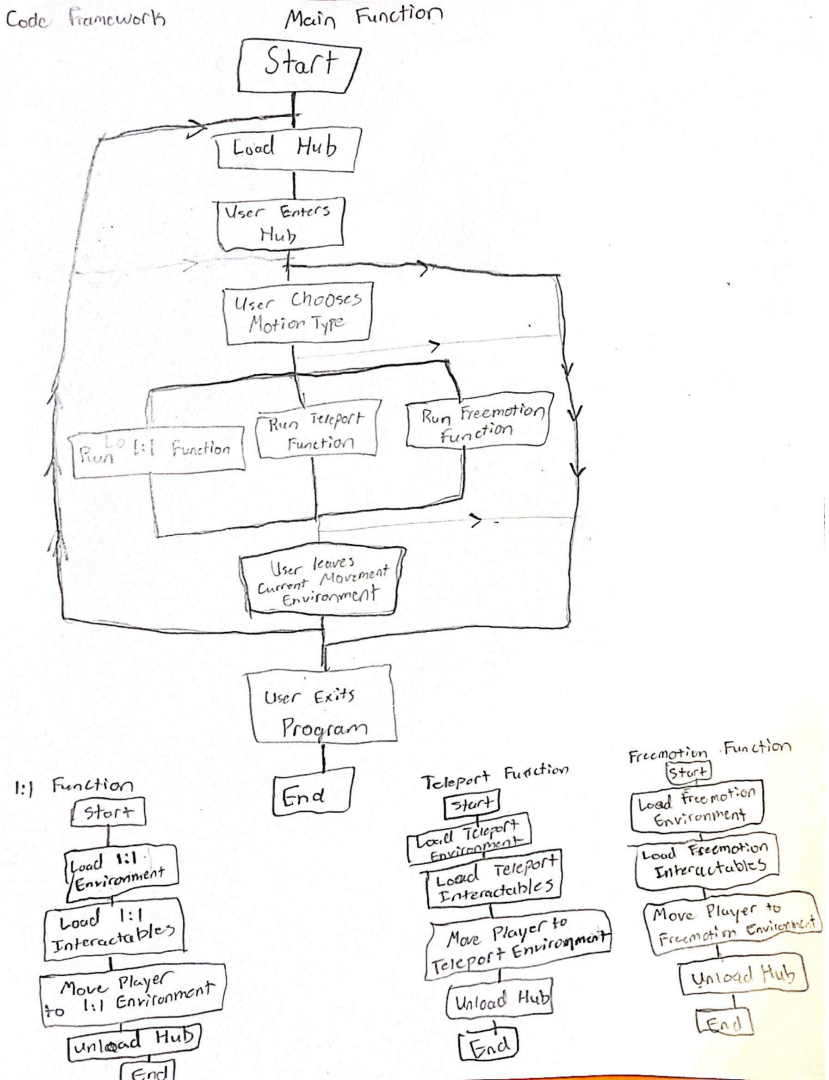
1. I don't know how to use Unity, yet.
2. I have an Oculus Quest 2 and don't have a USB-C port on my computer. Because the Oculus Quest 2 is special in the fact that it's the only headset console that uses cordless play instead of PC VR or corded play. Impressive technologically, bad for me. Normally you would need to purchase a USB-C cable to hook your Oculus Quest 2 to your computer unless your computer doesn't have one, in that case you've wasted $80 on a 6-meter cable you can't use. What makes it worse is that you can't use a USB-2 adapter, the Oculus application will not recognize your Oculus Quest 2, it will instead think it's a Oculus Rift, and this keeps it from properly connecting. Thanks Facebook.

Design Process Cont.

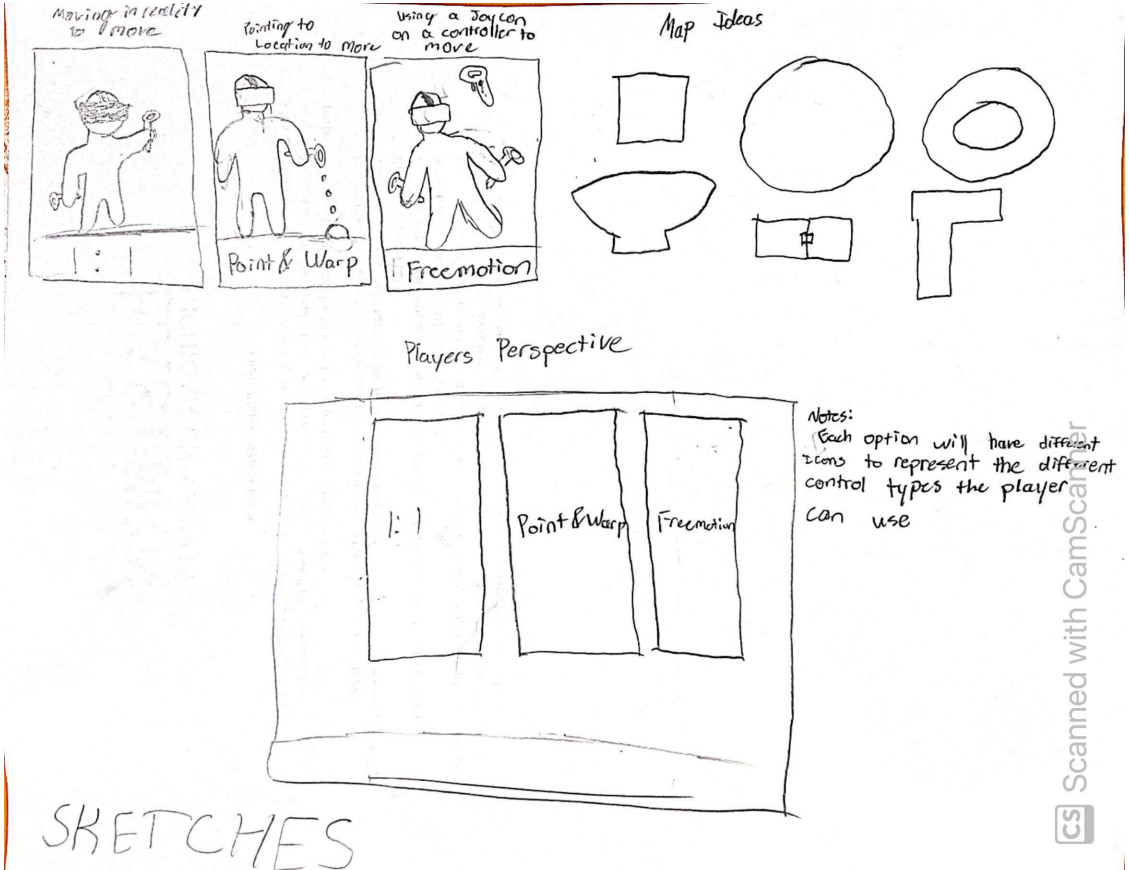
After learning through trial and error, I determined that at the moment I would have to forget about hooking up my VR headset to my computer. This means I can't test out any code to make sure it properly works in VR. Add the fact that I'm unfamiliar with Unity and I find myself set back in progress.

While I can't do any coding, the most I can do in the meantime is right out the steps I wish my code to be able to do. By having a simple framework to follow the future process of coding should be hopefully quicker and simpler. Other than that I also came up with some simple shapes that I hope give me inspiration for future environment designs

Code Layout


Code Framework
Main Function
Start
Load Hub
User Enters Hub
User Chooses Motion Type
Run 1:1 Function
Run Teleport Function
Run Freemotion Function
User leaves Current Movement Environment
User Exits Program
End
1:1 Function
Start
Load 1:1 Environment
Load 1:1 Interactables
Move Player to 1:1 Environment
Unload Hub
End
Teleport Function
Start
Load Teleport Environment
Load Teleport Interactables
Move Player to Teleport Environment
Unload Hub
End
Freemotion Function
Start
Load Freemotion Environment
Load Freemotion Interactables
Move Player to Freemotion Environment
Unload Hub
End

UI and Environment Sketches

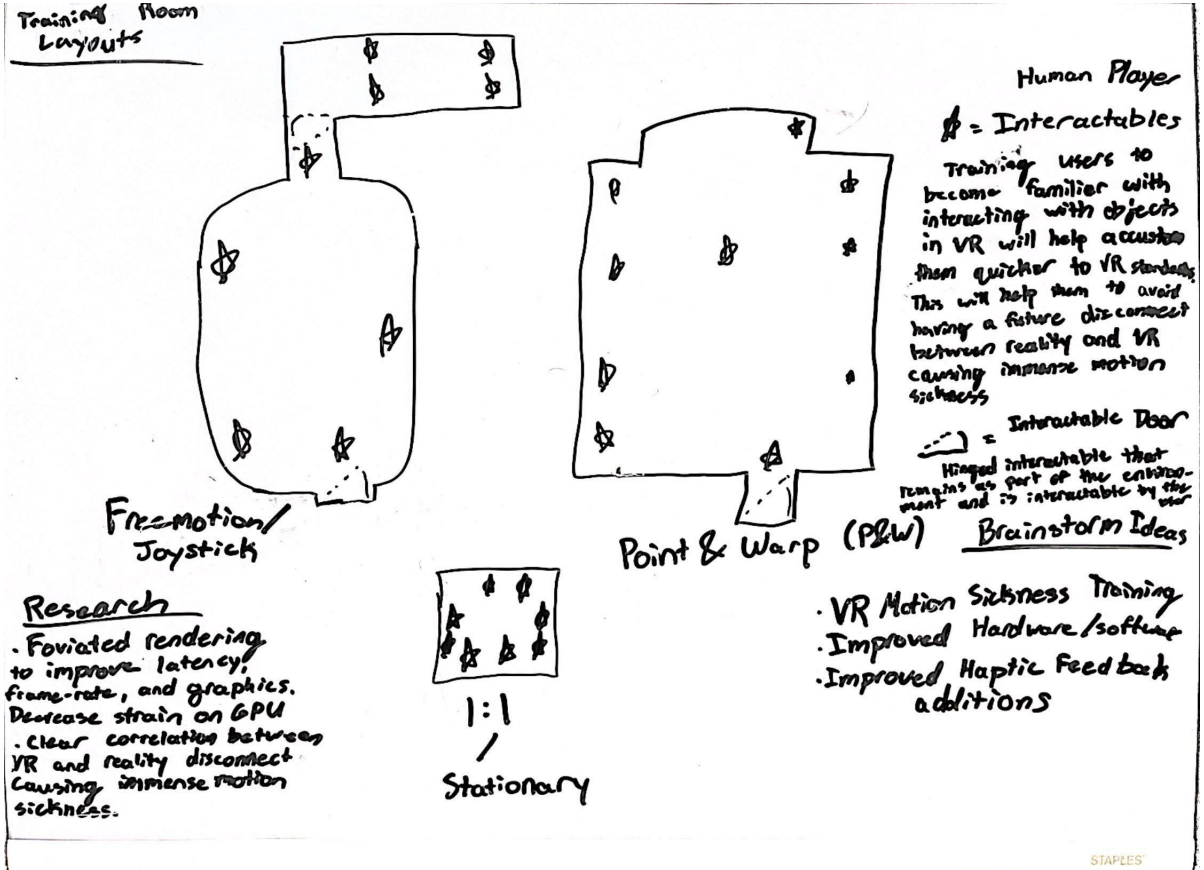


Design Process Cont. 2

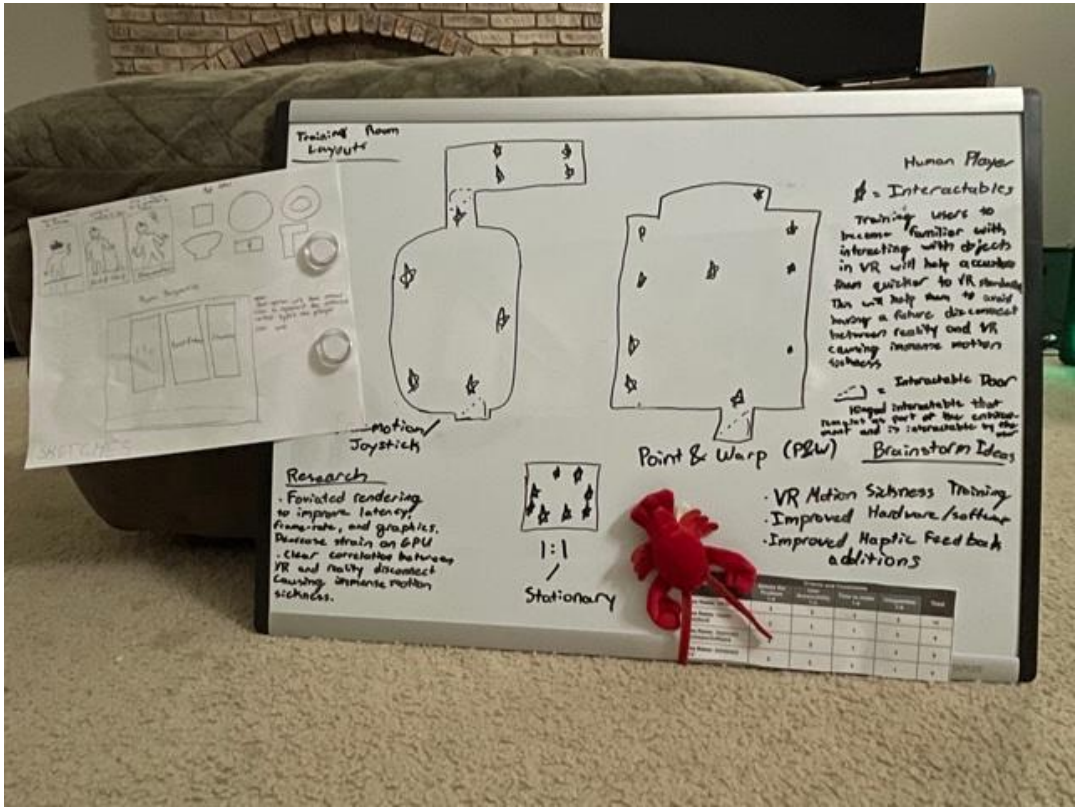
After writing out some simple designs for the different environments I came up with a more specific design for each movement type besides 1:1. All current drawings are 2D and when I'm able to design these layouts in Unity, if ever, I'll just import an image of the 2D layout and build 3D models based on the layout.

As I find myself shorter on work time, I'm doubtful that I'll be able to get proper equipment to test out any code I make in Unity. Thus I've decided to instead focus on fleshing out different possible user interactions within each environment.

Iterated Environment Designs



Presentation Whiteboard Photo



Next Steps.

As is, this project is far from total completion. I have written proof of concepts for what I wish for my code to accomplish and how I want my application to look but I have yet to properly convert anything into code. The absolute next steps would be starting to create the environment in 3D within Unity, from there it would depend on whether or not I have the ability to use VR when connected to my PC. Assuming I do, the next step would be enabling the multiple different movement types and enabling/disabling them depending on the environment chosen. Then once those are properly implemented I'll need to create simple interactions for the user to use, such as picking up objects, pushing objects, opening doors and throwing objects. At that point I would feel that the application has reached an adequate point. After it's aesthetically pleasing to look at, it would be 'complete.' It would be nice to implement other forms of interactions however I don't know how long that may take.