Exploring Shake Menus with Hand Tracking in Virtual Reality

Ryan Miles
Georgia Institute of Technology
rmiles30@gatech.edu

Harry Wang
Georgia Institute of Technology
hwang703@gatech.edu

ABSTRACT

Interaction in virtual reality should be convenient and immersive. Using hand-tracking to interact with a virtual environment can be a familiar experience since users have experience with hand interaction in the real world. Hand-gestures can allow users to interact with the underlying control and menus in a virtual environment. A hand gesture involving shaking an object could be used to display a menu around an object called a shake menu. We will conduct our experiment by designing and demoing a virtual reality environment with interaction through shake menus using hand tracking. We plan on creating an interaction that is convenient and immersive for the user through designing different types of radial menus and how they will interact with the world. Our conclusion will hopefully suggest possible approaches to best present and interact with shake menus.

Author Keywords

Menus, Shake Menus, Hand Tracking, Virtual Reality

CATEGORIES AND SUBJECT DESCRIPTORS

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies, Interaction Styles*;

INTRODUCTION

As VR technology and its devices become more popular among consumers, the question of how to design the UIs in a 3D virtual environment also gains relevance. VR software developers have taken the tried and tested point and click system of modern desktop GUIs and incorporated it into much of today's VR applications, using the controllers or even hands as pointers on a plane in 3D space. However, the fact is, the mouse and keyboard input paradigm of desktop computers does not translate well into these new 3D UIs as it limits the user's input to a single plane rather than utilizing the freedom that comes with another axis of motion.

ACM ISBN .
DOI:

We are interested in any new and experimental form of input/control scheme to navigate 3D UIs in VR as the more variety there is in options of input, the closer we will get to a more intuitive input scheme, or at least one that can utilize the 3D environment more than a flat plane. The concept of "shake menus" and "shaking gestures" are a less discussed paradigm, providing a relatively low skill-ceiling to utilize in AR and VR and have been proven to work in very rudimentary activities, such as making a menu appear upon shaking an object. [20]

An interesting point of these "shaking gestures" comes the ability to track movement on all three axes in a 3D environment. This allows the user to perform different inputs depending on the direction of the shaking and gives increased selection options for menus compared to the left and right click of a mouse. Additionally, the type of UI that accompanies such an input could take the form of a dynamically placed radial menu, each selection being in a different direction that a user could make a shaking gesture in, or a pie menu, similar to a color picker in a painting program, in which the input follows a similar direction.

In the remainder of this paper, we will discuss the work related to these types of input and menus as well as how we will design and conduct an experiment to determine the viability of them in VR. We will focus solely on the user experience when using shaking gestures and how it compares with traditional point and click menus that are commonplace in today's VR UIs.

RELATED WORK

Hand-Tracking & Hand Gestures

Hand-tracking Gestures is an actively researched topic in HCI. Some research involves wearing a type of data glove that contains multiple sensors. These sensors were then analyzed in order to build hand models. [11]

Other research has been done on hand-tracking from a video input and building a spatial model. [14] Relying on video input can create a more "natural" user experience. To recognize gestures, some research involves using shape models [12] or trained hidden Markov models [5] to recognize gestures. Both papers have also recognized that while the tracking and gesture detection is very accurate there will still be some inaccuracies. Due to this limitation we expect similar results where sometimes hand gestures will not be detected or the tracking might not always be fully accurate.

The use of hand tracking has allowed for multi-user interaction with tabletop displays with intuitive controls. [8] Hand

gestures can also be mapped to specific commands in order to control a computer task. [17]

From these studies we hope to build on exploring different hand gestures and mapping them to specific commands in our program. Due to the data gloves used in these papers being expensive and creating a less "natural" experience we will be focusing on using video input for hand tracking. We plan on utilizing a software development kit (SDK) for hand-tracking so we don't expect to build our own models. Consequently, we expect some inaccuracy when detecting hand gestures, at least more so than if we were to use data gloves.

Interaction Techniques in Virtual Reality

Controllers allow for 6 degrees of freedom (6DOF) control along with analog and digital buttons. Some include thumb-sticks or track pads for additional input. Controllers can also provide haptic feedback to a user. The controllers can then tracked and communicate with the virtual reality system to translate real world operations into a corresponding action in virtual reality. [18] One limitation we are expecting is the loss of ability to provide haptic feedback from using hand-tracking over controllers.

One study explored the types of interaction using hand-tracking with a card game in a virtual environment. The main mechanic was picking a card through a tapping gesture to select one of the virtual cards. Other gestures can be grabbing with the entire hand or pinching with two fingers. The study also found a 81.82% positive assessment of hand interaction. [10] The main factors that factored into the positive assessment was connivance and immersion. We expect to leverage similar benefits for the user experience through hand-tracking and gesture interaction.

The virtual reality headset can be used to provide controller-less interaction as well. Sensors on the headset can provide current position and rotation to be mapped inside the virtual reality system. Interaction can be through a gaze where the user is currently looking through the use of the headset's position and rotation. Using on-head detection can provide control to spend commands when a user is not currently wearing the headset or just put on the headset.

Eye tracking can provide similar controller-less interaction techniques. Eye tracking allows for gaze movement without the need for a user to move their head. One study compared two different eye trackers using gaze on selecting specific items in a virtual reality environment. [15] Performance limitations in accuracy were found with the VR technology's display. We do not expect to use eye tracking, so we don't expect to have these specific limitations.

Each different interaction technique can be combined in order to provide a more immersive environment in virtual reality. Some examples include using the virtual reality headset for a users gaze to look around while holding onto an object through the use of a controller or hand-tracking. By combining headset gaze and hand tracking depth can be created where items can only be selected through hand tracking if the user is currently looking near the item. We expect our study to exhibit a more immersive interactions through the use of taking advantage of combining hand-tracking and position and rotation from the headset instead of controller or gazed based interaction.

Radial Menus & Pie Menus

Pie and radial menus arrange items around a center point. Radial menus allow items to be selected through clicking where pie menus allow items to be selected by moving the cursor slightly in one direction. Early research has shown a decrease in pointing time when using pie menus compared to linear menus. [6] We expect similar decreased pointing time in virtual reality from this study when using a pie or radial menu.

Both menus can have a single hierarchy all selections are contained within one level from the center. Multi-hierarchy pie and radial menus can contain one level of items and selecting an item can create a second level of more items. Multiple levels can extend out from a longer radius from the enter or can also be layered on top of the previous level. One study explored a pie menu with customized interaction in each selection window called extended pie menus. [9] The selections on the pie menu can be extended when clicked on to a customized interaction such as a color picker or slider. Limitations were found when creating text boxes in the pie menu's sections and only a limited number of elements can be efficiently used. Due to these limitations, we plan on using on a select number of options in a radial menu.

The study also discussed multiple hierarchical menu layouts such as depth offset, in-plane offset, linear in-plane offset that we plan on exploring when creating our radial menus when adding hierarchies. Overall, Pie and radial menus can provide a on-demand selection menu for a user to interact with.

Shaking

With the use of accelerometers, shaking can be detected and used as an interaction gesture. One study has created smartphone and smartwatch motion gesture mappings from shaking to a specific action. [13] Shaking can also be specified along a specific axis to create multiple shaking gestures. One motion gesture called the DoubleFlip involves rotating a phone back and forth twice along a phone's long side to create a flipping motion. [16] The DoubleFlip gesture can be used to perform a command such as open the phone's camera or clear the screen.

The DoubleFlip study used Dynamic Time Warping (DTW) to determine if the gesture was performed and discussed how different DTW thresholds effect False and True positive rates. We expect to perform a similar method of tuning our threshold values when detecting the shake gesture in our program. We do not plan on using the Dynamic Time Warping (DTW) algorithm so the positive/false positive rates may differ with our study.

Another study explored the shaking gesture in augmented reality (AR) to display a menu called a shake menu. [19] The benefit of the shake menu included very little training to detect the gesture. We plan on extending this research in an VR environment. The study explored menu placement and found user's preferred the clipboard (static) and world-referenced placement. Some limitations were found in the AR

environment about orientation of the menu items. We do not expect this to be a problem in VR due to having more control over the virtual environment and rendering of 3D objects in the world.

EXPERIMENT

Experiment Setup

The experiment will be performed on an Oculus Quest standalone VR headset (henceforth to be referred to as just "the headset") using the Oculus hand-tracking feature as inputs. We will use the device's built in developer mode to side-load our application onto the headset.

The application will consist of a simple VR playground environment with the main focus being the shake menus that participants will need to navigate through. Participants will be able to select We will have a roughly square 25 square foot area that is clear of obstacles as well as two chairs at least 3 feet away from the square to serve as vantage points.

The participant will stand in the roughly square area, and we will be monitor their physical movement as well as what they see through the headset either while sitting in the chairs or moving around the square, making sure to keep at least a 3 foot distance between us and the square. A mobile phone or Google Chromecast streaming device alongside a laptop computer will be used to monitor the participant's point of view during the experiment.

Task

Participants will be asked to put on the headset and attempt to navigate through a set of applications using shake menus and static menus during the experiment. The order will remain the same between participants to remove any bias in initial muscle memory.

Pre-made Shake Menus

The main shake menu that will be tested will be similar to Sean White's study of shake menus in AR. [19] We will be exploring the object-referenced for the shake menu and world-referenced static menu in the virtual environment.

Planned Procedure

Upon putting on the headset, participants will be given a brief tutorial on how to navigate the shake menu before being given an instruction from us in the form of a set of menu selections to make in order.

Upon completing the selections, a confirmation sound will be played in the application, and we will give verbal confirmation that the participant completed the set. The participant's accuracy in number of attempts to complete the set as well as their speed of completion in time it took for them to complete the set will both be recorded on a spreadsheet for later perusal. After they acknowledge that they have completed the set, we will give another set of selections to make and repeat this process until the participant has completed 10 sets of selections or gives up before completion. Additionally, the participant will be given an option to retry a set of selections at any time if they did not make the correct input by pointing their head straight down, triggering the reset event.

At any point in the experiment, the participant can take a break by removing the headset and sitting down or walking around the experiment area. If the participant expresses any feelings of discomfort or nausea, we will stop the experiment immediately and allow them to take a break or stop the experiment if they deem it necessary.

IMPLEMENTATION

To use Oculus Quest's hand tracking experimental feature, we chose to go with Unity. We used Unity 19.3.9f1 and Oculus Unity Integration 15.0. [4] All resources and assets were either built by us or were available in the Oculus Unity Integration package.

Before starting we both followed the 10+ hour learn.unity.com course that covers designing for virtual reality in Unity. [2] The course taught us the basics of using Unity and how to set up an environment using the Oculus Integration and a few techniques such as hand presence, interaction, and locomotion. We also learned how to deploy the project onto the Oculus Quest for testing. Unfortunately, all the guides and tutorials used controllers, so we used the developer.oculus.com page to get a start with the hand tracking. [1]

Testing Environment

We designed the demo Unity environment to make it straightforward for anyone to test our project. When the user is loaded in they will be facing a table with a cube resting on top of it. We added three buttons to help control the demonstration. One button will reset the scene in case the cube falls off the table or any issues occur. Another button allows for easy switching between user interfaces. The last button is used for testing the static world referenced user interface.

Hand Tracking

We referenced the samples Oculus provides for the hand tracking in our implementation. Unfortunately, because the SDK was, and at the time of writing the report still is, experimental, there was not a lot of documentation. Additionally most of the existing documentation was out of date since the hand tracking SDK was changing quickly at the time. We started on the HandTest sample scene and built onto it. The sample gave us the assets needed and allowed us to track our hands in virtual reality right out of the box. We then extended the base OVRGrabber class to add a custom pinching detection. This allows us to grab an item and hold onto it as long as we continue pinching it with our index finger and thumb. The custom pinching detection uses a method called GetFingerPinchStrength(), which returns the estimated strength of a pinching action based on your fingers' positions. We tested various threshold values and decided on 0.7 as the minimum pinching strength for a good balance between detecting and holding a pinch. Using that, if the minimum strength is met to pick up the item and our hand was close enough to the object, then it would be picked up. If the minimum strength is no longer met after the item has been grabbed, then the object is released.

Shake Gesture Recognition

For the shaking gesture, we coded a physics-based approach to detect a "shake." We used a rapid change in acceleration to denote a shake. To do this, we first calculate the object's velocity from: 2 * ((position - lastPosition) / dt) and then use: [(velocity - lastvel) / dt], lastvel being the last measured velocity, to get acceleration. We quickly ran into issues of detecting false positives for shaking on long, drawn-out movements as well as light wrist flicks, so we referenced some of the advice in a previous Stack Overflow post for detecting shaking in Unity. [7] We then subtract a lowPassValue to remove any small yet erratic accelerations and compare it to our shake threshold of 4.0. We then log the event in a buffer of the last 10 events and use it as a rolling average to remove any more false positives due to slight movements. The buffer helps filter out small acceleration values and requires at least a few good shakes, or one strong flick, to trigger the menu. We also check if the hand is grabbing an item to detect when an object is being shaken. We then use this information to decide whether to expand or collapse the radial menu.

Some of our notes when tweaking the shake detection, specifically the delta time (dt) parameter

- > .2 == Continuous, erratic shaking, too difficult
- .2 == Strong flick, tiring
- .185 == Goldilocks flick, not too tiring, won't sense long movement unless dragging across entire hand tracking area
- .17 == Medium flick, still sensitive on long movements
- < .17 == Light flick, picks up every small movement

Radial Menu

The radial menu expands outward, using the center of the cube as its center point, when the cube is shook. We used three buttons, each 90 degrees apart, around the cube as our menu options. The 4th button was foregone, because grabbing the cube with your right hand would block the 4th option to the right of the cube. Once the menu has expanded, the user can then stick their hand through one of the options to select it. Currently, the options list different colors and when selected, will change the cube's color. We also implemented a "tracking" aspect into the UI wherein the option buttons will always be facing the user's head, and if the user moves, it will follow them in order to always be visible and easy to read. Because we are using hand tracking we made it so in order to activate one of the radial options the user must press it instead of pointing and clicking with a controller.

World Referenced Menu

The menu is implemented in a similar fashion to the radial menu, only without matching its position to another object, and when pressing the buttons, the color of the cube will change accordingly. This menu allows us to compare some of the shake menu functionality to a static based user interface.

Experiment Video

A video of the experiment can be accessed through YouTube. [3] The video demonstrations the hand-tracking, shake menu, and the world-referenced static menu.

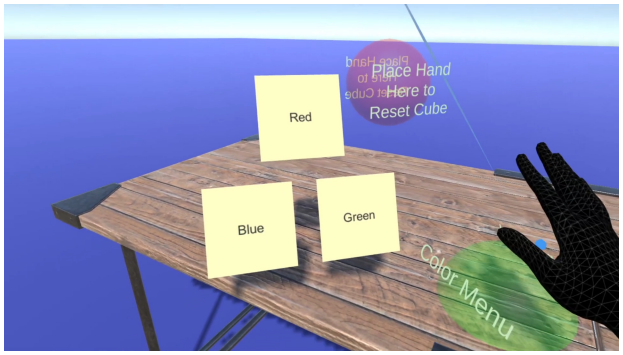

Figure 1. World Referenced Menu

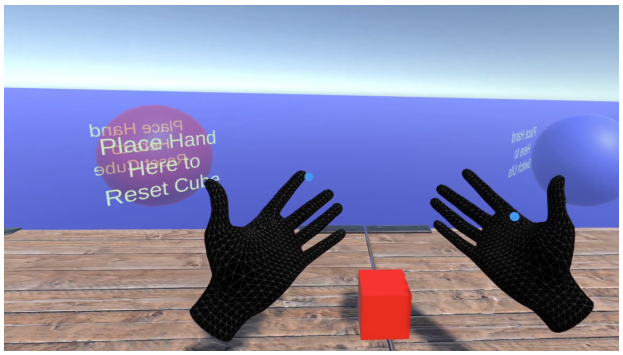

Figure 2. Hand Tracking in the Environment

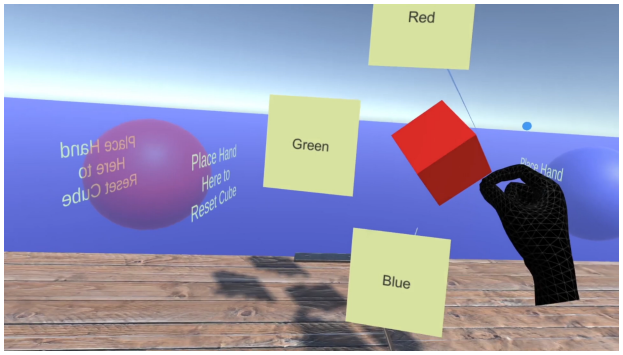

Figure 3. Hand Tracking in the Environment

DISCUSSION

Unfortunately due to extreme circumstances, we were unable to perform any experiments with our planned procedure, and instead relied on anecdotal accounts from close individuals we knew and ourselves for any feedback on our implementation. That being said, our preliminary testing has given this feedback regarding the menus:

Firstly, the shaking gesture can get a bit tiring after prolonged use and reuse. This problem is virtually non-existent in the mainstream point-and-click UIs as the process for bringing up a menu would only take the press of a button. We had done some preliminary testing for fine-tuning the shaking sensitivity and detection, and for our demonstration, we only used the "Goldilocks" shaking calculation in which we both thought the shaking was neither too tiring or too sensitive. However, especially given the feedback, further testing is required with a larger sample size to determine what parameter values would be optimal for prolonged and easy use among a more generalized population.

Secondly, even with our efforts to minimize false positives, we still occasionally encountered a false positive from a rapid movement across the screen or from tracking inaccuracy. Unless if we were to write our own hand-tracking SDK, the latter of those two will not change until a hardware or software revision releases. However, on the subject of long movements still being detected, we can perform more experiments to determine which parameter(s) needs to be changed to reduce them. We suspect that simply increasing the buffer size may provide better results.

And lastly, despite the initial hurdle to get used to the shake menu paradigm, the users who tried the shaking UI said that they enjoyed using it and preferred it over the traditional static UI. Again, we will need to conduct more experiments with a larger sample size to obtain more meaningful data, but given our anecdotal experience with them, shake menus seem like a viable alternative to a traditional 2D UI, especially when objects need to be interacted with.

CONCLUSION AND FUTURE WORK

Our conclusion for the usability and potential of shake menus in virtual reality is that while initially jarring, the shake menu can be an effective and user-friendly UI for a variety of applications. That being said, we would like to explore the topic more in the future. Our first line of future work would be to conduct more experiments on the UI in its current state with a larger sample size to further fine-tune the parameters used in detecting shaking and decrease the false positive rate. Additionally, we could modify the UI or create more variations of menu styles to conclude which style, along with method of input, would be better suited for a tracking shake menu. Furthermore, creating and allowing users to experience different testing environments or simulating different situations with experiment procedures would help identify in which applications and use cases shake menus would be best suited for, which has the potential to drastically change our current conclusion on the viability of shake menus in mainstream application.

REFERENCES

[1] CustomHands Sample Scene. (????). https://developer.oculus.com/documentation/unity/unity-sf-customhands/?locale=en_US

[2] Design, Develop, and Deploy for VR. (????). https://learn.unity.com/course/oculus-vr

[3] 2020. *Shake Menu with Hand Tracking Demo*. https://youtu.be/BEcxxFQL5rg

[4] 2020. Unity Integration. (Apr 2020). https://developer.oculus.com/downloads/package/unity-integration/

[5] Nguyen Dang Binh, Enokida Shuichi, and Toshiaki Ejima. 2005. Real-Time Hand Tracking and Gesture Recognition System. In *Proceedings of International Conference on Graphics, Vision and Image Processing (GVIP-05*. 362–368.

[6] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. 1988. An Empirical Comparison of Pie vs. Linear Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '88)*. Association for Computing Machinery, New York, NY, USA, 95–100. DOI: http://dx.doi.org/10.1145/57167.57182

[7] Bechard Dan. 2015. How can I detect a shake motion on a mobile device using Unity3D? C. (Jul 2015). https://stackoverflow.com/questions/31389598/how-can-i-detect-a-shake-motion-on-a-mobile-device-using-unity3 31389776

[8] K. C. Dohse, T. Dohse, J. D. Still, and D. J. Parkhurst. 2008. Enhancing Multi-user Interaction with Multi-touch Tabletop Displays Using Hand Tracking. In *First International Conference on Advances in Computer-Human Interaction*. 297–302. DOI: http://dx.doi.org/10.1109/ACHI.2008.11

[9] S. Gebhardt, S. Pick, F. Leithold, B. Hentschel, and T. Kuhlen. 2013. Extended Pie Menus for Immersive Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* 19, 4 (April 2013), 644–651. DOI:http://dx.doi.org/10.1109/TVCG.2013.31

[10] Seunghun Han and Jinmo Kim. 2017. A Study on Immersion of Hand Interaction for Mobile Platform Virtual Reality Contents. *Symmetry* 9 (2017), 22.

[11] J. Kim, N. D. Thang, and T. Kim. 2009. 3-D hand motion tracking and gesture recognition using a data glove. In *2009 IEEE International Symposium on Industrial Electronics*. 1013–1018. DOI: http://dx.doi.org/10.1109/ISIE.2009.5221998

[12] M. Kovalenko, S. Antoshchuk, and J. Sieck. 2014. Real-Time Hand Tracking and Gesture Recognition Using Semantic-Probabilistic Network. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. 269–274. DOI: http://dx.doi.org/10.1109/UKSim.2014.49

[13] Steven McGuckin, Soumyadeb Chowdhury, and Lewis Mackenzie. 2016. Tap "n" Shake: Gesture-Based

Smartwatch-Smartphone Communications System. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction (OzCHI '16)*. Association for Computing Machinery, New York, NY, USA, 442–446. DOI: http://dx.doi.org/10.1145/3010915.3010983

[14] Vladimir Pavlovic, Rajeev Sharma, and T Huang. 1997. Visual Interpretation of Hand Gestures for Human-Computer Interaction A Review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19 (08 1997), 677 – 695. DOI: http://dx.doi.org/10.1109/34.598226

[15] Thies Pfeiffer, Marc Erich Latoschik, and Ipke Wachsmuth. 2008. Evaluation of binocular eye trackers and algorithms for 3D gaze interaction in virtual reality environments. *JVRB-Journal of Virtual Reality and Broadcasting* 5, 16 (2008).

[16] Jaime Ruiz and Yang Li. 2011. DoubleFlip: A Motion Gesture Delimiter for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2717–2720. DOI: http://dx.doi.org/10.1145/1978942.1979341

[17] Jane J Stephan and Sana'a Khudayer. 2010. Gesture Recognition for Human-Computer Interaction (HCI). *Int. J. Adv. Comp. Techn.* 2, 4 (2010), 30–35.

[18] James S Webb. US9678566B2, June 2015. Hand-held controllers for virtual reality system. (US9678566B2, June 2015).

[19] S. White, D. Feng, and S. Feiner. 2009. Interaction and presentation techniques for shake menus in tangible augmented reality. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*. 39–48. DOI:http://dx.doi.org/10.1109/ISMAR.2009.5336500

[20] Sean White, David Feng, and Steven Feiner. 2009. Poster: Shake menus: Towards activation and placement techniques for prop-based 3D graphical menus. In *2009 IEEE Symposium on 3D User Interfaces*. IEEE, 129–130.