Head Orientation Control for VR Accessibility

Use and interact with virtual reality without the need for hands!

NIKOLAS MCNAMEE, University of Washington

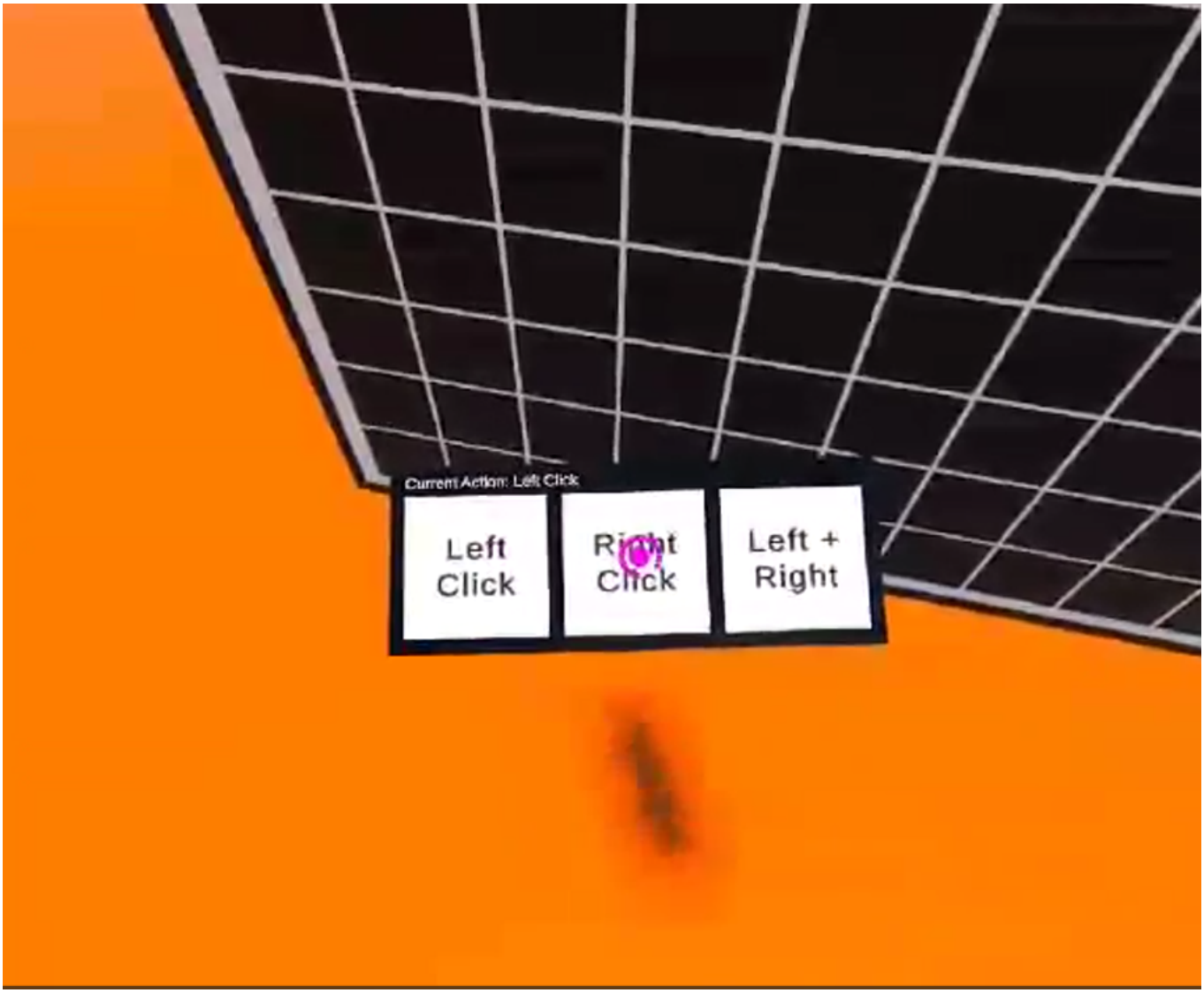

Fig. 1. The menu, or "palette", where the user can select which type of action will be performed when their head orientation remains still for long enough. The magenta cursor in the center of the user's view will determine what object is interacted with and around it is an indicator showing when the action will be performed. The indicator will reset upon the head orientation changing too much.

Abstract - Virtual reality (VR) with head-mounted displays (HMD) provides new ways to use and experience technology compared to the ubiquitous touchscreen or laptop/PC paradigms. With the near-unanimous use of HMDs for VR, there is ample opportunity to improve user experience and accessibility by taking advantage of head orientation and eye-tracking. This report explores developing a control scheme only reliant on the user's ability to move their head, making no assumption of the user's hands. Unlike previous designs of similar nature, e.g. Apple's dwell control on Vision Pro, this report aimed to add head orientation gestures to the control scheme, which can be mapped to additional user actions. While the work done here shows a

Author's address: Nikolas McNamee, mcnanik@cs.washington.edu, University of Washington.

usable prototype for such a control scheme, there is plenty of room to add more head gestures and user customization to make a truly seamless–and handless–control scheme.

1 INTRODUCTION

VR with HMDs has the potential to give people who cannot use traditionally hand-reliant devices better solutions to interact with technology. Hands-free control schemes for phones, laptops, and desktops usually require external devices or use of camera and/or microphones. Taking advantage of HMD's head-mounted-ness, an alternative control scheme can be used based on the orientation of the device which is directly related to and controlled by the orientation of the user's head.

Currently, control schemes based on eye tracking are the most popular option for hands-free accessible VR control. The most mainstream VR operating system that has eye tracking controls for accessibility is the Apple Vision Pro with the dwell control scheme. Here users can control and interact with any object or UI element that they could with the usual hand-based scheme, but instead with their eye gaze.

Head orientation based controls are not often used at all, and it seems like there is no good example of it on mainstream devices. Thus, head orientation continues to primarily be used to control the orientation of a camera. However, this report aims to show there is much more potential to using or incorporating head orientation into a usable control scheme. In particular, head orientation can also be used to make a dwell-like system where the focus is not determined by eye gaze but rather a ray straight forward from the user's head. Head orientation also offers more gestures than just having gaze remain still. Head nods, shakes, and even bobbles (common in south Asia) are all common gestures that can be detected with a HMD's IMU data, and can be used improve upon a dwell-like experience.

After implementing a prototype and demonstrative application, this report finds that a head orientation based control scheme is viable and offers an accessible way to use VR for people who have any hand related issues. While eye tracking may provide a superior cursor/focus selector, head gestures such as nodding or shaking the head could be incorporated to the control scheme in the future for the user to optionally map shortcut actions to.

1.1 Contributions

This report details the following contributions:

- A proposed method to determine the stillness of the user's head orientation based on the angular difference being within a threshold compared to an initial orientation
- A proposed method to detect head orientation gestures based on back-and-forth movements.

2 RELATED WORK

2.1 Eye Tracking Control

Currently the only (as far I know) state-of-the-art, handless, unique to HMD VR, accessible control scheme is Apple Vision Pro's dwell control. Here, users can leave their gaze on an interactable element/object and a timer will start. If the user moves their gaze before the timer ends, the timer will reset on the new gaze. Once the gaze remains long enough so the timer can fire, the user's action will be triggered, where the action was chosen by a separate menu in the user's view. This separate menu/palette is controlled in a similar manner, with the user's gaze being able to pick what action or gesture can be performed, e.g. tapping, zooming, scrolling, etc. In the dwell control settings, the user can adjust how long the gaze timer lasts and the tolerance for how short a distance will trigger a gaze change and timer reset [Apple [n. d.]a][Apple [n. d.]c]. Currently, this is the best system-level control scheme for handless VR use.

2.2 Other Hands-Free Control Schemes Outside of VR

Many companies provide hardware and software to aid people who need hands free controls. One example is Cephable, which offers controls based on facial gestures, spoken voice, camera detected head motion, and mechanical switches that can be used with parts of the body other than hands [Cephable [n. d.]]. Many of these technologies can theoretically be used with a HMD device, such as voice control or face-facing cameras detecting winks or mouth gestures.

2.3 Head Orientation

While there have been studies comparing the use of head orientation or head gaze control to eye gaze control [Deng et al. 2022], the closest thing to a commercially available head orientation control is an option in Apple Vision Pro to use the head gaze as a pointer [Apple [n. d.]b]. You might be able to connect the system to dwell to get head orientation controlled dwell, but I could not find concrete evidence of this. Despite the lack of commercially available head orientation controls, there is some user desire for the technology to be used and improved, shown in various user forum posts [loves 2022][qpr1324 2021].

3 METHOD

This section will describe my version of dwell-like head orientation controls with detecting a still versus moving head orientation, and it will describe detecting head gestures like shaking or nodding accurately with minimal false positives. It will also discuss how to design these with accessible user experience in mind, using the WCAG 2.1 guidelines, which is a widely used accessibility standard for web content, whose standards apply to many other user experiences as well.

3.1 Dwell controls

Making a dwell control system involves the main problem of detecting when the user is focusing on an object and when the user is no longer focusing on the object. To start, the focus or pointer of a head orientated system will be a ray coming directly forward from between the user's eyes and the focused object would be the first object the ray touches in world space. Then the question becomes how to detect when the pointer/focus is changed by the user, which involves a design decision: if the ray moves off of a focusable object to something else, or if the angle of the ray changes by a certain amount. I went with the latter because it makes the action have more intent, by making the user hold the same amount of stillness

regardless of if the interactable is right in front of them or a tiny pinprick in the distance.

Now the design is as follows: on start-up, the initial orientation of the user's head will be used and it will set a range of motion, a maximum angular difference, that the orientation can change before focus resets to the next point. As long as the user's head stays still within the maximum angle from the initial orientation, the timer will continue until it fires. Once the timer fires, the user's selected action will trigger on the interactable, the timer will restart with the current head orientation as the new starting orientation. If the user's head rotates beyond the maximum range, then the timer will reset and the orientation they exited the original range with will be used as the next initial orientation for a new range around that orientation.

To decide what a good maximum angle for the tolerance of detecting a still orientation, I turn to WCAG 2.2.5, where it states the target for pointer inputs should be 44 by 44 CSS pixels [WCAG 2024a]. 44 CSS pixels equates to about 0.937 degrees of visual angle [WCAG 2024b], and a circle circumscribing such a square would have a radius of 0.6627 degrees. Thus, I used this value as the default maximum angle the user's head orientation could deviate because it would at least be able to maintain focus without resetting when aimed at an accessibility-sized element. Ideally, however, the user would have the option of adjusting the maximum angle in some settings menu to whatever works best for them.

For the user to be able to choose what action will be performed once the timer fires, there is a permanent menu/palette that follows them near their feet or legs, always in line with the center of their view, so that they can interact with it when they tilt their head down. When focus hits the menu, it will freeze its lateral movement so the user can move the pointer across it. The user can then use the dwell control to select which action they want to be using.

3.2 Head Orientation Gestures

Detecting head orientation gestures such as shaking first involves defining what a head shake looks like from the perspective of the HMD's orientation. If you perform the motion now, it is a rotation back and forth. The motion is constrained by the physical limitations of the neck and the start and end point of the motion can both be anywhere in this range and still seem like a head shake if your head went back and forth at least once. With these constraints in mind, I define a head shake as any yaw movement withing a limited timeframe that has a total angle traveled at least twice the angle between starting and ending angle.

Using the previously defined dwell system's definition of movement, once the user's head orientation breaks away from its current dwell orientation and another head gesture is not in progress, it will initialize a timer to detect a head gesture. Between when the timer starts and expires, the integrated angular speed of the headset will be used to calculate the total angle moved, and the initial starting angle will be recorded as well. When the timer expires, the absolute difference between starting and current angle will be compared with the total angle. If the total angle is greater than twice the start-to-end angle, that means the user must have made a back and forth motion with their head, and a the system can initiate whatever action is tied to detecting that head gesture.

There are many ways a human can move their head back and forth, this report only looks at the rotational movements. Thus, this detection algorithm can be easily applied to yaw movements (shaking), pitch movements (nodding), and roll movements (bobbling).

4 IMPLEMENTATION DETAILS

For my implementation of a prototype and demo for a head orientation controlled system, I used a Quest 2 and developed in Unity. The Quest 2 has Unity integrated packages, of which I importantly used the XR Rig, which provides a "camera" between the two eye cameras to use as the focus ray emitter.

Unity has a built in event system which I used to transfer information of my two different systems: a click (from orientation remaining still), and a head gesture (from shaking or nodding the head). The main component I built was what I called a HeadOrientationRayInteractor, which could be added to any object, usually the central camera, and would send a ray where the object was facing, that would perform the dwell controls mentioned earlier. If the ray stayed still long enough, it would see if the object had a HeadOrientationRayInteractable, which would then receive the click event and handle it.

For measuring the angles of the headset, I was able to just use the world orientation of the central camera, as it was directly tied to the headset. Thus, I could use the Euler angles from the camera's transform to make all the angle measurements and calculations detailed previously. Integrating the angular speed for head gesture detection was relatively straightforward too, just adding all the absolute differences in angles between each update frame.

For the head gestures I implemented 3: shaking based on yaw movement, nodding based on pitch movement, and bobbling based on roll movement. For my demonstration I ended up mapping them all to the same action of cycling through the different click action options there was, but for other projects these could each be mapped to different things which I will leave as an open question.

In my implementation there were also many variables that users could end up changing to best suit their desired experience. These include the time it takes to click when still, the angle tolerance to determine stillness, the minimum total angular distance a head gesture should take, the maximum start-to-end angular distance a head gesture can take, and the time it takes to detect a head gesture.

Additionally, I also implemented Minesweeper to be the demo application to interact with, as it the game supports 3 different types of clicks and making a wrong click can easily end the game. Please see the appendix for the full implementation and demo videos.

5 EVALUATION OF RESULTS AND LIMITATIONS

Evaluating user experience would ideally use a user study, but unfortunately I was the only user I had access to, so this will unfortunately have to be mostly qualitative. Furthermore, I lacked the means to compare head orientation control with the alternative eye gaze control, else I would have framed this section as a comparison between the two.

There were five potentially user-defined variables that my system depended on (the time it takes to click when still, the angle tolerance to determine stillness, the minimum total angular distance a head gesture should take, the maximum start-to-end angular distance a head gesture can take, and the time it takes to detect a head gesture), so I will discuss my findings in regards to each of these variables.

For the time it takes to click, I found that when I first started working with the control scheme, I was able to handle around 2 seconds, but soon that became too slow to use. Over the course of development, I was able to comfortably use it with a time of 0.5 seconds, but anything faster than that was hindering as random objects would click if I did not pay attention for a moment.

For the angle tolerance to determine stillness, my original approach was based on it being tied to visual angle, but upon using the original value of 0.7 degrees, it was too small and small movements of my head or body was enough to reset the timer. Thus, I now realize the angle needs to take into account the average angle that someone's head will move when idle, rather than the visual angle of a visually accessible object.

For the fixed time to detect a gesture, I found that 1 second was comfortable and consistent enough to get working. Anything quicker I was not really able to move my head back and forth quickly enough (without hurting), and anything slower was too annoying and had the possibility of detecting a false positive. The timer turned out to have more problems though, as detailed below.

For the total head orientation angle and start-to-end orientation angle thresholds for head gesture detection, I tested a variety of ranges. My original hypothesis that the total angle should be twice the start-to-end angle worked well enough and I surprisingly never saw a false positive during all my testing! I did, however, experience many failed detections, where I would shake my head a lot and get no detection. This seems to have been in part due to how the timer to detect a gesture was fixed and static, blocking subsequent attempts for it to trigger until the original one expired. As an improvement, the head gesture detection time should be more dynamic, ending early when possible, but also staying on until a head gesture is fully over. A more complex algorithm measuring the orientation data could likely achieve this, or perhaps even patterns detected my a ML algorithm. Overall, though, I tested 30 shakes, nods, and bobbles, and 27 were detected correctly while 3 were not detected at all, which is acceptable for a prototype (to me at least).

Overall, the control scheme I implemented was usable and not a bad user experience (to me, at least). Even with some failed detections on the head gestures, it worked well enough for me to finish many games of Minesweeper in my demo without frustration, and I am confident anyone who can move their head could play my demo successfully, too.

6 FUTURE WORK

Despite my implementation being mostly usable, I still think the current use of eye gaze is superior as it is far faster and natural to change what you are looking at than to rotate your head. However, there is one advantage a head orientation pointer has over eye gaze and it is that you can look somewhere other than where your pointer is. This could be particularly useful in gaming, where you might need to hold an interaction while looking somewhere else.

One far-fetched possibility is the use of the head orientation cursor and eye gaze cursor at the same time for two-point controls. It would have a crazy (and potentially painful) learning curve, but it could provide a handless control scheme that can zoom or scroll without the need to first go to the control menu.

As for head gestures, there is far more possibility than my three axis constrained gestures. There are so many cultures in the world with unique head movement gestures and each has the possibility to be mapped to a different user action. Future work done here would likely involve training a model to read a stream of orientation data and determine what parts of the stream correspond to what head gestures.

Ultimately, I think head orientation based cursors, dwell, and head gestures can have a place in accessible control schemes as an option or setting to be activated by the user to augment a more established control schemes with eye tracking.

7 CONCLUSION

Despite other control schemes like eye tracking being far more popular than head orientation control schemes, head orientation can have a large role to play in accessible controls, giving users yet another alternative to interact with VR and providing more possible handless gestures based on the movement of the head. While it may be inferior as a standalone control scheme, it might be able to be used in tandem with other control schemes to streamline user experience.

ACKNOWLEDGMENTS

I would like to thank Douglas Lanman and John Akers for providing the Quest 2 I used for development. I am also incredibly grateful for the opportunity to be part of this class and literally experience VR for the first time in my life. I originally thought VR was more of a fad but now I am sold and will buy a headset soon to play and develop on. Have a great spring break!

APPENDIX

Videos and Project links (please email mcnanik@cs.washington.edu if they aren't working). Try to download teaser video from google drive:

Teaser showing controls:
https://www.youtube.com/shorts/Mmnqu8SUezA
https://drive.google.com/file/d/1sYD_PHSRuTtlr5QI_aL-VPgP4NTdnp6X/view

Full demo, playing minesweeper with no hands!
https://www.youtube.com/shorts/b30RZ-GECaQ
https://drive.google.com/file/d/1aU1gEEOVxh6rQV70bXv_8NVpsERLz22l/view

Unity Project Repository (see the README for code):
https://gitlab.cs.washington.edu/mcnanik/cse-493-vr-project

REFERENCES

Apple. [n. d.]a. Control the Pointer Using Dwell on Mac. https://support.apple.com/guide/mac-help/control-the-pointer-using-dwell-mchl437b47b0/mac Last accessed

19 March 2025.

Apple. [n. d.]b. Use a pointer to navigate your Apple Vision Pro. https://support.apple.com/guide/apple-vision-pro/use-a-pointer-to-navigate-tan3869c8a85/2.0/visionos/2.0 Last accessed 19 March 2025.

Apple. [n. d.]c. Use Dwell Control on Vision Pro. https://support.apple.com/guide/apple-vision-pro/perform-actions-with-your-eyes-tan0ba69a1f1/visionos Last accessed 19 March 2025.

Cephable. [n. d.]. Products for Individuals. https://cephable.com/products/personal/for-individuals/ Last accessed 19 March 2025.

Cheng-Long Deng, Chen-Yu Tian, and Shu-Guang Kuai. 2022. A combination of eye-gaze and head-gaze interactions improves efficiency and user experience in an object positioning task in virtual environments. *Applied Ergonomics* 103 (2022), 103785. https://www.sciencedirect.com/science/article/pii/S0003687022001089

loves. 2022. VR Options for Quadriplegic with some Head Control. https://www.reddit.com/r/virtualreality/comments/o1qesm/vr_options_for_quadriplegic_with_some_head_control/

qpr1324. 2021. How about controlling the mouse cursor with VR head tracking? https://forums.flightsimulator.com/t/how-about-controlling-the-mouse-cursor-with-vr-head-tracking/376402

WCAG. 2024a. Target Size. https://www.w3.org/WAI/WCAG21/Understanding/target-size.html Last accessed 19 March 2025.

WCAG. 2024b. Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/#dfn-css-pixels Last accessed 19 March 2025.