AnyOrbit: Orbital navigation in virtual environments with eye-tracking

Benjamin I. Outram
Yun Suen Pai
benjamin.outram@gmail.com
yspai1412@gmail.com
Keio University Graduate School of Media Design

Tanner Person
Kouta Minamizawa
tannerperson@gmail.com
kouta@kmd.keio.ac.jp
Keio University Graduate School of Media Design

Kai Kunze
kai.kunze@gmail.com
Keio University Graduate School of Media Design

ABSTRACT

Gaze-based interactions promise to be fast, intuitive and effective in controlling virtual and augmented environments. Yet, there is still a lack of usable 3D navigation and observation techniques. In this work: 1) We introduce a highly advantageous orbital navigation technique, AnyOrbit, providing an intuitive and hands-free method of observation in virtual environments that uses eye-tracking to control the orbital center of movement; 2) The versatility of the technique is demonstrated with several control schemes and use-cases in virtual/augmented reality head-mounted-display and desktop setups, including observation of 3D astronomical data and spectator sports.

CCS CONCEPTS

• **Human-centered computing** → *Interaction design theory, concepts and paradigms*;

KEYWORDS

eye tracking, 3D user interface, 3D navigation, orbiting, orbital mode, virtual reality, augmented reality

ACM Reference Format:
Benjamin I. Outram, Yun Suen Pai, Tanner Person, Kouta Minamizawa, and Kai Kunze. 2018. AnyOrbit: Orbital navigation in virtual environments with eye-tracking. In *ETRA '18: 2018 Symposium on Eye Tracking Research and Applications, June 14–17, 2018, Warsaw, Poland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3204493.3204555

1 INTRODUCTION

With eye muscles being extremely fast and precise, gaze-based interactions seem a natural, efficient way for human-computer interaction tasks [Hansen et al. 2014; Majaranta and Bulling 2014]. Despite their potential, they are still not widely used. One huge issue is the high number of unintended activations, also known as Midas Touch problem [Jacob and Stellmach 2016]. Two other prevalent problems are eye fatigue and, especially for navigation tasks, simulator/motion sickness [Mardanbegi and Hansen 2011; Stellmach and Dachselt 2012].

Common solutions utilize eye gestures that are very different from everyday gaze, employ multimodal interactions or rely on smooth pursuit (the user follows a specific moving object on screen) [Esteves et al. 2015; Heikkilä and Räihä 2012; Meena et al. 2017]. Other innovative approaches use prediction algorithms to decide if a user wants to interact with or just see an object [Bednarik et al. 2012]. We present a novel gaze-based interaction in which we combine eye gaze with motion (in particular head motion) for navigation. It follows work from Mardanbegi et al. extending a combination of head and eye gaze interaction from just 1D volume control and paging [Mardanbegi et al. 2012] to unconstrained movement in 3D space. The user can look around naturally and will activate the gaze-based interaction as soon as he moves a body part (implemented on mouse and head movement).

Our approach uses an algorithm that positions the camera in virtual space on an orbital path around points of interest (POIs) which are selected with eye-tracking. Orbital motion is ubiquitous in computer-aided design (CAD) software systems, is instantly intuitive, and particularly suited to observational tasks [Khan et al. 2005; Ortega et al. 2015]. Perspective selection around an object can be achieved much faster and with less effort than conventional 'flying' metaphors, while maintaining the POI in sight at all times [Koller et al. 1996]. Head rotation was the preferred method out of several alternatives in a movement and observation task [Chung and Chung 1994]. Orbit-like motion is similar to fly-by camera shots in film and sports coverage and to strafe-and-shoot strategies in first-person-shooter (FPS) video games. In addition there is a tendency towards 3D and free-viewpoint video formats and technologies [Smolic et al. 2006].

We, therefore, propose that the use of orbital motion controlled by head-rotation and eye gaze can be exploited to provide a navigation strategy with several key advantages: 1) It allows continuous movement and 2) it is suitable to many use-cases in which orbital and sideways type motion is already employed.

Here we describe AnyOrbit, which exploits the geometry of toroids to allow ideal spiral paths to orbits around new POIs, allowing 6 degree-of-freedom (DOF) movement. Using eye-tracking, the POI is determined by the user's gaze. Finally, we explore a variety of use cases, which indicate that the use of eye-tracking for controlling the center of rotation provides a powerful technique for navigation and observation.

ETRA '18, June 14–17, 2018, Warsaw, Poland

ACM ISBN 978-1-4503-5706-7/18/06…$15.00
https://doi.org/10.1145/3204493.3204555

2 ANYORBIT

Orbital techniques generally provide 3DOF of movement (two orbital directions and one radial direction), which limits the user's view to only the radial direction in towards the center. Orbital techniques are often then combined with flying, panning and other 'modes' that can be switched between [Fitzmaurice et al. 2008; Tan et al. 2001; Zeleznik and Forsberg 1999] to allow other types of movement and 6DOF navigation. These methods work well for desktop CAD applications, but within an immersive virtual environment (VE) context, panning and other types of motion are known to cause excessive motion sickness [Chen et al. 2011; Pausch et al. 1993; Psotka 1995; Stanney and Hash 1998].

To overcome this problem, Koller *et al.* suggests allowing the user to switch between traditional ego-centered rotation, and orbital viewing modes [Koller et al. 1996]. While in the ego-centered view, the user can select an object of interest to become the orbital center using a peripheral input method such as a control stick. The user is then teleported to a location determined by their current rotation angle, the current radius between the user and the object, and the new selected center of orbit, or alternatively the user is locked such that their forward-facing vector is no longer facing the center of the orbital motion. The problem is that teleportation is sometimes disorientating, and facing towards any direction other than towards the center of rotation is uncomfortable.

AnyOrbit solves these problems by providing an all-in-one mode that can reproduce both ego-centered and orbit-like movement at appropriate times. For example, if the POI is in the peripheral vision, the user's intent is likely to look towards the POI in an ego-centered fashion, before choosing to orbit around it.

With AnyOrbit, we automate this process by taking the relative position of the POI in the field of view (FOV) as an input to determine the type of motion. We can think of ego-centered and orbital modes as being two ends of a continuum of behaviour between cases where the radius of curvature of the user's motion is zero (ego-centered rotation), and where it is equal to the distance to the POI (purely orbital). AnyOrbit alters the radius of movement on the fly depending on where a POI is in the FOV of the user. If the POI is in the peripheries of the FOV, we shorten the user's movement radius, which allows the user to first turn towards the POI, such that the POI moves towards the center of their FOV. As the POI approaches the center of the FOV, we gradually lengthen the radius such that the user finds themselves on an orbital path looking in towards the POI. The result is a natural and intuitive way of looking and orbiting around VEs, in which the user in general moves on spiral arcs between orbital paths about a changing POI.

Another related work, GazeSphere, switched between ego-centric and orbit-like motion along predefined linear paths by gazing at specific POIs in 360-video [Pai et al. 2017]. AnyOrbit extends this interaction to 3-dimensional movement in 3D environments, by allowing the user free choice of POI and by improving the transitions between ego-centered and orbital rotation.

Figure 1 (a) illustrates how the orbital motion resulting from such a process creates a smooth outward spiral trajectory from the current location and orientation towards a circular orbital trajectory with the new marker location at the center. In the reverse case, in which the user rotates their head away from the marker, the radius

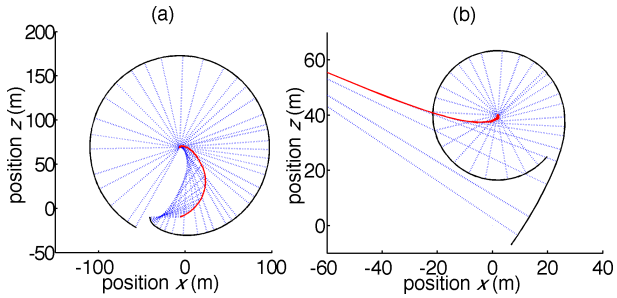

Figure 1: Illustrates the smooth path taken from one position and orientation to a position and orientation about a new orbital center. (a) and (b) show cases in which the user rotates towards and away from a new POI respectively. The black line indicates the path taken by the user, the red line indicates the path taken by the dynamically controlled orbital center, and the dotted blue lines indicate the user's forward-facing vector and the radius at that moment. By controlling the radius dynamically the user's path spirals towards the new orbital path.

is extended with the result that, again, their trajectory smoothly transitions, this time via an inward spiral trajectory towards a circular orbit with the marker at the center, as illustrated in 1 (b).

Since the marker could in general be at any horizontal and vertical position in the FOV, we would like to independently control both horizontal and vertical orbital curvatures of rotation. A spherical orbit is problematic because it has the same curvature in both horizontal and vertical directions. Therefore, we calculate the motion on a torus, whose radii are in general different in horizontal and vertical directions. By calculating the torus surface based on the current relative positions of both the user and the orbital marker, we can thus allow the user to smoothly move from orbit about one center to an orbit about any POI in the user's FOV. Whichever way the user turns will produce an optimal smooth path to any perspective they choose about any new POI. Next, we outline how this process is implemented in an algorithm to calculate the new position of the user in the 3D environment at each frame.

The technical implementation is as follows (code is available on GitHub [Outram 2018]). A new position $\mathbf{P}$ in each frame is calculated based on the following initial parameters: $\mathbf{P}_0$ the position of the user in the last frame; the azimuthal, ϕ_0, and zenithal θ_0 angles defining the orientation of the user in the last frame; $\mathbf{M}$ the position of the orbital marker relative to the user and; the current orientation of the user, ϕ_1 and θ_1. In addition, the fixed parameter a controls the pace at which a rotation will cause the orbital distance to reach an equilibrium with the new orbital center (we use $a = 2$).

In the following, x, y and z refer to left-to-right horizontal, down-to-up, and straight-outward local directions relative to the user's current orientation (ignoring tilt about the z axis), and X, Y and Z are right-handed world coordinates with Y in the upward direction. In addition r_x and r_y refer to radii of the movement curvature in x and y directions. The algorithm for calculating the user's current position is as follows:

(1) Determine whether the user's head movement relative to the last frame is towards or away from the orbital marker in the x direction.
(2) Calculate the radius of curvature in the x direction r_x: In the case that the user is rotating towards the marker,

$$r_x^{\text{towards}} = M_z - a|M_x| \tag{1}$$

where M_z and M_x are components of $\mathbf{M}$ in z and x directions. In the case that the user is rotating away from the marker,

$$r_x^{\text{away}} = M_z^2 / r_x^{\text{towards}} \tag{2}$$

We also constrain r_x such that $0.2 < r_x/M_x < 5$ to limit the maximum velocity and remove large accelerations.
(3) Repeat steps 1 and 2 for the y direction.
(4) Find the center of the torus on which we wish to move. In the case that $r_x > r_y$, we can consider a position $\mathbf{T}(\theta, \phi, r, R)$ on the surface of a torus whose symmetry axis is along Y and whose center is at the origin, defined by

$$\begin{aligned} T_x &= -\big(R + r\cos(\theta)\big)\sin\phi \\ T_y &= r\sin\theta \\ T_z &= -\big(R + r\cos(\theta)\big)\cos\phi \end{aligned}$$

where T_x, T_y and T_z are the components of $\mathbf{T}$ in X, Y and Z-axes, r and R are the torus minor and major radii, and θ and ϕ are zenithal and azimuthal angles relative to world coordinates. The center of the torus on which we wish to move is thus given by

$$\mathbf{T}_0 = \mathbf{P}_0 - \mathbf{T}(\theta, \phi, r, R) \tag{3}$$

with $\theta = \theta_0$, $\phi = \phi_0$, $r = r_y$ and $R = r_x - r_y$.
(5) Finally, calculate the new position, which is given by,

$$\mathbf{P} = \mathbf{T}(\theta, \phi, r, R) + \mathbf{T}_0 \tag{4}$$

this time with $\theta = \theta_1$, $\phi = \phi_1$, and again with $r = r_y$ and $R = r_x - r_y$.
(6) In the case that $r_y > r_x$, follow a similar process as in steps 4 and 5, but instead consider a torus whose axis of symmetry is in the horizontal x axis of the previous frame. Since the torus is perpendicular to the forward-facing direction of the last position and orientation defined by ϕ_0 and θ_0, the center can be trivially found by extending this vector direction out from $\mathbf{P}_0$ by a distance of r_y. Then for step 5, define a torus in world coordinates with symmetry axis along X, substitute $\theta = \theta_1$, $\phi = \phi_1 - \phi_0$ and rotate the resultant $\mathbf{T}$ about the origin by ϕ_0.

It is helpful for user control that the marker always be not too distant and in most cases visible [Fitzmaurice et al. 2008], and so we recommend limiting its position, depending on the environment context, to for example $M_z < 100$m. If the marker is outside the FOV, we constrain $r_y = r_x = 0$, i.e. egocentric rotation.

If using a head-tracked HMD, the 3 rotational DOF are sufficient as input to AnyOrbit, allowing it to be used with 3DOF mobile VR headsets. If available, the extra 3 translational DOF could be ignored, but we found it more comfortable to allow the user free translational movement relative to the orbital center. To achieve this, record the translational movement of the camera since the last frame and add it to the position in step 5.

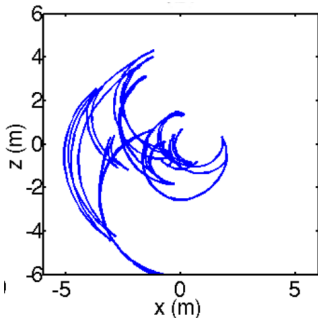

Figure 2: Shows position data of the user in the VE while using AnyOrbit, as viewed from above, using HMD with directed movement control.

3 CONTROL SCHEMES

There are 4 control inputs of the AnyOrbit system: zenithal (pitch) and azimuthal (yaw) angles, POI (desired orbital center) and desired orbit radius. Here, the angles are coupled to the corresponding head rotation angles as in previous work [Chung and Chung 1994; Koller et al. 1996]. Control of the POI and orbital radius can be achieved with mouse, eye-tracking, or chosen by a director. A previous study using a mouse input with AnyOrbit enabled effective navigation to new viewpoints at a rate comparable to changes in perspective in broadcast sport, and did not cause significant increases in simulator sickness after the first 5 minutes [Outram et al. 2016]. Unique to this work, we outline the technical implementation, and report on directed control and the use of eye-tracking, which have the added benefit of allowing hands-free interaction. Table 1 summarises the control configurations, and next we describe user experience.

3.1 HMD Directed: Head Rotation with Directed Position and Radius Control

3D film and storytelling often has the problem that the director cannot control which direction the user is looking. StyleCam proposed a solution in which navigational control is shared between the user and the content producer to direct the user experience [Burtnyk et al. 2002]. However with AnyOrbit, the director can control the POI that the user is facing, while simultaneously giving full rotational control to the user.

We created a VE consisting of a sample of 27,000 stars taken from the HYG Database [Nash 2011]. Figure 3 shows the VE, in which the stars' positions, colours, brightnesses and velocities are rendered using aesthetically chosen scaling factors. In this case, we predefined the desired POI and orbital radius. Once a user wants to move on, they can trigger a change to the next predetermined POI and radius by aligning the current orbital center with a designated point. User position data is shown in Figure 2. The predetermined POIs and radii were selected to give a variety of perspectives, from both within the field of stars, and looking in from outside. Navigation and observation were reported to be instantly intuitive, with one user remarking that it *"feels very dramatic and gives a heightened sense of perspective"*. An ideal use-case would be of consuming sport/e-sport recorded in 3D. Broadcasters can direct the user's attention while the user remains in control of their rotation and perspective.

Table 1: Various control schemes we have tried using AnyOrbit.

	AnyOrbit Parameter Control Schemes			
Configuration Name	*Pitch*	*Yaw*	*Desired orbit radius*	*Desired orbit center*
HMD Directed	head pitch	head yaw	Chosen by director	Chosen by director
Desktop Eye-gaze	mouse y	mouse x	Fixed	Eye-gaze x,y
HMD Eye-gaze	head pitch	head yaw	Fixed	Eye-gaze x,y

3.2 Desktop Eye-gaze: Mouse Rotation with Eye Gaze Position Control

Here we explored the use of AnyOrbit in a desktop environment. The desired orbital radius was kept fixed, while the desired orbital center was placed at a distance from the user equal to the fixed desired orbital radius, with the x-y position of the marker determined by the x-y on the screen of the user's gaze. A Tobii EyeX eye tracker was mounted at the bottom of the computer monitor facing the user. After a brief calibration, the eye-tracking data is sent to the Unity environment via a Tobii plugin, allowing the eye-gaze data to be used to control the POI marker.

To test the control scheme, we developed a simple environment consisting of a ground plane with two cubes on the surface separated by some distance (see Figure 3). The user can navigate to orbital paths around either of the cubes, by simply tracking the cubes with their eyes as they rotated the camera with the mouse. The technique felt intuitive, and the quality of the POI following the eye gaze position felt like magic. Users preferred not to have a visible marker that followed their gaze, saying it was too distracting.

3.3 HMD Eye-gaze: Head Rotation with Eye-Gaze Position Control

As with the "Desktop Eye-gaze" example, we fixed the orbital radius and used eye gaze position to control the x-y position of the AnyOrbit POI. In this case however, the environment was experienced through an HMD and camera angle was coupled to head rotation angle. PupilLabs eye-tracking technology was installed into an HTC Vive headset, which required a short calibration task before entering the VE. We tested the same star-field VE as in the "HMD Directed". As in the desktop eye-gaze scheme, users preferred it when the orbital marker was not visible.

Leveraging eye-gaze to control the POI marker not only frees up the hands for non-navigation specific controls, but may also further reduce simulator sickness. With mouse control of POI marker, the user could be looking at a part of the VE in the foreground of the orbit center, in which the visual-field optical flow is opposite to what would be normally expected. From previous research [Stanney and Hash 1998], and from our experience, there is reason to believe this could be the cause of increased simulator sickness. If the POI marker is controlled by the eye-gaze, such a situation is avoided.

Indeed, we have only began to explore eye-gaze's use with AnyOrbit, but our implementation points to intriguing possibilities. It feels like the world anticipates your movement intentions, and eye-gaze control may heighten a sense of immersion. It can also help for motor-impaired users [Jankowski and Hachet 2015].

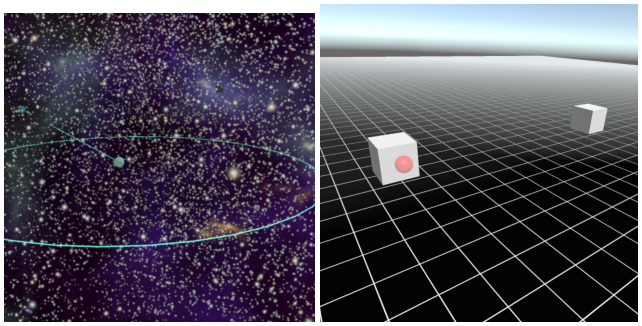

Figure 3: VEs used in our implementation. Top: User perspective of a VE consisting of stars. The user is guided on an orbital viewing path to different POIs (see accompanying video). The green object in the center identifies the POI and the circle marks the desired radius. Aligning the POI with the distant object advances to the next POI. Bottom: One of the VEs used for demonstrating eye-gaze control of orbital center. The user can orbit around either of the boxes by looking at them and navigate on a smooth trajectory.

4 CONCLUSION

We have described the design and implementation of AnyOrbit, a technique for hands-free orbital navigation around and between POIs selected by the user's eye gaze. Several different POI control schemes were tested, including user control through eye gaze, and also directed control. Directed control was found to be instantly intuitive and gives POI control to the director while not compromising on user's freedom to rotate. Control using eye-tracking was effective in both desktop and HMD scenarios, and allowed surprisingly intuitive hands free navigation.

A limitation of AnyOrbit is that it is easy for users to travel through virtual objects, which is known to be disorientating. Mitigation strategies exist which may be applied to our system [Fitzmaurice et al. 2008; Mackinlay et al. 1990; Phillips et al. 1992]. In particular, the orbital radius could be shortened as a user approaches an object, reducing their velocity and steering them away from virtual objects. Simulator sickness also remains a limitation of our system. FOV restrictors could mitigate this problem [Fernandes and Feiner 2016].

The technique potentially leads to new types of interactive media experience, and can be applied widely to CAD systems, sports and e-Sports, 3D recorded media, data visualisation and games.

REFERENCES

Roman Bednarik, Hana Vrzakova, and Michal Hradis. 2012. What Do You Want to Do Next: A Novel Approach for Intent Prediction in Gaze-based Interaction. In

Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12). ACM, New York, NY, USA, 83–90. https://doi.org/10.1145/2168556.2168569

Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. 2002. StyleCam: interactive stylized 3D navigation using integrated spatial & temporal controls. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*. ACM, 101–110.

Wanjun Chen, JZ Chen, and Richard Hau Yue So. 2011. Visually induced motion sickness: effects of translational visual motion along different axes. *Contemporary Ergonomics and Human Factors* (2011), 281–287.

James Che-Ming Chung and James C Chung. 1994. Intuitive navigation in the targeting of radiation therapy treatment beams. (1994).

Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 457–466.

Ajoy S Fernandes and Steven K Feiner. 2016. Combating VR sickness through subtle dynamic field-of-view modification. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on*. IEEE, 201–210.

George Fitzmaurice, Justin Matejka, Igor Mordatch, Azam Khan, and Gordon Kurtenbach. 2008. Safe 3D navigation. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. ACM, 7–15.

John Paulin Hansen, Alexandre Alapetite, I Scott MacKenzie, and Emilie Møllenbach. 2014. The use of gaze to control drones. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 27–34.

Henna Heikkilä and Kari-Jouko Räihä. 2012. Simple gaze gestures and the closure of the eyes as an interaction technique. In *Proceedings of the symposium on eye tracking research and applications*. ACM, 147–154.

Rob Jacob and Sophie Stellmach. 2016. What you look at is what you get: gaze-based user interfaces. *interactions* 23, 5 (2016), 62–65.

Jacek Jankowski and Martin Hachet. 2015. Advances in interaction with 3D environments. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 152–190.

Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. 2005. Hovercam: interactive 3D navigation for proximal object inspection. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 73–80.

David R Koller, Mark R Mine, and Scott E Hudson. 1996. Head-tracked orbital viewing: an interaction technique for immersive virtual environments. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*. ACM, 81–82.

Jock D Mackinlay, Stuart K Card, and George G Robertson. 1990. Rapid controlled movement through a virtual 3D workspace. In *ACM SIGGRAPH Computer Graphics*, Vol. 24. ACM, 171–176.

Päivi Majaranta and Andreas Bulling. 2014. Eye tracking and eye-based human–computer interaction. In *Advances in physiological computing*. Springer, 39–65.

Diako Mardanbegi and Dan Witzner Hansen. 2011. Mobile Gaze-based Screen Interaction in 3D Environments. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications (NGCA '11)*. ACM, New York, NY, USA, Article 2, 4 pages. https://doi.org/10.1145/1983302.1983304

Diako Mardanbegi, Dan Witzner Hansen, and Thomas Pederson. 2012. Eye-based head gestures. In *Proceedings of the symposium on eye tracking research and applications*. ACM, 139–146.

Yogesh Kumar Meena, Hubert Cecotti, KongFatt Wong-Lin, and Girijesh Prasad. 2017. A multimodal interface to resolve the Midas-Touch problem in gaze controlled wheelchair. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*. IEEE, 905–908.

David Nash. 2011. The Hyg Database. *URL: https://github. com/astronexus/HYG-database* (2011).

Michael Ortega, Wolfgang Stuerzlinger, and Doug Scheurich. 2015. SHOCam: A 3D Orbiting Algorithm. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 119–128.

Benjamin I Outram. 2018. AnyOrbit Unity3D demos and code implementation GitHub repository. (2018). Retrieved April 18, 2018 from https://github.com/bio998/AnyOrbit

Benjamin I Outram, Yun Suen Pai, Kevin Fan, Kouta Minamizawa, and Kai Kunze. 2016. AnyOrbit: Fluid 6DOF Spatial Navigation of Virtual Environments using Orbital Motion. In *Proceedings of the 2016 Symposium on Spatial User Interaction*. ACM, 199–199.

Yun Suen Pai, Benjamin I Outram, Benjamin Tag, Megumi Isogai, Daisuke Ochi, and Kai Kunze. 2017. GazeSphere: navigating 360-degree-video environments in VR using head rotation and eye gaze. In *ACM SIGGRAPH 2017 Posters*. ACM, 23.

Randy Pausch, M Anne Shackelford, and Dennis Proffitt. 1993. A user study comparing head-mounted and stationary displays. In *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*. IEEE, 41–45.

Cary B Phillips, Norman I Badler, and John Granieri. 1992. Automatic viewing control for 3D direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics*. ACM, 71–74.

Joseph Psotka. 1995. Immersive training systems: Virtual reality and education and training. *Instructional science* 23, 5-6 (1995), 405–431.

Aljoscha Smolic, Karsten Mueller, Philipp Merkle, Christoph Fehn, Peter Kauff, Peter Eisert, and Thomas Wiegand. 2006. 3D video and free viewpoint video-technologies, applications and MPEG standards. In *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2161–2164.

Kay M Stanney and Phillip Hash. 1998. Locus of user-initiated control in virtual environments: Influences on cybersickness. *Presence: Teleoperators and Virtual Environments* 7, 5 (1998), 447–459.

Sophie Stellmach and Raimund Dachselt. 2012. Investigating gaze-supported multimodal pan and zoom. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 357–360.

Desney S Tan, George G Robertson, and Mary Czerwinski. 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 418–425.

Robert Zeleznik and Andrew Forsberg. 1999. UniCam—2D gestural camera controls for 3D environments. In *Proceedings of the 1999 symposium on Interactive 3D graphics*. ACM, 169–173.