Chapter 4: Eye-tracking in Virtual Reality

Authors: Nicola C. Anderson (corresponding author) (University of British Columbia, Vancouver, Canada; nicola.anderson@ubc.canc), Walter F. Bischof (University of British Columbia, Vancouver, Canada; wfb@ualberta.ca), & Alan Kingstone (University of British Columbia, Vancouver, Canada; alan.kingstone@ubc.ca)

Abstract

This chapter explores the current state of the art in eye-tracking within 3D virtual environments. It begins with the motivation for eye tracking in Virtual Reality (VR) in psychological research, followed by descriptions of the hardware and software used for presenting virtual environments as well as for tracking eye and head movements in VR. This is followed by a detailed description of an example project on eye and head tracking while observers look at 360° panoramic scenes. The example is illustrated with descriptions of the user interface and program excerpts to show the measurement of eye and head movements in VR. The chapter continues with fundamentals of data analysis, in particular methods for the determination of fixations and saccades when viewing spherical displays. We then extend these methodological considerations to determining the spatial and temporal coordination of the eyes and head in VR perception. The chapter concludes with a discussion of outstanding problems and future directions for conducting eye- and head-tracking research in VR. We hope that this chapter will serve as a primer for those intending to implement VR eye tracking in their own research.

1 Why track eyes in VR?

VR is a good design choice for human performance experiments for a number of reasons, but tracking the eyes in VR provides several key advantages over traditional computer-based or mobile eye tracking research that we touch on throughout this chapter. Most notably, VR eye tracking allows for the simultaneous tracking of the eyes and other head and body movements with respect to a common reference frame (see Section 5.1), allowing for the precise calculation and dissociation of the relative contributions of these different movements to more general attentional control (see Section 6). Note that throughout this chapter, we use the terms looking, eyes, and gaze synonymously.

Much of what we know about visual attention and eye movement control is derived from studies that require people to look at images presented on a computer monitor while their head is restrained. There is, however, growing recognition that eye movements measured in the lab when an observer's head movements are discouraged or restrained are not representative of how people move their eyes in everyday life when their head is free to move (e.g., Backhaus et al., 2020; Hooge et al., 2019; Kingstone et al., 2008; Land & Hayhoe, 2001; Risko et al., 2016; ’t Hart et al., 2009). For example, Foulsham, Walker, and Kingstone (2011) asked participants to watch first-person video clips of someone walking across campus. While there was some bias to look in the center of the video, their gaze (i.e., the direction of their eyes) was spread over the whole scene, looking at objects and the people that the walker encountered. When the same participants physically walked across campus with a mobile eye tracker, they often focused on the path, and the eyes remained relatively centered in the visual field as defined by their head orientation (Foulsham & Kingstone, 2017). In other words, when the head was free to move, people tended to move their head in order to redirect their gaze to objects and people. In the lab, Solman and colleagues (2017) have shown that when participants are required to look at a scene through an asymmetric window that is yoked to their eyes, eye movements target regions within the window. However, when the window is yoked to an observer's head movements, the head moves to reveal new information outside the window, presumably so that the eyes can then examine visual information within the new window. These studies suggest that when people are allowed to move their head, they do so, and that the head acts to reveal new information for the eyes to exploit.

In the analysis of head movements, we have learned that the relative timing of eye and head movements may suggest whether attentional selection is reflexive or volitional (Doshi & Trivedi, 2012; Freedman, 2008; Zangemeister & Stark, 1982). For instance, when the eyes move before the head, these are unplanned, reflexive movements (usually to a suddenly presented stimulus such as a flash of lightning) and involve shifts of less than 45° (Barnes, 1979; Flindall et al., 2021). When the head leads the eyes, however, these are thought to be large, planned, purposeful movements, often to a known target location. These conclusions are mainly based on experiments where participants respond to simple light displays or targets on a screen but it has also been shown in more naturalistic settings, where, for example, when we prepare to cross a street or when we prepare a lane change while driving (Doshi & Trivedi, 2012).

In VR, when people are asked to view scenes in 360°, the attention system must coordinate eye movements with other head and body movements to explore the full range of the visual field (if this is desired). When looking at 360° panoramic scenes, observers spend the majority of the time exploring along the horizon (Rai et al., 2017; Sitzmann et al., 2018), using their head and other body movements to extend the field of view for the eyes (Bischof et al., 2020). When viewing landscape and fractal panoramic scenes that are rotated (for example, 45° clockwise), the head tends to rotate in a similar manner in order to bring the scenes closer to their canonical upright position for the eyes (Bischof et al., 2020), converging with other evidence suggesting that the head acts in service of the eyes to extend the range of possible viewable locations (Solman et al., 2017; Solman & Kingstone, 2014).

On the other hand, studies in VR have taught us that the eyes, head (and body) may move in ways that diverge from what we might expect. When observers are asked to search 3D environments, the effective field of view, or visual span, is much larger than reported in studies using smaller images on a computer monitor (David et al., 2021). In a study where observers viewed large, flat (i.e., non-panoramic) landscape and fractal scenes, it was found that the head and eyes responded differently to scene rotations, where the eyes seemed to be more sensitive to rotation than the head (Anderson et al., 2020). In other work with panoramic scenes, it has been shown that the head is less affected by the masking of central or peripheral scene information than the eyes (David et al., 2022). In addition, in many of these works, the extent that participants move their head (and body) varies largely between individuals (Anderson et al., 2020; Bischof et al., 2020; Jacobs et al., 2020). This variation mirrors earlier observations by researchers who looked at the extent that observers prefer to recruit their head in response to peripheral target acquisition – resulting in some observers being dubbed "movers" and others "non-movers" (e.g., Delreux et al., 1991; Fuller, 1992a, 1992b; Goldring et al., 1996; Stahl, 2001). Taken together, these works provide varying degrees of evidence that head and eye movements may diverge in their control strategies, leading researchers to speculate that the head may be under more deliberate, cognitive control (David et al., 2022), or utilize different spatial reference frames (Anderson et al., 2020). Importantly, these and other studies in VR have uncovered interesting findings and generated novel questions about the complex dynamics between eye, head, and other movements in fields of view that extend beyond the standard computer monitor.

2 VR and Eye Tracking – Hardware

In this section we review major hardware and software used for eye tracking in VR. We first present the hardware used or stimulus presentation, namely head-mounted displays and projection displays, followed by the presentation of eye-tracker and head-tracker hardware.

2.1 Stimulus Hardware

VR stimuli can be presented in one of two major configurations, head-mounted displays and projection displays. In *head-mounted displays* (HMDs), stimuli are presented using a head-mounted viewer with two displays, one for each eye (see Figure 1). An eye tracker can be mounted inside the viewer, and the position and orientation of the HMD can be tracked with multiple methods dependent on the headset used.

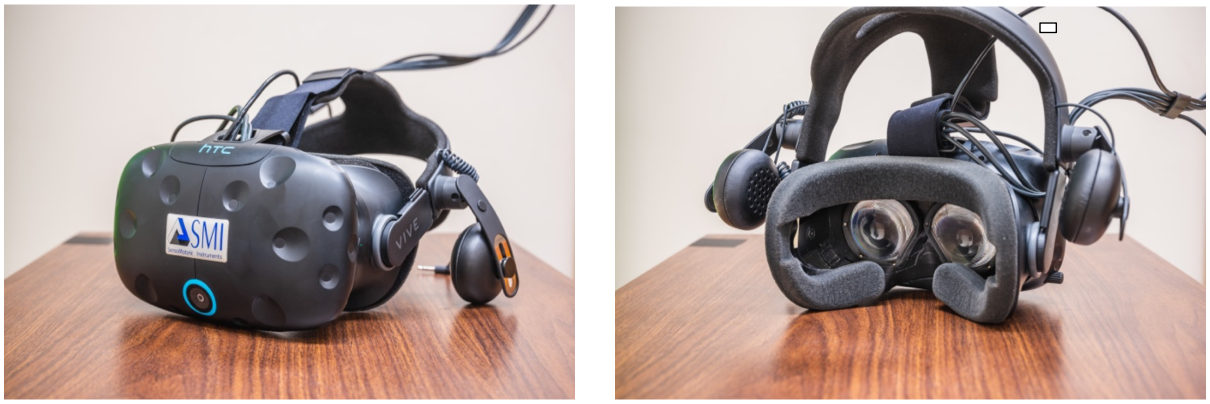


Figure 1: (left) Outside view of HTC Vive HMD. (right) Inside view of HTC Vive HMD with SMI eye tracker. Photos taken by Jacob Gerlofs.

When setting up a VR eye tracking lab, some consideration needs to be taken to how participants physically interact with the headset. Most headsets with built-in eye tracking are (to date) still tethered to a computer via a cable (as in the HTC Vive). In our experience, participants are aware of any cables from the headset or response apparatus (i.e., keyboard) that are attached to the computer. This may affect their ability or tendency to move freely in the VR environment (if this is desired). For example, in recent published and unpublished work (Bischof et al., 2020; Jacobs et al., 2020) where participants responded on a tethered keyboard while looking at 360° scenes, participants rarely, if ever, rotated the full 360°, despite being seated in a swivel chair. One way to mitigate this is to mount the cables to the ceiling and use VR controllers or wireless keyboards for manual responses. It remains an interesting open question whether it makes a difference in how participants attend to VR scenes if they are seated (as in the vast majority of our studies) or standing (as in, for example, David et al., 2020, 2021).

In *projection displays*, sometimes referred to as caves, the stimuli are projected onto 1 - 6 walls with the observer standing in the middle (e.g., *Visbox, Inc.*, n.d.). Eye movements of observers can be recorded using a remote eye tracker attached to a screen or using a head-mounted eye tracker. In the latter case, head movements need to be tracked to convert gaze direction in head-centered coordinates into an environmental stimulus-centered (i.e., allocentric) framework. Alternatively, the gaze direction can be converted to allocentric coordinates using markers attached to the walls of the cave.

2.2 Eye Tracker Hardware

The goal of eye tracking is often to determine the direction of gaze in the allocentric coordinate system of the presented virtual world. This chapter focuses on the use of head-mounted displays for VR. In this case, gaze determination involves two parts: determining gaze direction in head-centered coordinates (e.g., *Pupil Invisible - Eye Tracking Glasses for the Real World - Pupil Labs*, n.d.; Sensomotoric, 2017) and determining the position and orientation of the head using a head tracker (see below) or via scene camera motion analytics (Ohayon & Rivlin, 2006).

2.3 Head Tracker Hardware

Several systems are available for tracking the head position and direction. In most commercial VR systems (e.g., the HTC Vive), the motion tracker is a system component. In other systems like those using projection displays, this must be achieved using an independent head tracker, for example with visual sensing (*V120*, n.d.) or with an inertial tracker composed of accelerometers and gyroscopes (*Blue Trident IMU | Inertial Sensor by Vicon | Biomechanic Tracking*, n.d.).

3 VR and Eye Tracking – Software

There are three major software systems for creating VR worlds, Unity, Unreal Engine, and Vizard. We review each system briefly, but since our own experience is based in Unity, we later provide a detailed example using that system. Not every VR eye tracker is compatible with each of these systems (e.g., Pupil Labs does not yet have a plugin for Unreal Engine), so it is worth making note of the availability and support for different software prior to purchasing any VR eye tracking hardware.

3.1 Unity

Unity is a game engine developed by Unity Technologies in 2005. The engine has been extended to support a large range of platforms, including desktops, mobiles, and VR platforms. It is very popular for mobile game development and used for games such as Pokémon Go or Call of Duty. It is considered easy to use for beginners and can be used for creating 3D and 2D worlds. Creating a virtual world consists of setting the world up using an interactive development environment and expanding and controlling it using C# or Java scripting language.

There are a number of toolboxes that help in the development of Unity-based experiments, such as UXF (Brookes et al., 2020), USE (Watson et al., 2019), bmlTUX (Bebko & Troje, 2020), and VREX (Vasser et al., 2017). Later in this chapter we present a basic and straightforward approach to developing experiments with Unity that focuses on the measurement and use of eye and head movements in experiments (see Section 4). Nevertheless, these could potentially be used in conjunction with any of the above tools.

3.2 Unreal Engine

Unreal Engine is a game engine developed by Epic Games in 1998. The engine was first developed for first-person shooter games, but has since been used in an expanding range of 3D games and has been adopted by the film and television industry. The unreal engine is scripted in C++ and supports a wide range of desktop, mobile, console and virtual reality platforms. The Unreal Engine has only limited support for popular eye trackers, for example the Tobii or the SMI eye trackers.

3.3 Vizard

Worldviz has developed a Python toolkit for developing VR applications, which supports multiple stimulus devices, eye and body trackers, input devices, such as e.g., gloves, haptic devices, and controllers (*Vizard | Virtual Reality Software for Researchers*, n.d.). Figure 2 shows a small example of a Vizard program, which includes setting up a VivePro eye tracker.

```
import viz
import vizact

viz.setMultiSample(4)
viz.fov(60)
viz.go()
# Set up the VivePro eye tracker
VivePro = viz.add('VivePro.dle')
eyeTracker = VivePro.addEyeTracker()
# Create an empty array to put some pigeons in.
pigeons = []
# Go through a loop six times.
for eachnumber in range(6):
    # Create pigeon
    newPigeon = viz.addAvatar('pigeon.cfg')
    # Place the new pigeon on the x-axis.
    newPigeon.setPosition([eachnumber, 0, 5])
    #Add the new pigeon to the "pigeons" array.
    pigeons.append(newPigeon)
# Move the view to see all pigeons
viz.MainView.move([2.5,-1.5,1]
# Get gaze matrix in local HMD coordinate system
gazeMat = eyeTracker.getMatrix()
# Transform gaze matrix into world coordinate system using main view
gazeMat.postMult(viz.MainView.getMatrix())
# Intersect world gaze vector with scene
line = gazeMat.getLineForward(1000)
info = viz.intersect(line.begin, line.end)
if info.valid:
    print('User is looking at', info.point)
```

Figure 2: Simple Vizard example program

3.4 Stimuli

One of the primary advantages of VR is the flexibility of the virtual world, where participants can be immersed in a fully rendered 3D replication of the world to any number of more simplified (or fantastical) environments. This has some implications for eye tracking that are worth noting. For example, in some of our recent work, we were interested in how image (and screen) rotation might have affected saccade direction biases. In traditional, desktop-based setups, it has been shown that rotating scenes affects the predominance of horizonal and vertical saccades (Foulsham et al., 2008; Foulsham & Kingstone, 2010). However, in VR, it is possible to not only rotate the image itself, but also the entire screen it is projected on – a real-world equivalent would be rotating the entire computer monitor, not just the image content (Anderson et al., 2020). In this case, we were interested in how observers looked at the scenes themselves (and not so much whether they looked at the rest of the virtual scene). Therefore, eye movements were reported based on their 2D position on the plane in the virtual world, very similar to how they would be reported in a traditional, computer-monitor-based eye tracking study (see Figure 3).

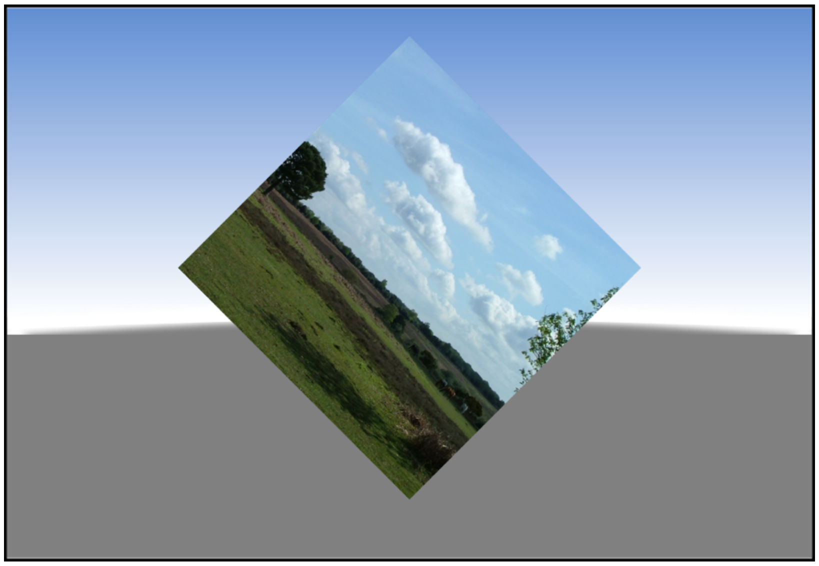


Figure 3: Example participant view of a rotated scene in Anderson et al. (2020).

In more complicated situations, researchers might be interested in how participants search for objects in a fully rendered 3D scene (e.g., David et al., 2021). In this case, the researchers were interested in the objects the participants looked at, as well as general search measures such as scanning, verification, and search time. For fully immersive 360° panoramic scenes, gaze position might be reported with respect to the scene sphere. Each of these scenarios has different demands from the data processing and analysis that need to be kept in mind. We discuss the reference frames commonly used in VR below (section 5.1).

4 Eye and Head Movements in 360° Scenes – an Example in Unity

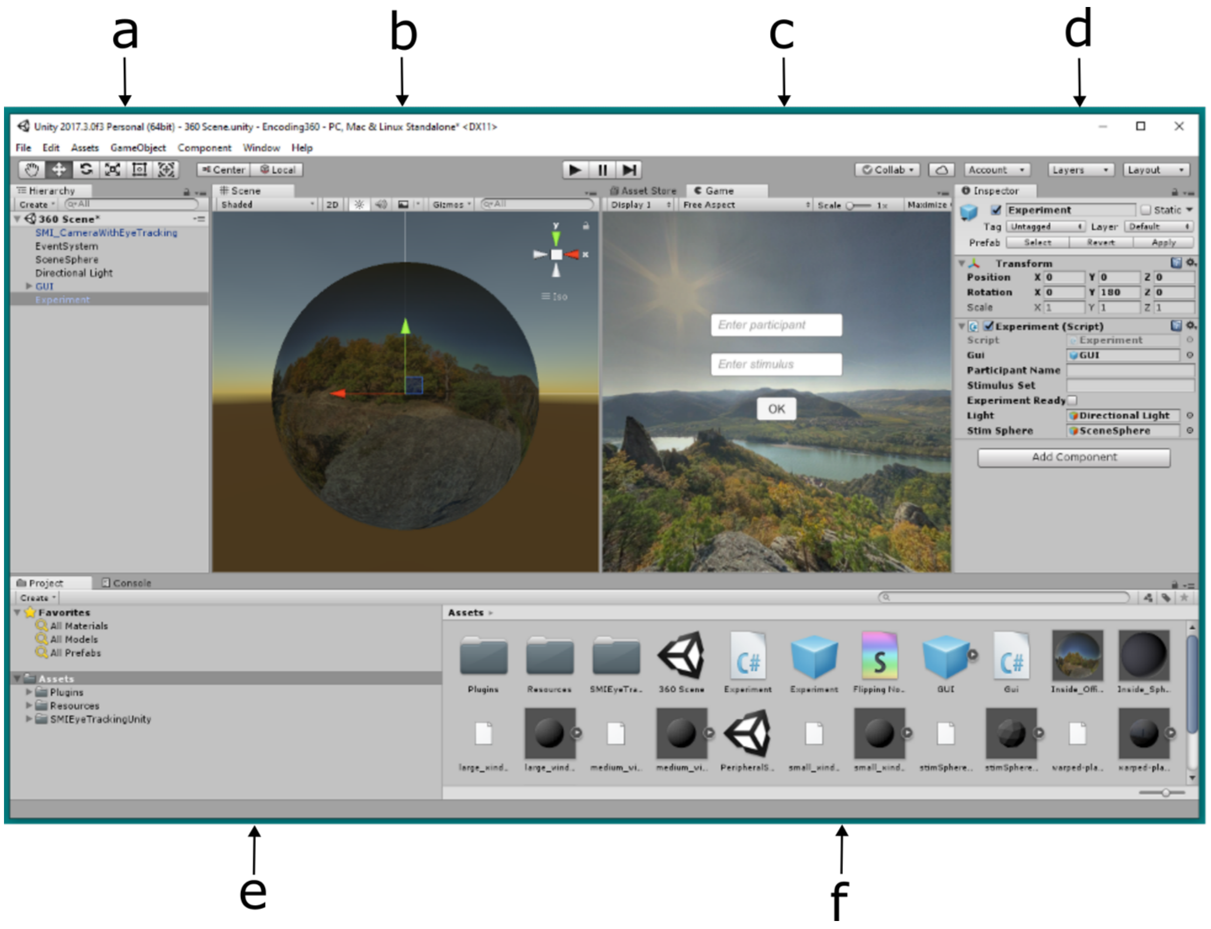
In this section, we present our work on eye and head movements in 360° panoramic scenes, providing a concrete example and "how-to" information for developing experiments in Unity. We also include details that might not be in our published work (for example, experiment control flow principles in C#). Note that several excellent experiment builder type programs have been in development for Unity (Section 3.1), but here we aim to provide a working example of a basic Unity project that includes eye tracking. For the purposes of this example, we assume that the reader has a basic understanding of the Unity Editor and project setup, as explained in the Beginner Unity tutorials. A good place to start might be the "Unity Essentials pathway in Unity Learn" (*Unity Essentials*, n.d.).

In this example study, participants were asked to move their eyes and head to look at a random selection of 80 indoor and outdoor 360° panoramic scenes and remember them for a later memory test. Participants looked at the scenes using a head-mounted display with a built-in SMI eye tracker, and each scene was displayed for 10 seconds. A uniform gray scene with a black fixation cross presented directly ahead of the participant was displayed between images, and participants were instructed to look at this fixation cross and press a key to initiate the next trial. At the beginning of every 20th trial, the eye tracker was calibrated using a 9-point display specific to the SMI software.

4.1 Unity 3D Environment

The Unity project architecture of our example experiment is quite straight-forward. The scene consists of a few basic elements (see Figure 4) defined below:

- Camera: The camera is a Unity object provided by SMI. It represents the headset, and moves in the space when the headset is moved or worn by participants. The camera provided by SMI determines not only the point of view, but it also manages the eye tracking. Eye-tracker related functionality can be set and changed via an SMI-specific script, which is attached to an otherwise standard camera object.
- Scene Sphere: This is a 3D sphere game object with a shader and sphere collider attached. The shader is somewhat special, such that textures applied to the surface of the sphere (the panoramic scenes) are visible not from the outside, but the inside, to allow observers in the middle of the sphere to see the scenes. The collider provides the physical properties needed for this object to interact with other physical aspects of the project.
- Directional Light: A light source is required for illuminating the inside of the sphere. That light source is positioned below the camera and aligned with the forward direction of the camera.
- Experiment: The empty game object 'Experiment' has a script attached with the name Experiment.cs. The bulk of the work in the experiment is done in this script, key components of which are described in greater detail below. Importantly, game objects that are modified or referred to in this script are attached to this script in the Unity environment (for example, the SceneSphere and DirectionalLight objects shown in the Inspector window of Figure 4).
- GUI object: This object is responsible for interacting with the experimenter to record participant and stimulus information.
- Event system: This is used for sending events to other objects based on input, for example, a keyboard or an eye tracker. The Event System consists of a few components that work together to send events.



Experiment ✓ — Static ☐
Tag: Untagged — Layer: Default
Prefab: Select | Revert | Apply

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 180 Z 0
Scale X 1 Y 1 Z 1

Experiment (Script) ✓
Script: Experiment
Gui: GUI
Participant Name:
Stimulus Set:
Experiment Ready ☐
Light: Directional Light
Stim Sphere: SceneSphere

Figure 4: Example of a simple experimental setup for a Unity project. Objects in the scene are listed in the “Hierarchy” window (a), which consists of the camera (in this case, a camera combined with an SMI eye tracker), an EventSystem, a sphere (SceneSphere) with a special shader and a collider, a directional light, a GUI object and an empty game object called ‘Experiment.’ The current scene (the spherical stimulus as seen from the outside) is shown in (b) and the initial display (for entering experiment information) is shown in (c). The “Inspector” window (d) shows the details of the ‘Experiment’ game object, which has a transform indicating where this invisible empty game object is in the virtual space, as well as a script (also named ‘Experiment’) attached to it. The bottom of the display lists assets that can be included in the experiment (e and f).

4.2 *Experimental Control Flow*

One of the things typically taken care of by experiment builders such as PsychoPy (Peirce et al., 2019) and OpenSesame (Mathôt et al., 2012) is experimental control flow, that is, the movement from one trial to the next, and the management of different experimental blocks. In Unity, experimental control is complicated by the fact that Unity is in charge of program execution and is calling the user-defined Experiment script at more or less regular intervals. Essentially, the script that takes care of experiment control flow is called during each headset frame refresh. For this reason, the script has to keep track of the state of execution to guarantee an orderly execution. This is described in detail in the next section.

4.3 Script Experiment.cs

The Experiment.cs script is attached to the 'Experiment' Unity object and is responsible for most of the "real work" done in the experiment. This script has links to relevant objects in the scene, such as the light source and the stimulus sphere, and it initiates and modifies their states. The script is written in C# and has two main functions, Start() and Update().

The Start() function is called at the beginning of the experiment and contains code that must be executed at the beginning of the experiment (e.g., initializing the variables that are constant for the entire experiment such as participant information and accessing the stimulus sphere material and setting it to a variable for later use). This script reads in a control file with descriptive information about each trial. Each row in this file represents a trial and includes information about the image name, what type of image it is (in our example, either an indoor or outdoor scene) and further information associated with the trials (see Table 1):

uid	*image*	*imagetype*	*task*
1	indoor_pano_aaabgnwpmpzcgv	indoor	encoding
2	indoor_pano_aaabnctbyjqifw	indoor	encoding
3	indoor_pano_aaacisrhqnnvoq	indoor	encoding

Table 1: Example control file containing trial information.

The Update() function is responsible for the bulk of the experimental control flow code. It is called from Unity approximately 50 – 70 times per second. For this reason, the code must keep track of the state of the experiment, such that, on every call to Update(), the script continues at the correct place. To achieve this, we use a C# programming structure called a switch statement (essentially a series of if… else if statements), where a code block gets executed based on a match to a list of possible states called experimentPhase (see Figure 5). Then we only need to keep track of the state experimentPhase between calls to Update(). Each phase state takes care of a particular part of the experiment such as calling for a calibration (in our example, this occurs every 20 trials), showing a stimulus, or waiting for a participant response.

```
enum ExperimentPhase { preparation, calibration, waitForCalibration,
       waitForParticipant, stimulus, finished };

switch (phase) {

       case ExperimentPhase.preparation:

              // code run prior to starting experiment
              phase = ExperimentPhase.calibration;
       break;

       case ExperimentPhase.calibration:

              // code to run calibration coroutine
              phase = ExperimentPhase.waitForParticipant;
       break;

       case ExperimentPhase.waitForCalibration:

              // code to show fixation screen while waiting for calibration to run
              phase = ExperimentPhase.waitForParticipant;
       break;

       case ExperimentPhase.waitForParticipant:

              // wait for participant to indicate they're ready for the trial to start
              if (ParticipantResponse()) {
                     phase = ExperimentPhase.stimulus;
              }
       break;

       case ExperimentPhase.stimulus:

              // code to run trial sequence coroutine
       break;

       case ExperimentPhase.finished:

              // code to run at end of experiment
              QuitExperiment();

       break;
}
```

Figure 5: Switch statement used to keep track of what state the experiment is in on every given call of Update().

Each case in the switch case statement has links to others, so that for example, after the ‘waitForParticipant’ state, which waits for the participant to press a specific key on the keyboard, the variable ‘phase’ is assigned to the ‘stimulus’ case and the code for controlling what happens during stimulus presentation is executed. This basic structure can be adapted to suit many types of simple experimental design. For example, including an experimental block structure requires the addition of a phase that checks the trial number after each trial is run. If it matches the number of trials in a block, an instruction screen specific to that block is run. How the trial changes across blocks, in this case, is handled in the stimulus phase with an if statement checking which block is currently being run. The block information itself could be located in the .csv file containing the trial information.

4.4 Eye Tracking Implementation

The SMI eye tracker is able to record eye movements at a rate of about 250 Hz, but the Unity system invokes the Update() function at a rate of only about 50 – 70 Hz. For this reason, if eye tracking is controlled through the Unity system, eye tracking speed is substantially reduced. In our example, we run the eye tracker independently of the Unity system and at the maximum possible speed (i.e., at 250 Hz) via multithreading. This is accomplished by creating and starting a gaze tracking function in a new thread, i.e., in a program segment that runs independently of the rest of the program. The function called in this thread GazeTrack() creates two lists, gazeTime that contains the timestamp output from a custom SMI function, and gazeDirection, which indicates the current gaze direction. At the end of each trial, the data in these lists are then passed to a function called WriteGazeData(), which transforms the data appropriately and writes them to the output file.

```
IEnumerator StartTrialDuration(string imageName, float duration)
{

        ...
        recordGaze = true;

        Thread gazeThread = new Thread(GazeTrack);
        gazeThread.Start();

        SetSphereImage(imageName);

        yield return new WaitForSeconds(duration);

        recordGaze = false;
        WriteGazeData();
        currentTrialNumber++;

        ...
}

...

void GazeTrack()
{
        Vector3 rayCast;

        while (recordGaze)
        {
                timeStamp = SMI.SMIEyeTrackingUnity.Instance.smi_GetTimeStamp();

                if (timeStamp > oldTimeStamp)
                {
                        if (nGazeReadings < maxGazeReadings)
                        {
                                rayCast = 
SMI.SMIEyeTrackingUnity.Instance.smi_GetCameraRaycast();
                                gazeTime[nGazeReadings] = timeStamp;
                                gazeDirection[nGazeReadings] = rayCast;
                                nGazeReadings += 1;
                        }
                        oldTimeStamp = timeStamp;
                }
        }
}
```

Figure 6: Example code for implementing eye tracking in C# via multithreading.

As shown in Figure 6, gaze direction is returned by the SMI system as a ray emanating from the center of the headset. This must be translated to spherical coordinates (longitude and latitude) using functions that compute the intersection of the gaze ray with the stimulus sphere. Note that due to characteristics of Unity, hits can only be detected from the outside, so the ray must be sent out and then reversed in direction in order to hit the outside of the sphere. Alternatively, this could be computed directly using information on the location and orientation of the headset with respect to the center of the sphere, the gaze direction, and the radius of the sphere.

Other data, in particular headset position-related data obtained through Unity, is recorded at about 50-70 Hz. In WriteGazeData() from Experiment.cs the two data streams can be combined and synchronized, provided that precise timing information has been recorded for gaze and the other variable. In our example, this is done via linear interpolation (see Experiment.cs).

5 Data Handling

In this section, we outline some of the data handling issues that are unique to VR.

5.1 Reference Frames

Before diving into the specifics of eye, head, and gaze analysis in VR, it is worth clarifying the reference frame we are dealing with (see Hessels et al., 2018). Most readers may be familiar with the popular desktop-based eye tracking technology, where observers are required to sit in a chin rest at a set distance away from a computer monitor. In this situation, the head is fixed, and eye movements are reported with respect to the computer monitor, usually in some form of pixel location or degrees of visual angle. In other words, the reference frame for eye movements is the screen, which typically encompasses approximately 30-50° visual angle, depending on the particular setup.

In mobile eye tracking experiments, the eyes are tracked by one or two cameras pointed toward the eyes, while the scene is recorded in a forward-facing camera and the head and body is free to move naturally. The reference frame in this case is the scene camera, with eye position reported with respect to their location in the scene camera, ranging from around 60-120° visual angle. In this case the head is free to move, so the reported eye coordinates are essentially head-based. Head movements can be estimated from the movements of the scene camera (e.g., Dufaux & Konrad, 2000; Ohayon & Rivlin, 2006), while gaze position is typically hand coded, or it can be extracted with the use of reference markers placed in the world and detected via software (for e.g., using Pupil Labs, Kassner et al., 2014).

In VR experiments, similarly to mobile eye tracking, the eyes are tracked by cameras mounted within the HMD. The head and body are often free to move naturally, although the weight of the headset, participant pose (standing vs. sitting), and potential tethering can influence participants' range of motion. One significant advantage of this setup is that eye~~,~~ and head movements can be tracked with respect to a common reference frame. This means that things like eye eccentricity in the head coordinate system, as well as the contributions of head movements to gaze position can be calculated precisely (more on this below). Note however, that without special equipment, neck movements cannot be differentiated from chair and torso movements. For the sake of simplicity, in the example we present below, we use the position of the HMD in space as a proxy for head movements alone.

5.2 Fixation and Saccade detection

Unlike more standard, desktop and mobile-based eye tracking, to date, VR eye tracking implementations do not come with analysis programs such as Eyelink's DataViewer (SR Research Ltd.), or Pupil Lab's Pupil Player (*Core - Pupil Player*, n.d.) that automatically parse gaze data into blinks, fixations, and saccades. In this section, we review event detection in gaze analysis that can be used in VR with a focus on the detection of fixations and saccades, while other ocular events, such as smooth pursuit, micro-saccades, or blinks are ignored (see for example, Holmqvist & Andersson, 2017). Note that the SMI eye tracker used in our example in Section 6 automatically omits blinks from the recorded data. Figure 7 shows an example output from an event detection algorithm for a single trial in our example dataset.

There are two fundamentally different approaches to gaze analysis. The first approach starts with the detection of fixations, and saccades are defined as differences between successive fixations, whereas the second approach starts with the detection of saccades, and fixations are defined as stable points between saccades. A popular method for the detection of fixations is the Dispersion-Threshold (IDT) algorithm (Komogortsev et al., 2010; Salvucci & Goldberg, 2000), which assumes that the dispersion of gaze points within a fixation is relatively small (in our studies typically 2.5-3°) and that the duration of fixations exceeds a minimum duration (in our studies typically 80 ms). Specifically, the IDT algorithm proceeds as follows:

1. Initialize a window of gaze points to cover duration threshold. Drop the data points if the dispersion of the gaze points exceeds the dispersion threshold.
2. Add further gaze points to the window as long as the dispersion of the gaze points does not exceed the dispersion threshold.
3. Define the fixation position as the centroid of gaze points.
4. Remove the gaze points of the fixation and start again from step 1.

An alternative method for fixation detection relies on gaze vector velocities, where in step 2, gaze points are added to the window as long as the velocity of successive gaze points does not exceed the velocity threshold. For both, the IDT algorithm and the velocity algorithm, saccades are defined as differences between successive fixations.

The second approach begins with the detection of saccades, and fixations are defined as stable points between saccades. The detection of saccades is based on the assumption that motion above a velocity threshold is assumed to be (part of) a saccade. Specifically, the algorithm proceeds as follows:

1. Calculate all gaze velocities between successive gaze points.
2. Detect peak velocities (which are assumed to define the middle of a saccade).
3. Add velocities immediately before the peaks and immediately after the peaks as long as they exceed a velocity threshold. Velocities below that threshold are assumed to be part of a fixation.
4. Peak velocities must be below a certain limit to exclude artefacts, such as blinks.
5. Finally, fixations are defined as the relatively stable positions between saccades.

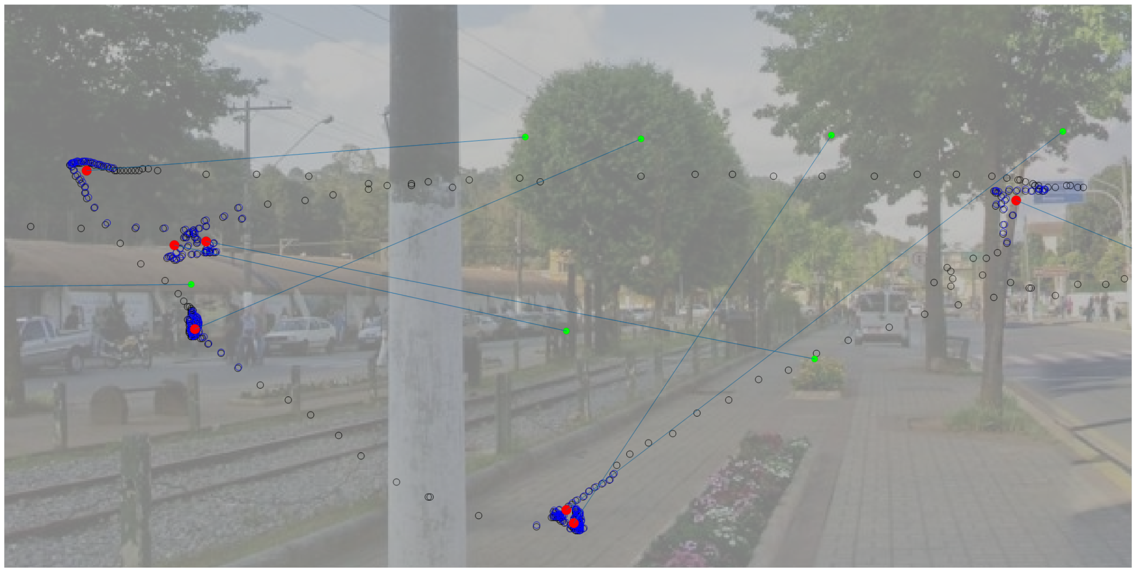


Figure 7: Example visualization of the IDT algorithm (zoomed in for clarity). Black dots are gaze samples, red dots are fixations, and blue dots are the gaze samples that contribute to each nearby fixation. Green dots are averaged head positions during a given fixation.

5.3 Using Spherical Coordinates

Event detection is typically done on gaze positions represented as pixel locations on a computer monitor (or with respect to some standard coordinate frame, such as the world video in mobile eye tracking systems). In VR the situation is complicated by the fact that gaze positions could occur at any point around the participants in the fully immersive space. One way to represent gaze positions in such a situation is in longitude and latitude. In the situation where a participant is looking at a 360° panoramic scene, these could be calculated with respect to the scene sphere (as in the example presented in Section 4). However, in a fully immersive 3D environment, this has typically been done by computing gaze positions in longitude and latitude with respect to an imaginary unit sphere surrounding the headset (see Figure 8). Event detection is then done using these spherical coordinates (David et al., 2020, 2021), and other gaze measures, such as what 3D object a participant was looking at, can be obtained via raycasting or by utilizing physical interactions of the gaze ray and objects in the virtual world. For a detailed example of how to extract gaze ray information from a virtual sphere see Section 4.4.

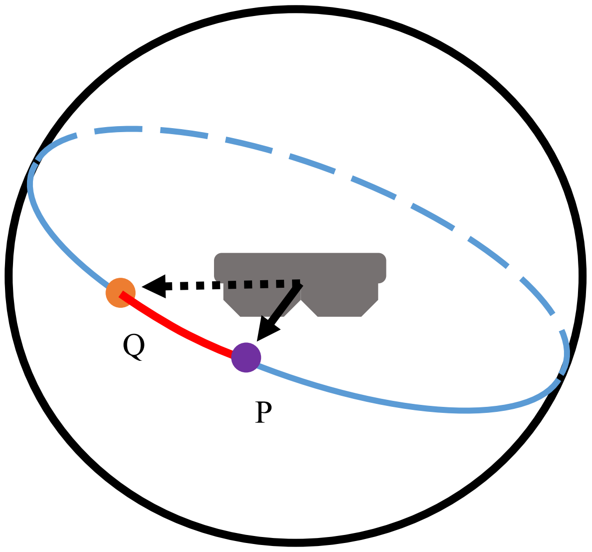


Figure 8: Example gaze points P and Q projected onto a virtual unit sphere surrounding the VR headset. The red line represents the angular distance between the fixations, which is calculated using the great circle distance.

There are a few key points to keep in mind when doing event detection in a fully immersive VR environment where circular statistics must be taken into account (Batschelet, 1981; Bischof et al., 2020 Appendix 1). One must pay particular attention to how distances between successive gaze positions are calculated. This has implications for the dispersion threshold (usually represented in degrees visual angle) as well as the saccade amplitudes (distances between fixations). In a fully immersive 360° world, these distances must be calculated using the great circle distance, otherwise known as the orthodromic distance, which is defined as the shortest distance between two points on a sphere (see Figure 8).

6 Data Analysis

In this section, we focus on the analysis of gaze and head movements of observers in a 360° panoramic virtual environment. We use the data from our example project from Section 4, however, many of the measures we outline below can be adapted to other situations in VR that are not specific to panoramic scenes. To recap, the virtual environment is produced by projecting images on the interior of a sphere with the observer's head at the center of the sphere. In our examples, the environment is static, with all points of the environment at the same distance from the observer, and it contains no moving objects that can be tracked. In the following sections, we present first the basics of gaze analysis and the analysis of saccades and head movements. Finally, we present an analysis of eye-in-head measures and the spatial and temporal relationship between eye and head movements.

6.1 Gaze Measures

A gaze point is defined as the intersection of the gaze vector with the virtual sphere on which the panoramas are projected. Azimuth and elevation of this point are described with coordinates longitude in the range [-180,180] degrees and latitude in the range [-90,90] degrees. Similarly, we define the head point as the intersection of the vector pointing forward from the face with the virtual sphere, and it is also defined in world coordinates (i.e., longitude and latitude of the panorama, see Section 5 for more details).

One way to visualize gaze points, or sets of gaze points, is to project them onto a flat map, for example, an equirectangular (or equidistant) projection map. This projection maps meridians into to vertical straight lines of constant spacing, introducing distortions near the poles compared to the equator. This is illustrated in Figure 9.

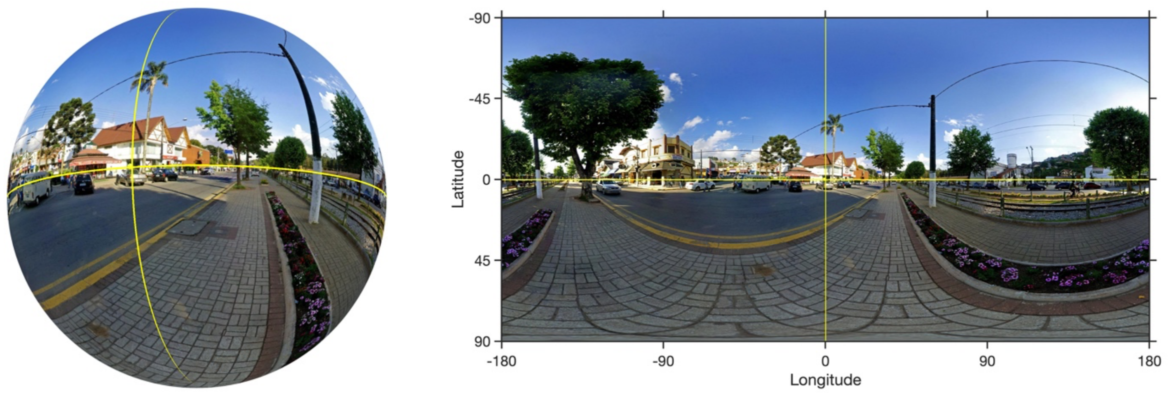


Figure 9: (left) Stimulus sphere. (right) Equirectangular (equidistant) map of the stimulus. The yellow lines indicate the equator and the center meridian. Note the distortions near the north and south poles. In analyzing fixation patterns, these distortions must be taken into account.

An analysis of typical fixation patterns in spherical displays shows that there seems to be a preference for fixations along the equator of the virtual sphere, a tendency that is referred to as equator bias (see Figure 10 (left)). There are multiple causes that may contribute to the equator bias. First, if participants inspect the panorama with neck extension and flexion in a resting state and the eyes are centered in the head coordinate system then there is a natural preference for fixations along the horizon. Second, an analysis of typical panorama images shows that on average, edge density is strongest along the equator (see Figure 10 (right)), which may be due to the fact that there is simply more content along the horizon in typical panoramic scenes (as in, for e.g., Torralba et al., 2006). Figure 10 (right) was constructed by computing the edge images of a large number of panorama images and averaging them.

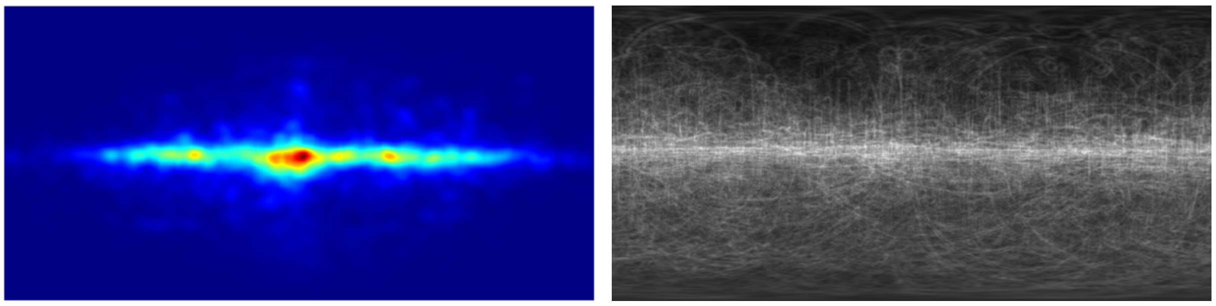


Figure 10: (left) Fixation heatmap. (right) Edge density averaged over many images.

Edges occur in regions where there is a strong change in grey-level of the panorama images. To generalize the edge density map, one can compute the entropy of local neighborhoods in the panorama images. This is done by computing grey-level histograms in a grid overlayed over the image, then computing the entropy of the histograms, and finally averaging the entropy images over many panorama images. In this analysis, regions with large entropy are assumed to "attract" attention, leading to peaks in the fixation heatmaps. As a further generalization of local entropy maps, one can compute the saliency maps of panoramic images. In Engineering and Computer Science (e.g., De Abreu et al., 2017; Sitzmann et al., 2018), there is substantial work on the saliency of panoramic images, but that work always includes fixation patterns for predicting saliency, while we are interested in predicting fixation patterns from saliency based on image properties.

For other general gaze movement measures that might be of interest in panoramic scenes see Bischof and colleagues (2020) and David and colleagues (2022).

6.2 Analysis of Saccades

Saccade patterns in panoramic scenes are closely related to the fixation distributions. Given the large spread of fixations along the horizon of the images, it is plausible that saccade directions also align with the scene horizons. This is illustrated in Figure 11 (left), which shows a polar histogram of saccade directions. The histogram shows that most saccades were made along the horizon direction of the panoramas. In addition, Figure 11 (right) also shows a histogram of the saccade amplitudes. On average, in free viewing of natural panoramic scenes saccades are typically on the order of 10-20° visual angle. For a more detailed analysis of saccade characteristics in panoramic scenes, see David and colleagues (2022).

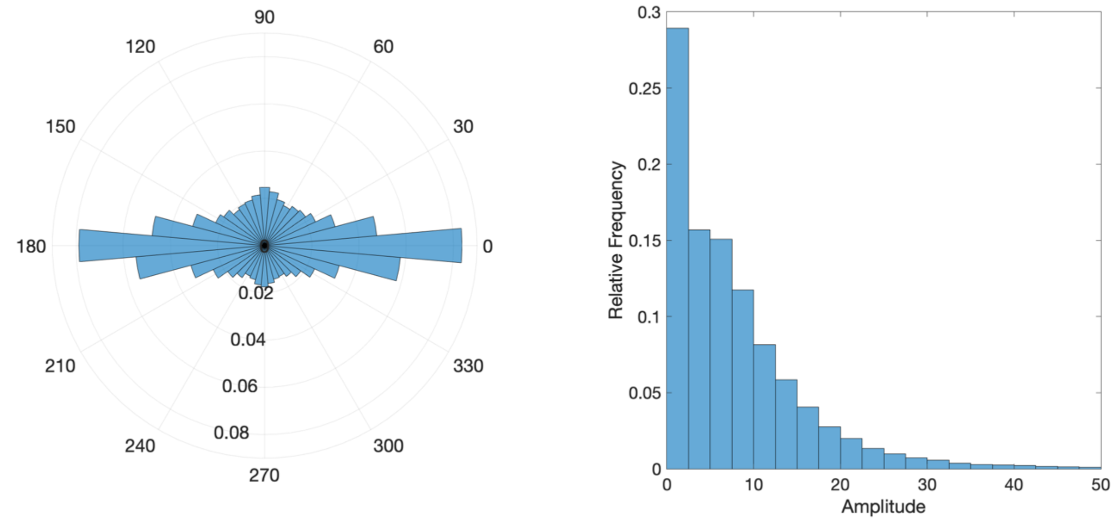


Figure 11: (left) Polar histogram of saccade directions. Most saccades are aligned with the horizon of the panorama images. (right) Histogram of saccade amplitudes.

6.3 Head Analysis

Head movements are defined by head pitch, which is achieved through neck extension and flexion, head yaw, which is achieved through lateral neck rotation, and head roll, which is achieved through lateral bending of the neck. Head points are defined as the intersection of the vector pointing forward from the face with the virtual panorama sphere surrounding the observer and is defined in world coordinates (i.e., longitude and latitude of the panorama). Head movements with a VR viewer are not ballistic in the way that eye movements are. For this reason, there are, in contrast to gaze, no natural demarcations for head shifts and head fixations.

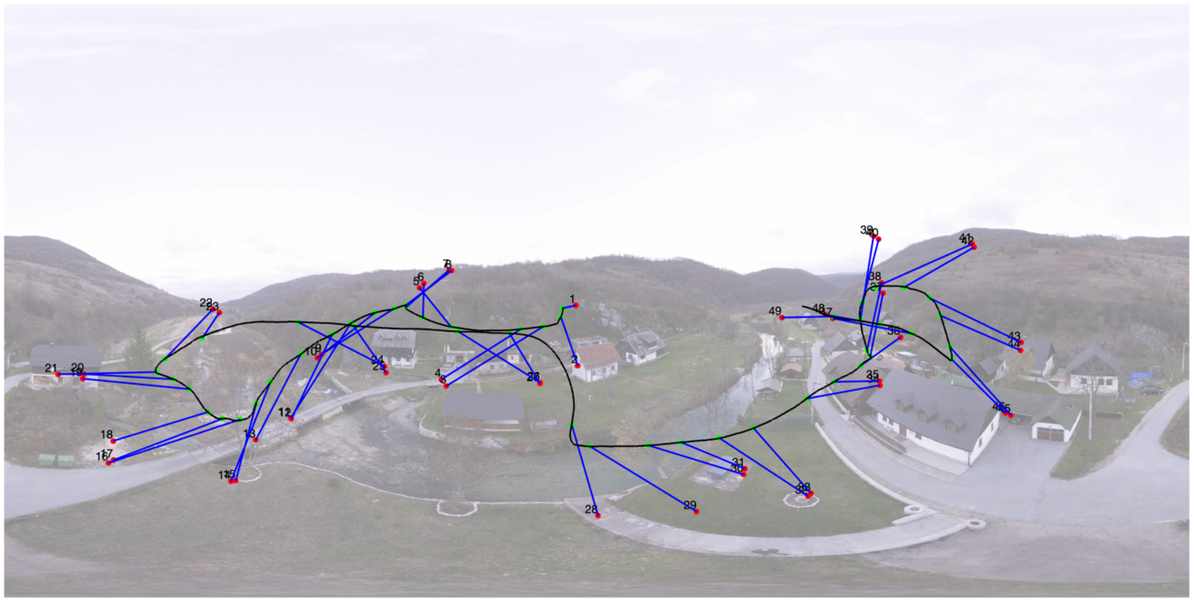


Figure 12: Example panorama map with gaze and head positions. The red circles indicate gaze fixations, the black line shows the head positions, the green circles indicate head averages during the fixations, and the blue lines connect fixations with the corresponding head averages. These lines thus represent distances between gaze and head, and are determined by the eye direction in a head-centered coordinate system.

Gaze and head positions are illustrated in Figure 12. One way to analyze head movements is to examine head point patterns independent of gaze. To relate head movement information to gaze, one can link the two using common timing information. Alternatively, one can compute the average of head points during a fixation (referred to, in this chapter, as head average) and then directly relate the head averages to fixations. In our experience, the latter approach has been the most straight-forward for relating head movements to fixations, however, care must be taken when interpreting head positions in this way.

Comparable to gaze, one can plot a heatmap of head averages for sets of images. Figure 13 shows a head average heatmap for the same images used in Figure 10. Given that gaze positions deviate only by a moderate angle from the head position (see Section 6.4.1 below), it is not surprising that the head points are also concentrated along the horizon.

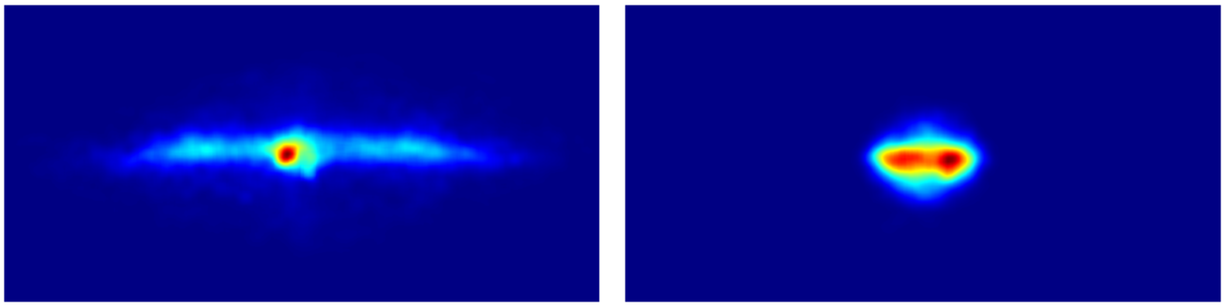


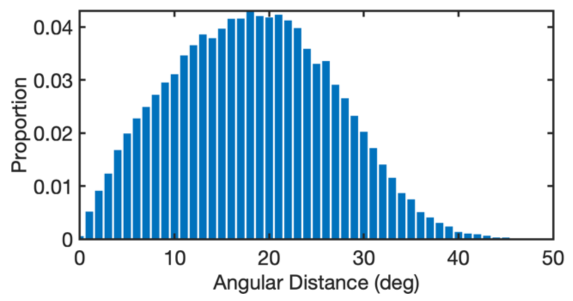
Figure 13: (left) Heatmap of head averages; (right) Heatmap of eye-in-head positions. Both heatmaps were computed for the same set of images as in Figure 10.

6.4 Eyes

We define eyes-in-head as the gaze direction in a head-centered coordinate system. The eye hit point, or eye point, is defined as the difference between gaze point and head point and is also expressed in world coordinates (i.e., in longitude and latitude), with the origin (longitude and latitude equal to 0°) at the head point. Figure 13 (right) illustrates the eye heatmap corresponding to the gaze heatmap in Figure 10 (left) and the head heatmap in Figure 13 (left). Note that in computing they eyes-in-head map, care must be taken to account for the distortions near the north and south poles because the distance between meridians is much smaller near the poles than at the equator. This is achieved using circular statistics (See Section 5.3).

6.4.1 Spatial relation between gaze and head

As illustrated in Figure 13 (right), gaze points deviate only moderately from head points. Typically, in unrestrained viewing of panoramic scenes, gaze deviates from the head direction only by a moderate amount, is somewhat ahead of the head movement, and covers a larger area of the visual field. The latter helps to preserve energy because moving the eyes requires less physical effort than moving the head. The relationship between gaze and head is further illustrated in Figure 14 (left), which shows the histogram of distances between gaze fixations and head averages. This corresponds to eye eccentricity in the head-defined visual field. As seen in Figure 14, most eye eccentricities are in the range 10 – 25°.



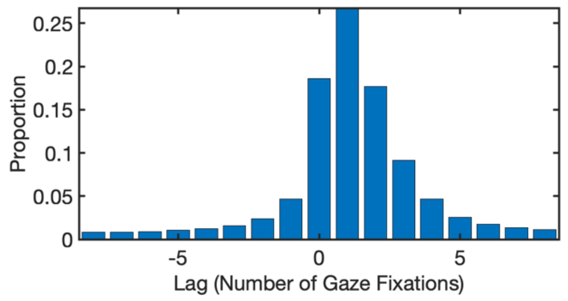


Figure 14: (left) Distribution of distances between head points and gaze points. (right) Histogram of gaze-head lags, with the lag expressed in number of gaze fixations. If gaze has a positive lag, then gaze is leading. If gaze has a negative lag, then gaze is trailing.

6.4.2 Temporal relation between gaze and head

To determine the temporal relationship between gaze fixations and head positions, one proceeds as follows: Given a gaze fixation g_i and a set of head fixations h_j before and after the time point i, one determines the h_j^{min} with minimum distance. If the h_j^{min} occurs before g_i then it is concluded that head is leading gaze, otherwise it is concluded that head is lagging behind gaze. The results of this minimum distance analysis for a set of panorama images are shown in Figure 14 (right), which indicates that most minimum distances occur at a positive lag (i.e., the gaze positions are most often leading the head positions). The lag peak is around 1 – 2 fixations, suggesting that gaze leads the head by approximately 200 ms.

6.5 Observations on eye and head movement behaviour while looking at 360° scenes

The studies mentioned throughout the chapter and the data reported in this section revealed several important characteristics of the interplay between eye and head movements in 360° panoramic scenes. First, not unsurprisingly, the head and eyes tend to follow along the horizon of 360° scenes, where most head and gaze positions are found along the horizon and most saccades are made along the horizon (at least in cases where a notable horizon exists, Bischof et al., 2020). Second, we found consistently that the head tends to follow the eyes, indicating (based on earlier research, Doshi & Trivedi, 2012; Freedman, 2008; Zangemeister & Stark, 1982) that viewing 360° scenes in VR follows a pattern of reflexive orienting, most likely with the eyes and head responding to image cues. This observation stands in contrast to studies where the head played a more central role by moving to reveal new information to the eyes (Solman et al., 2017). Third, the eyes tend to stay relatively close to the center of the head-defined visual field, consistent with the results obtained with mobile eye tracking (e.g., Foulsham et al., 2011; ’t Hart et al., 2009).

More speculatively, we have noticed that there are typically substantial individual differences in the amount that an observer moves their head during visual explorations. While some observers move their head extensively, others keep their head very still. This distinction between head “movers” and “non-movers” has been found repeatedly in the kinematic literature (e.g., Delreux et al., 1991; Fuller, 1992a, 1992b; Goldring et al., 1996; Stahl, 2001). Taken together, these works provide evidence that head and eye movements diverge in their control strategies, leading researchers to speculate that the head may be under more deliberate, cognitive control (David et al., 2022) or are sensitive to different spatial reference frames (Anderson et al., 2020).

7 Open Questions and Future Directions

Implementing eye tracking into experiments in VR can be challenging, but we hope that this chapter provides researchers with practical and actionable advice on how to begin. At the start of this chapter, we made a case for why it is important to track eye movements in VR, and in this Section, we leave readers with a sense of the types of questions that VR can help us answer about visual cognition and behaviour:

1) What is the nature of the relationship between eye, head, and other body movements in supporting human cognition?

   This question is not only a kinematic one, where researchers may be interested in exactly how and when the cognitive system recruits different effectors like the head and body to support ongoing thought and behaviour, but it also can provide clues to how these systems relate to everyday, realistic environments. VR sits between the extremely constrictive computer-based eye tracker situations (where most research has been conducted) and the unrestrained but more natural approach taken with mobile eye trackers by researchers interested in the more complex but less controlled everyday situations. VR provides a way to precisely measure human movements in naturalistic situations, while simultaneously controlling the external inputs to the system. One can imagine that by simplifying the VR environment similar to computer-based studies (e.g., Folk et al., 1992; Henderson, 2016; Silvis & Donk, 2014; Theeuwes, 1994; van Zoest et al., 2004), combined with added head and body movement information, it would be possible to tackle some of the fundamental, unanswered questions involving eye and head movements (see for example, Flindall et al., 2021).

2) What are the consequences of studying human cognition in a situation where the head is restrained?

   This question refers to the growing realization among researchers that restricting head movements in a chin rest, and pre-selecting the stimuli that observers are allowed to view, may bias the type of data one obtains and the conclusions one reaches. First, participants must explore the stimuli using eye movements alone. In contrast, our studies have shown that the eyes and head work in conjunction to explore the visual world. Second, in a paradigm with peripheral masking, David and colleagues (2021) have demonstrated that the previously reported visual span of 6° visual angle obtained with head-fixed studies may be a gross underestimation. It should be emphasized that the head movements in our studies involved only changing the head orientation (pitch, yaw, roll), but not changing the position of the head in space.

3) What are the similarities and differences between perceiving and acting on virtual items and versus those that are real?

   This question is motivated by the growing recognition that the way people see and act on objects that are virtual may often engage very different forms of cognition and behaviour, as well as different brain systems, than those that are real (e.g., Dosso & Kingstone, 2018; Freud et al., 2018; Gallup et al., 2019). Determining when one's findings are specific to virtual stimuli versus when they generalise to real-world situations is of fundamental and profound importance, and one that VR research promises to unlock in the future.

It is our hope and intention that the material presented in this chapter will provide researchers with the basic skills necessary to begin to engage in this exciting research enterprise. We have focused on elements that are critical to conduct research in VR that focuses on the movement of the head and eyes, and how they relate to one another as well as the external environment. The power of VR is that one can create, manipulate, and control the environment that an individual is immersed within, ranging from the simple environments that are routinely used in lab experiments, to much more complex real-world situations, to creating environments that are, literally, not of this world! The challenge, of course, is how one can make sense of the data that one obtains in such studies. We hope that the material presented in this chapter will enable researchers to pursue their own research questions in a manner that is both theoretically exciting and empirically tractable.

8 References

Anderson, N. C., Bischof, W. F., Foulsham, T., & Kingstone, A. (2020). Turning the (virtual) world around: Patterns in saccade direction vary with picture orientation and shape in virtual reality. *Journal of Vision*, *20*(8), 21–21.

Backhaus, D., Engbert, R., Rothkegel, L. O. M., & Trukenbrod, H. A. (2020). Task-dependence in scene perception: Head unrestrained viewing using mobile eye-tracking. *Journal of Vision*, *20*(5), 3–3. https://doi.org/10.1167/jov.20.5.3

Barnes, G. R. (1979). Vestibulo-ocular function during co-ordinated head and eye movements to acquire visual targets. *The Journal of Physiology*, *287*(1), 127–147. https://doi.org/10.1113/jphysiol.1979.sp012650

Batschelet, E. (1981). Circular statistics in biology. *Academic Press, 111 Fifth Ave., New York, NY 10003, 1981, 388*.

Bebko, A. O., & Troje, N. (2020). *bmlTUX: Design and control of experiments in virtual reality and beyond*. PsyArXiv. https://doi.org/10.31234/osf.io/arvkf

Bischof, W. F., Anderson, N. C., Doswell, M. T., & Kingstone, A. (2020). Visual exploration of omnidirectional panoramic scenes. *Journal of Vision*, *20*(7), 23. https://doi.org/10.1167/jov.20.7.23

Blue Trident IMU | Inertial Sensor by Vicon | Biomechanic Tracking. (n.d.). Vicon. Retrieved March 30, 2022, from https://www.vicon.com/hardware/blue-trident/

Brookes, J., Warburton, M., Alghadier, M., Mon-Williams, M., & Mushtaq, F. (2020). Studying human behavior with virtual reality: The Unity Experiment Framework. *Behavior Research Methods*, *52*(2), 455–463. https://doi.org/10.3758/s13428-019-01242-0

Core—Pupil Player. (n.d.). Pupil Labs. Retrieved March 31, 2022, from https://docs.pupil-labs.com

David, E. J., Beitner, J., & Võ, M. L.-H. (2020). Effects of Transient Loss of Vision on Head and Eye Movements during Visual Search in a Virtual Environment. *Brain Sciences*, *10*(11), 841. https://doi.org/10.3390/brainsci10110841

David, E. J., Beitner, J., & Võ, M. L.-H. (2021). The importance of peripheral vision when searching 3D real-world scenes: A gaze-contingent study in virtual reality. *Journal of Vision*, *21*(7), 3–3.

David, E. J., Lebranchu, P., Perreira Da Silva, M., & Le Callet, P. (2022). What are the visuo-motor tendencies of omnidirectional scene free-viewing in virtual reality? *Journal of Vision*, *22*(4), 12. https://doi.org/10.1167/jov.22.4.12

De Abreu, A., Ozcinar, C., & Smolic, A. (2017). Look around you: Saliency maps for omnidirectional images in VR applications. *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, 1–6.

Delreux, V., Abeele, S. V., Lefevre, P., & Roucoux, A. (1991). Eye–head coordination: Influence of eye position on the control of head movement amplitude. *Brain and Space*, 38–48.

Doshi, A., & Trivedi, M. M. (2012). Head and eye gaze dynamics during visual attention shifts in complex environments. *Journal of Vision*, *12*(2), 9–9.

Dosso, J. A., & Kingstone, A. (2018). Social modulation of object-directed but not image-directed actions. *PLOS ONE*, *13*(10), e0205830. https://doi.org/10.1371/journal.pone.0205830

Dufaux, F., & Konrad, J. (2000). Efficient, robust, and fast global motion estimation for video coding. *IEEE Transactions on Image Processing*, *9*(3), 497–501.

Eyetracking solutions for behavioral studies. (n.d.). *ERGONEERS*. Retrieved March 30, 2022, from https://www.ergoneers.com/en/hardware/eye-tracking/

Flindall, J., Sara, A., & Kingstone, A. (2021). Head and eye movements are each facilitated by the offset of a central fixation point in a virtual gap paradigm. *Experimental Brain Research*, *239*(1), 117–126. https://doi.org/10.1007/s00221-020-05905-9

Folk, C. L., Remington, R. W., & Johnston, J. C. (1992). Involuntary covert orienting is contingent on attentional control settings. *Journal of Experimental Psychology Human Perception and Performance*, *18*, 1030–1030.

Foulsham, T., & Kingstone, A. (2010). Asymmetries in the direction of saccades during perception of scenes and fractals: Effects of image type and image features. *Vision Research*, *50*(8), 779–795. https://doi.org/10.1016/j.visres.2010.01.019

Foulsham, T., Kingstone, A., & Underwood, G. (2008). Turning the world around: Patterns in saccade direction vary with picture orientation. *Vision Research*, *48*(17), 1777–1790.

Foulsham, T., Walker, E., & Kingstone, A. (2011). The where, what and when of gaze allocation in the lab and the natural environment. *Vision Research*, *51*(17), 1920–1931.

Freedman, E. G. (2008). Coordination of the eyes and head during visual orienting. *Experimental Brain Research*, *190*(4), 369.

Freud, E., Macdonald, S. N., Chen, J., Quinlan, D. J., Goodale, M. A., & Culham, J. C. (2018). Getting a grip on reality: Grasping movements directed to real objects and images rely on dissociable neural representations. *Cortex*, *98*, 34–48. https://doi.org/10.1016/j.cortex.2017.02.020

Fuller, J. (1992a). Comparison of head movement strategies among mammals. *The Headneck Sensory Motor System. Oxford University Press, New York*, 101–112.

Fuller, J. (1992b). Head movement propensity. *Experimental Brain Research*, *92*(1), 152–164.

Gallup, A. C., Vasilyev, D., Anderson, N., & Kingstone, A. (2019). Contagious yawning in virtual reality is affected by actual, but not simulated, social presence. *Scientific Reports*, *9*(1), 294. https://doi.org/10.1038/s41598-018-36570-2

Goldring, J. E., Dorris, M. C., Corneil, B. D., Ballantyne, P. A., & Munoz, D. R. (1996). Combined eye-head gaze shifts to visual and auditory targets in humans. *Experimental Brain Research*, *111*(1), 68–78.

Henderson, J. M. (2016). Gaze Control as Prediction. *Trends in Cognitive Sciences*. http://www.sciencedirect.com/science/article/pii/S1364661316301929

Hessels, R. S., Niehorster, D. C., Nyström, M., Andersson, R., & Hooge, I. T. (2018). Is the eye-movement field confused about fixations and saccades? A survey among 124 researchers. *Royal Society Open Science*, *5*(8), 180502.

Holmqvist, K., & Andersson, R. (2017). Eye tracking: A comprehensive guide to methods. *Paradigms and Measures*.

Hooge, I. T., Hessels, R. S., Niehorster, D. C., Diaz, G. J., Duchowski, A. T., & Pelz, J. B. (2019). *From lab-based studies to eye-tracking in virtual and real worlds: Conceptual and methodological problems and solutions.*

Jacobs, O., Anderson, N. C., Bischof, W. F., & Kingstone, A. (2020). *Into the unknown: Head-based selection is less dependent on peripheral information than gaze-based selection in 360-degree virtual reality scenes*. PsyArXiv. https://doi.org/10.31234/osf.io/2qtcw

Kassner, M., Patera, W., & Bulling, A. (2014). Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 1151–1160.

Kingstone, A., Smilek, D., & Eastwood, J. D. (2008). Cognitive ethology: A new approach for studying human cognition. *British Journal of Psychology*, *99*(3), 317–340.

Komogortsev, O. V., Gobert, D. V., Jayarathna, S., & Gowda, S. M. (2010). Standardization of automated analyses of oculomotor fixation and saccadic behaviors. *IEEE Transactions on Biomedical Engineering*, *57*(11), 2635–2645.

Land, M. F., & Hayhoe, M. (2001). In what ways do eye movements contribute to everyday activities? *Vision Research*, *41*(25), 3559–3565.

Mathôt, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, *44*(2), 314–324.

Ohayon, S., & Rivlin, E. (2006). Robust 3d head tracking using camera pose estimation. *18th International Conference on Pattern Recognition (ICPR'06)*, *1*, 1063–1066.

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*(1), 195–203. https://doi.org/10.3758/s13428-018-01193-y

Pupil Invisible—Eye tracking glasses for the real world—Pupil Labs. (n.d.). Retrieved March 30, 2022, from https://pupil-labs.com/products/invisible/

Rai, Y., Gutiérrez, J., & Le Callet, P. (2017). A dataset of head and eye movements for 360 degree images. *Proceedings of the 8th ACM on Multimedia Systems Conference*, 205–210.

Risko, E. F., Richardson, D. C., & Kingstone, A. (2016). Breaking the fourth wall of cognitive science: Real-world social attention and the dual function of gaze. *Current Directions in Psychological Science*, *25*(1), 70–74.

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, 71–78.

SensoMotoric (2017). SensoMotoric Instruments. [Apparatus and software]. https://en.wikipedia.org/wiki/SensoMotoric_Instruments.

Silvis, J. D., & Donk, M. (2014). The effects of saccade-contingent changes on oculomotor capture: Salience is important even beyond the first oculomotor response. *Attention, Perception, & Psychophysics*, *76*(6), 1803–1814.

Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Masia, B., & Wetzstein, G. (2018). Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics*, *24*(4), 1633–1642. https://doi.org/10.1109/TVCG.2018.2793599

Solman, G. J., Foulsham, T., & Kingstone, A. (2017). Eye and head movements are complementary in visual selection. *Royal Society Open Science*, *4*(1), 160569.

Solman, G. J., & Kingstone, A. (2014). Balancing energetic and cognitive resources: Memory use during search depends on the orienting effector. *Cognition*, *132*(3), 443–454. https://doi.org/10.1016/j.cognition.2014.05.005

Stahl, J. S. (2001). Eye-head coordination and the variation of eye-movement accuracy with orbital eccentricity. *Experimental Brain Research*, *136*(2), 200–210.

’t Hart, B. M., Vockeroth, J., Schumann, F., Bartl, K., Schneider, E., König, P., & Einhäuser, W. (2009). Gaze allocation in natural stimuli: Comparing free exploration to head-fixed viewing conditions. *Visual Cognition*, *17*(6–7), 1132–1158. https://doi.org/10.1080/13506280902812304

Theeuwes, J. (1994). Endogenous and exogenous control of visual selection. *Perception*, *23*(4), 429–440.

Torralba, A., Oliva, A., Castelhano, M. S., & Henderson, J. M. (2006). Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychological Review*, *113*(4), 766–786. https://doi.org/10.1037/0033-295X.113.4.766

Unity Essentials. (n.d.). Unity Learn. Retrieved March 31, 2022, from https://learn.unity.com/pathway/unity-essentials

V120:Duo—An optical tracking system in a single, plug-and-play package. (n.d.). OptiTrack. Retrieved March 30, 2022, from http://optitrack.com/cameras/v120-duo/index.html

van Zoest, W., Donk, M., & Theeuwes, J. (2004). The role of stimulus-driven and goal-driven control in saccadic visual selection. *Journal of Experimental Psychology: Human Perception and Performance*, *30*(4), 746.

Vasser, M., Kängsepp, M., Magomedkerimov, M., Kilvits, K., Stafinjak, V., Kivisik, T., Vicente, R., & Aru, J. (2017). VREX: An open-source toolbox for creating 3D virtual reality experiments. *BMC Psychology*, *5*(1), 4. https://doi.org/10.1186/s40359-017-0173-4

Visbox, Inc. (n.d.). Retrieved March 30, 2022, from http://www.visbox.com/

Vizard | Virtual Reality software for researchers. (n.d.). Retrieved March 30, 2022, from https://www.worldviz.com/vizard-virtual-reality-software

Watson, M. R., Voloh, B., Thomas, C., Hasan, A., & Womelsdorf, T. (2019). USE: An integrative suite for temporally-precise psychophysical experiments in virtual environments for human, nonhuman, and artificially intelligent agents. *Journal of Neuroscience Methods*, *326*, 108374. https://doi.org/10.1016/j.jneumeth.2019.108374

Zangemeister, W. H., & Stark, L. (1982). Types of gaze movement: Variable interactions of eye and head movements. *Experimental Neurology*, *77*(3), 563–577. https://doi.org/10.1016/0014-4886(82)90228-X