Testing Methods to Shift Visual Attention from Wearable Head-Up Displays to Real-World Locations

Matt D. Ward

2017

A dissertation submitted in partial fulfilment of the requirements

for a degree of Doctor of Philosophy in Psychology

Acknowledgements

I would like to thank my three supervisors, Deak Helton, Mark Billinghurst and Paul Russell. Without your support and expertise this would have never been possible and I greatly respect your commitment to all your students, even during times of personal upheaval.

I would also like to thank the staff of both the University of Canterbury Psychology Department and the Human Interface Technology Lab NZ who supported me throughout the PhD process. Robyn, Jacki, Barbara, Sharyn, Ken and Rob; you all looked after me far better than I deserved. Despite my time being split between two departments, your doors were always open for me and so many other students and I appreciate everything you do for us.

Table of Contents

List of Figures

List of Tables

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Deputy Vice-Chancellor's Office
Postgraduate Office

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:
Chapter 3 of the thesis, Visual Cues to Reorient Attention from Head Mounted Displays
Is a reproduction of Ward, M., Barde, A., Russell, P. N., Billinghurst, M., & Helton, W. S. (2016, September). Visual cues to reorient attention from head mounted displays. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 60, No. 1, pp. 1574-1578). Sage CA: Los Angeles, CA: SAGE Publications.

Please detail the nature and extent (%) of contribution by the candidate:
Amit Barde assisted with the data collection (50%) and created Figure 3.1. He assisted with reviewing the paper for readability but did not contribute directly to the writing. Paul Russell and Mark Billinghurst contributed their supervision and guidance and assisted with the editing of the paper, each directly contributing changes affecting no more than 2% of the paper. William (Deak) Helton assisted in the same ways, but also supported me more significantly in expanding the initial idea for the paper and making practical changes to the experiment design to make it feasible. He provided editing and assisted in reducing the length of the paper to meet the publication guidelines. His direct editing contributions make up less than 5% of the final paper.

Certification by Co-authors:
If there is more than one co-author then a single co-author can sign on behalf of all
The undersigned certifies that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: Deak Helton Signature: *Deak Helton* Date: *16/07/2017*

Chapter 1: Cognitive resource models and the limits on attentional resources.

Mobile Computing

Modern smartphones, tablets and other portable computers are extraordinarily powerful devices. The ability to access the internet anywhere in the world and utilize specialised software from a device that fits in a pocket has changed how we work (Pitichat, 2013), how we travel (Wang, 2013), and how we entertain ourselves (Hollister, 2017). The size and portability of these devices come at a trade-off for computation power compared with traditional desktop systems. Desktop computers also have an advantage in terms of modularity, where components can be easily swapped or upgraded to match the requirements of their function. These benefits, along with the familiarity many users have with desktop computing peripheral devices and already established software, are why desktop computers are still the preferred device for workplace use in large businesses (Olivarez-Giles, 2017).

Despite these limitations, demand for mobile devices is large and still growing. Of the 2.4 billion devices expected to be sold in 2016, 80% are expected to be smartphones (Keizer, 2017). The major advantage of mobile device, as their category name suggests, is their ability to be used while moving. Desktop computers require a user to be in a fixed location to access virtual information and services and laptop computers, while easily movable, still restrict a user to a relative stable and flat surface during operation. Smartphones, tablets and other mobile computers can be held with a single hand, leaving a second hand free for operation. Moving while operating a device opens a range of possibilities ranging from simple to complex. A simple use case might be reading an article while walking home. An example of a complex use case is the crowd navigation tools being investigated for use at Disney World parks, where the movement of visitors can be tracked as to allow real time tracking of which rides are available and to move employees around to areas that are being underserved (Barnes, 2010).

Handheld mobile computers allow users to save time by simultaneously moving and operating a device but they also prevent many forms of task, particularly those that require a user to manipulate

their device alongside objects in the physical world. One hand stabilising the device means users have at most one hand available to move real-world objects and those actions must be made in serial, rather than in parallel with any virtual actions within the domain of their device. A solution to this issue is to move the device from a hand-held object to a wearable object. To date, wearable computers have mostly taken the form of objects people are already familiar with wearing, a watch on the wrist (Apple, 2017; Samsung, 2017), a piece of clothing (Nuñez, 2015) or the frame of a pair of glasses (Recon Instruments, 2015). By adding the functionality of a mobile computer to an object that is worn rather than held, the stabilisation hand is freed up, returning the use of both hands to the user. Wearables thereby allow users to undertake tasks where they must access virtual information while manipulating a real-world object with both hands and tasks where they must manipulate their virtual device and a real-world object in parallel.

Head mounted wearable computers (HMDs) add additional features to the potential of wearable devices. A HMD with a transparent display, such as the Google Glass (Google, 2017; Starner, 2013) or Alto Tech's Cool Glass (Hachman, 2016), can overlay virtual information on top of the physical world rather than have the user attend to a screen which occludes the environment. Wearing the device on the head also means the display can always stay within the user's field of view without physical manipulation of the device. For tasks where both hands are needed and the user may need to move their head to scan wide areas of the environment, HMDs allow users to still receive and use virtual information without interruption. This feature is currently unique to HMDs with a transparent heads up display. Exercise and navigation apps make particular use of the hands-free and always-in-view features. For example, the Recon Jet app library features programs to display biometric, navigation and performance feedback for cyclists, skiers and recreational pilots. These are all users who while performing their tasks will have limited opportunities to have hands free to reposition or interact with the device.

Overlaying information on top of the user's vision also increases the speed at which important information can be displayed to a user. To retrieve an incoming message using traditional desktop or

laptop systems requires significant set up time if the user is not already operating the device and can be easily missed. Mobile phones can also cause delayed responses to incoming messages, and some messages will be missed completely. Smartphones are placed down out of reach (further than 2 meters) approximately 40% of the time they are turned on (Patel et al., 2006) and even when the device is available to the user, users frequently miss calls due to either not detecting the phone's ringtone or vibration or because the user fails to retrieve the phone from their pocket or bag in time (Cui, Chipchase & Ichikawa, 2007). Transparent display HMDs, by comparison, can turn on their display and have the incoming message notification immediately within the user's view, reducing device handling delays and reducing the chance an important message will be missed.

The new features of wearable computers and head-mounted devices in particular come with some significant costs. The unique features of wearables center on enabling multitasking and performing tasks in parallel means the user needs to allocate and prioritize the allocation of mental resources to both tasks.

Models of Mental Resources

Early models of divided attention were built around the concept of a perceptual bottleneck. In these models, multiple stimuli were thought to be perceived by the human participant but somewhere in the sequence of processes occurring between perception and response execution there existed a module that was limited to processing one input at a time, which slowed the entire system. Prior to this bottleneck, stimuli are processed in parallel, and after this bottleneck, stimuli are processed in series. Donald Broadbent's filter model was one of the earliest of this family of models. This model assumed multiple incoming stimuli could enter a mental sensory store, also called sensory memory (Broadbent, 1957; Broadbent, 1958).

From the sensory store, information was passed through a selective filter which concerned physical properties of the stimulus, such as loudness or brightness or location of source. Some cognitive control of this filter was possible, with the human subject being able to selectively filter

some, but not all, unwanted information. For Broadbent's Filter Model, this was the stage that the bottleneck occurred. The model stated that only one stimulus could pass through the filter at any given time. Divided attention performance detriments were accounted for by participants alternating the targets of their filter, a process that requires effort and relies on the key components of the stimulus being stored in the sensory memory long enough for the signals to be processed in serial.

The next module in the model was one where the incoming stimuli were assessed with higher level of cognitive processing. At this stage, the stimuli were screened for abstract meaning, coherence, and perceived value for the purposes of deciding which information gets passed to the final module, working memory. From the working memory, the processed stimuli could then be encoded into long term memory or used to select an appropriate response.

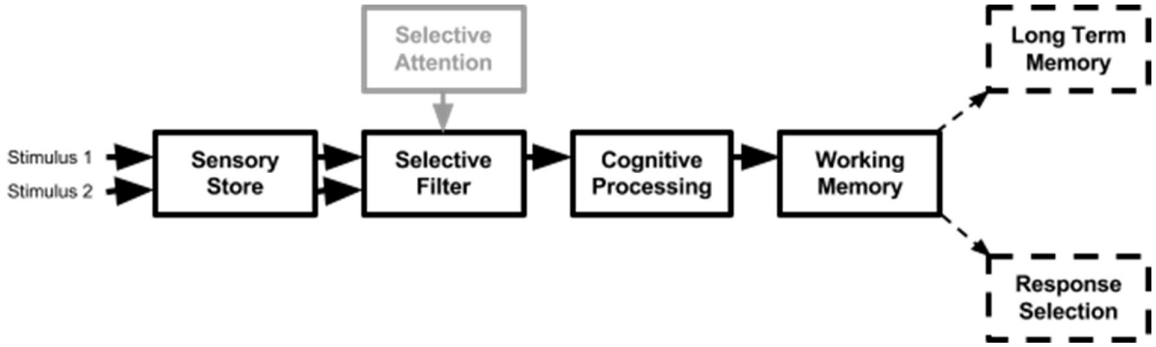


Figure 1.1: Broadbent's Filter Model

Evidence for this model can be found in behaviour. When two stimuli are presented at once, often only one is perceived and when both are noted, the responses to the events are usually in serial. For example, in a study where participants were required to respond to either a light flash or a tone, 50 conflict trials where both stimuli were presented. In 49 of these trials the subject responded to the light alone, and after the one exception, the subject apologised and explained he had accidentally left a switch closed from the previous trial. Furthermore, subjects were unaware that the tone had even been presented on 16 of these trials (Colavita, 1974).

Initial controversy around this model centered on a debate over where the bottleneck actually occurred. Broadbent's Filter Model is notably unable explain the 'cocktail party effect' where a human subject is able to listen to one conversation, filtering out all others (although this ability does breakdown at a point), but still can attend to the sudden mention of their name or a high-interest keyword in another audible conversation. Because it is the higher level cognitive processes that scan a message for content, this means that at least some kind of cognitive analysis precedes the bottleneck. Deutsch and Deutsch (1963) investigated the effects of message importance and concluded that the behavioural evidence showed that filtering unwanted messages required "discriminatory mechanisms of as great a complexity as those in normal perception". This means the high level cognitive processing module proposed in Broadbent's original model needs to be active before the selective filter.

Capacity models of divided attention are an attempt to describe and predict how tasks performed concurrently cause mutual interference with one another, while also allowing for the observed possibility that human subjects can prevent the detriment to one task by reducing performance on the others. In the filter model, this interference is caused by two signals needing to be processed concurrently. In the capacity models, it is caused by the global cognitive demands of both task exceeding the total capacity available in the moment. Yet, despite the difference in the expected cause of multi-task detriments, the capacity models and bottleneck models are not necessarily mutually exclusive. Notably it can be logically proven that at least one bottleneck must occur in human responding in some tasks as the human tongue can only produce one sound at a time and therefore verbal responses at least must be made in serial. Similarly, hands can only occupy a single location in space at a time. These observable limitations mean that bottlenecks in resources must be considered alongside any capacity model.

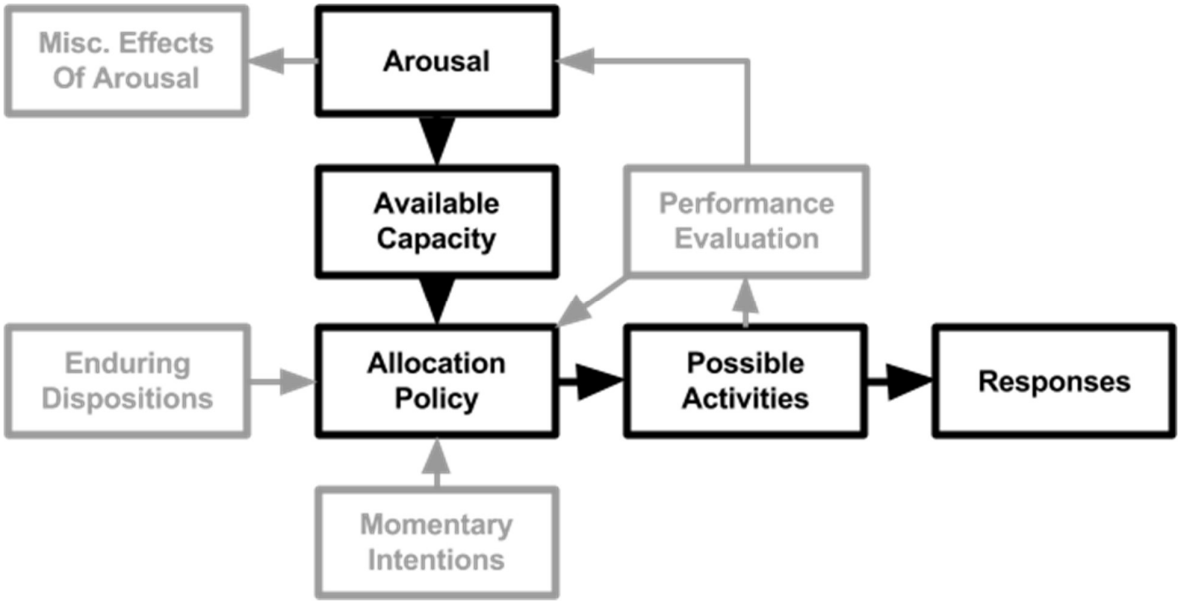


Figure 1.2: Kahneman's Capacity Model. Adapted from Kahneman (1973), Figure 1.2.

Kahneman's Capacity Model (1973) was developed after the finding that the primary determinant of a human subject's arousal was the difficulty of the task or tasks they were currently engaged in (Kahneman, Peavler & Onuska, 1968). This correspondence between arousal and task demands has subsequently been supported for arithmetic (Bradshaw, 1968a); short-term memory tasks (Kahneman & Beatty, 1966); pitch discriminations (Kahneman & Beatty, 1967); standard tests of "concentration" (Bradshaw, 1968b); sentence comprehension (Wright & Kahneman, 1971); paired-associate learning (Kahneman & Peavler, 1969); and imagery tasks with abstract and with concrete words (Simpson & Paivio, 1968).

In Kahneman's Model the relationship is between task demand and arousal is causal, but mediated such that increased arousal causes an increase in available capacity, the total pool of mental resources. Figure 1.3 shows the predicted effect of capacity demand vs capacity supply. As needed capacity increases, so too does the capacity supplied, but at a slower rate. The disparity between these increases causes spare capacity, which is used for other tasks like environment monitoring, to become constrained and eventually begins to impact performance on the primary task.

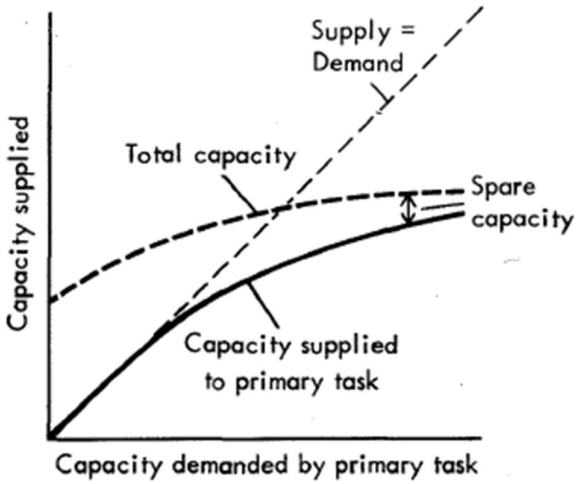


Figure 1.3: Predicted relationship between capacity demand and capacity supply. Reproduced from Kahneman (1973), Figure 2.1.

Kahneman's Model also includes an allocation policy which divides mental resources between possible activities. This allocation policy was based on "enduring dispositions" which included learned preferences and innate preferences such as the human adult's preference for colour, curve complexity and objects that appear to be the foreground based on other visual cues, and "momentary intentions." which includes the effects of priming, expectancy, or the intention to prioritize one task over another. This allocation policy is then further altered by a feedback loop which evaluates performance and uses this evaluation to both reprioritize tasks and increase arousal and consequently the available capacity.

Tasks which are allocated insufficient resources suffer in terms of performance, increasing response times or error rates. Like the allocation of resources to a task, the resources demanded by tasks can vary from moment to moment. Other sources of task complexity include the discriminability of target stimuli, the amount of motion of the head or body needed for detection or response execution, and time limits set on responding. In particular, any task requiring information to be held

in the short-term working memory naturally imposes such time limits because the participant's rate of activity must now exceed the rate at which items are lost from short term memory.

This model of human multitasking performance aligns well with Rasmussen's Skills, Rules, Knowledge (SRK) model (Rasmussen, 1983) which proposes a tripartite classification of skills. Overlearned tasks which require little to no effort to process, like tapping fingers to a beat, fall within the Skills category. Knowledge tasks, requiring integration of information and holding information in short-term memory, are predicted to be the most difficult class of tasks by both theories while rule based tasks fall between the two.

Kahneman's model had reasonable success at predicting effects of dual tasking on the individual component tasks but the model does not specifically state what resource is actually being divided (Kahneman 1973). Additionally, over the next few years' evidence was found that the patterns of dual task decrements could not be explained by the combination of task resource demand and the participants' allocation policy alone. Notably, tasks which used separate sensory units or cognitive skills had less interference than in conditions where tasks used overlapping ones (Kantowitz & Knight, 1976; Wickens 1976).

A meta-analysis by Christopher Wickens (1980) found evidence that there was a strong separation between certain internal mental structures like those involved in visual vs auditory processing and noted that they behaved like separate resources as opposed to manifestations of a single resource. These separate resources have the advantage of being able to explain behavioural differences in dual-tasking efficiency while also mapping to neurophysical entities to give a stronger definition for proposed resources.

Further development of this new multiple resources theory led to the development of the multiple resources model which has four dimensions that overlap in various configurations (Wickens, 2002). Figure 1.4 is a visualization of the predicted resources.

The first of the four dimensions is the 'Stage'. There are three levels of this dimension and they occur in order, starting at perception where the incoming stimuli are processed, then cognition,

where the processed stimuli are analysed, and finally responding, where the appropriate response is decided and performed. The second dimension is the 'Processing Code' which has two levels referring to the type of information the stimuli carries; spatial information or verbal information. This dimension intersects with the 'Stages' dimension at all levels.

The third dimension, 'Perceptual Modalities', or 'Modes' divided the stimuli into categories based on the sensory organ used to detect them. While not included in the model as described by Wickens (2002), other senses, such as tactile signal detection, could hypothetically be included in the model. The "Perceptual Modalities' dimension divides stimuli in the perception stage, with the information being combined upon reaching the cognition stage.

The fourth and last dimension further divides visual perception into focal and ambient 'Visual Channels'. 'Focal' visual stimuli come from within the three-degree cone of clear visual perception for human adults, and the remaining 'Ambient' come from the remained of the peripheral visual field.

<table>
<tr><td colspan="2"></td><td colspan="3">Stages</td></tr>
<tr><td>Codes</td><td>Modes</td><td>Perception</td><td>Cognition</td><td>Responding</td></tr>
<tr><td rowspan="2">Spatial</td><td>Auditory</td><td colspan="2">Auditory S</td><td rowspan="2">Spatial Cognition</td><td rowspan="2">Manual/Spatial Responding</td></tr>
<tr><td>Visual</td><td>Focal VS</td><td>Ambient VS</td></tr>
<tr><td rowspan="2">Verbal</td><td>Visual</td><td>Focal VV</td><td>Ambient VV</td><td rowspan="2">Verbal Cognition</td><td rowspan="2">Vocal/Verbal Responding</td></tr>
<tr><td>Auditory</td><td colspan="2">Auditory V</td></tr>
</table>

Figure 1.4: Multiple Resources Model. Adapted from Wickens (2002).

Competition for Perceptual Resources

Using the Wickens' model, it is possible to estimate where the major conflicts over allocation of mental resources will arise when using the HMDs. First and foremost, HMDs add visual information

to the user's perceptual space which increases the demand on the focal visual perception components of Wickens' Model. Posner, Snyder and Davidson (1980) described focal visual attention as a spotlight that improves detection efficiency for objects and regions within the beam. Eriksen and St. James (1986) extended this model by showing the 'spotlight' could be stretched to account for uncertainty as to where an object of interest may appear, or contracted to reduce the likelihood of incorrectly attending to distractors (see also Jonides, 1983; LaBerge, 1983). Changes to the size of the attentional field are inversely related with the sensitivity of the visual system to detect changes within it. Sensitivity decreases progressively outside the region of focal vision attention on a smooth but not necessarily linear gradient (Andersen, & Kramer, 1993; Eriksen & St. James, 1986; Henderson & Macquistan, 1993; Mangun & Hillyard, 1988). Andersen & Kramer (1993) further present that the vertical and horizontal sensitivity gradients are not equal. They found that the shape vertical gradient was steeper, making the area of attention a flattened ellipse rather than a circle.

The ability to adjust the sensitivity gradient of visual attention allows a target to more easily be detected within a cluttered environment (Eriksen & Hoffman, 1972). Eriksen and St. James (1986) found that response times increased as the number of cued possible target positions increased. This means that performance will be impaired if the region of attention is wide enough to include distractors while centered on the target. Tightening the focus of the attention region helps mitigate these costs. Shrinking the region of visual attention can create a new issue, however. When a cue directs attention to a region that does not contain the target, there is a performance cost larger than when a cue is absent entirely (Posner, Nissen & Ogden, 1978). When cues are unreliable or imprecise, it is more likely for targets to fall outside the region of visual attention when a cluttered visual field causes a restriction of this region. This is particularly relevant for HMDs which are designed for use by a mobile user in a real-world environment, where no cue will be perfectly precise or reliable. Adding visual clutter through HMDs may make a user more likely to miss targets close to, but not directly at, the position of a guiding cue that would have been otherwise detected by a wider lens of visual attention. Supporting this, Lavie and Tsal (1994) proposed as part of their model determining the stage

in cognition at which visual information is filtered that in addition to an upper limit of internal mental resources that can be allocated to visual attention there is also a lower limit. When perceptual demands are high and demand exceeds capacity for visual attention resources, little to no processing of peripheral stimuli occurs. When perceptual demands are low, the excess resources are diverted to at least partial processing of surrounding irrelevant stimuli.

More generally, focussing visual attention on a particular component of a scene (Simons, & Chabris, 1999) or one out of two superimposed scenes (Neisser, & Becklen, 1975) with the intent to filter out the remainder causes inattentional blindness; a phenomenon where irregular and highly salient events are missed despite occurring in the same perceptual space to which the participant's attention is allocated. Assuming no special interactions between virtual and real-world information, this means that adding more information to the user's perceptual space through a HMD will increase clutter within the user's visual field. If this tightens the focus of the user's visual attention, the user may be at higher risk if missing unexpected events in the real world. This would impair task performance on whichever task was being augmented by the HMD and in the worst cases pose a physical danger to the user.

Investigations in to the allocation of visual attention in three-dimensional space proved the analogy of a spotlight to be incomplete. Instead of continuing unbroken until colliding with a surface like a ray of light, visual attention has been repeatedly shown to center on a particular depth point within a participant's visual field (Andersen, & Kramer, 1993; Atchley, Kramer, Andersen, & Theeuwes, 1997; Finlayson, & Grove, 2014; Parks, & Corballis, 2006; Theeuwes, Atchley, & Kramer, 1998). The sensitivity gradient extends forwards and backwards in depth from the center of focal attention in the same manner as it does horizontally and vertical, creating a volume of focal attention.

The sensitivity gradient is not necessarily symmetrical in depth. Several studies instead find support that visual attention volume is "user-centric"; that the sensitivity gradient is shallower when extending back towards the user (Andersen, & Kramer, 1993; Finlayson, & Grove, 2014; Parks, & Corballis, 2006). This means it is easier to detect information or objects that appear unexpectedly

between the user and the center of their focal attention, but also that users are more likely to be distracted by irrelevant distractors which are displayed close to their person. Therefore, as HMDs sit close to the participant's eye, it is likely HMDs will be able to grab a user's attention even when they are attending to the real-world environment but unnecessary updates or messages may be a hazard, pulling attention away from the environment unnecessarily.

Support for the viewer-centric property of visual attention has not been universal. Many studies have failed to find differences in responses to distractors on depth planes closer to the participant than further (Andersen, 1990; Atchley, Kramer, Andersen, & Theeuwes, 1997; Theeuwes, Atchley, & Kramer, 1998). At the current time, the evidence for a user-centric visual attention volume is stronger in the opinion of the author. The studies by Atchley, Kramer, Andersen, & Theeuwes (1997) and Theeuwes, Atchley, & Kramer (1998) only use two depth planes in their experiments which give only a limited measure of the attentional gradient forwards and backwards from the center of attention. The limited number of planes may also have encouraged a strategy wherein participants set their attention beyond the priming cue in depth. If it is easier to detect changes and objects closer to the user, then such a strategy would serve to balance out response times to near and far positioned targets and would be undetectable without more depth options. By comparison, the study by Andersen, & Kramer (1993) which supported user-centric gradients uses seven depth planes allowing a more fine-grained analysis. The studies which failed to find user-centric gradients also all use behavioural outcome measures whereas the research conducted by Parks, & Corballis (2006) adds electrophysiological evidence and evidence from multiple domains is stronger than evidence from only one.

Posner, Nissen and Ogden (1978) conducted a study wherein participants were required to detect whether the abrupt appearance of luminous stimuli within a dark field could be responded to more quickly if the participant knew where the stimuli would appear. During their analysis, the authors investigated a subset of trials in which participants did not move their eyes from the position of the fixation cue which preceded each trial. The participants gained an advantage in terms of response

speed when the target appeared at the cued location and a penalty when it instead appeared at an unexpected location despite the lack of an overt shift of attention during these trials. This is evidence that humans can make 'covert' shifts of attention and are not limited to attending only to foveal region in the center of their visual field (Posner, 1980). Instead humans are able to center their locus of visual attention on any position within their 3-dimensional visual field to increase the discriminability of stimuli within (Carrasco, & McElree, 2001; Downing, 1988; Nakayama & Mackeben, 1989).

The process of moving covert attention is faster than typical saccadic latencies for eye movements (Carrasco, & McElree, 2001; Nakayama & Mackeben, 1989). Research by Hoffman & Subramaniam (1995), and later, Shepherd, Findlay and Hockey (2007) found that instead of eye movements leading visual attention, it is the covert visual attention which leads saccadic movement around the perceptual space. Eye movements and foveal attention are still beneficial on tasks where detailed properties of an object must be discerned, such as reading words and extracting their meaning (Latham & Whitaker, 1996), but is unnecessary for detection tasks (Posner, Nissen and Ogden, 1978) and tasks when discrimination requires only salient features such as colour and orientation (Carrasco, & McElree, 2001; Downing, 1988; Nakayama & Mackeben, 1989). As one example of this, in research by Nakayama & Mackeben (1989) participants were asked to search for and report the presence of an irregular target within a 64-element display after a variable presentation duration while maintaining a center fixation on a marked point. Eye movements were monitored to enforce this fixation and trials in which a participant failed to maintain fixation were automatically excluded. When the position of the target was cued and participants could pre-emptively allocate their covert attention, reactions times significantly improve for trials in which the elements were displayed for more than 100 ms. In a single session where a participant was not required to maintain fixation and thus could deploy overt, foveal attention, showed only minor improvements in performance and then only on trials with durations longer than 600ms. The authors conclude from this that there was no special advantage gained by foveation in their task.

Remington and Pierce (1984) showed that facilitation for a centrally cued peripheral target developed at the same rate for laterally near or far targets. Sperling and Weichselgartner (1995) found similar results, and concluded that the duration of a shift of attention is independent of lateral distance and interposed distractors. This is because covert visual attention is extinguished at one location and re-established at the destination with no travel through the space in between. Travel through the perceptual space is not required as the initial attention movement is the covert attention shift. Covert attention is directed mentally and therefore is not subject to the limitations of physical movement, where muscles needs to pull a body part from one position to another through all intervening points.

If the time taken to remove and re-establish covert attention is independent of eccentricity between two visual locations, this creates an odd corollary for cases in which the angular distance to travel is zero. Hypothetically the costs of shifting attention between two points should also then apply to shifting attention between two objects occupying the same space. To investigate a person's ability to divide attention, Rodrıguez, Valdes-Sosa & Freiwald (2002) super imposed two circular objects marked with a semi-transparent surface consisting of 100 small shapes, circles for one object and squares for the second. The two objects then rotated in opposite directions to distinguish themselves as separate perceptual objects. In two of three experiments, colour was also used to differentiate the two objects' surfaces. When participants were asked to report a property from each of the two surfaces, the second property was reported with significantly less accuracy, demonstrating a cost when dividing attention between the two surfaces. This occurred regardless of whether a participant was reporting the same property for both surfaces or a different property for each surface. When two different properties were to be reported for the same surface there was no decrease in accuracy for the second property (see also Ernst, Palmer & Boynton, 2012).

Current generation HMDs such as the Google Glass (Google, 2017; Starner, 2013) or Alto Tech's Cool Glass (Hachman, 2016) have a monocular display which prevents the use of parallax to create illusions of depth. Future devices may have screens for each eye and allow for displaying

information at different depths. This would allow the creation of conformal virtual scenes where virtual objects remain anchored to a particular position in three-dimensional space regardless of how the device displaying the virtual information is moved or oriented. But even if information could be placed conformally with the real-world environment such that the distance between real-world objects of interest and the virtual information needed to assist with interactions with that object to zero, there will still be a cost of dividing attention between environment and device. This sets a minimum cost for switching attention between environment and device when augmenting user behaviour with HMDs. There is still potential for improvement, however. The complexity of virtual information displayed will be dependent on the complexity of the task to be augmented. There will be many situations in which the information cannot be encoded in simple detection tasks or single-salient feature stimuli. Users will need to make both a covert and overt attention shift in these situations. While covert attention movements are independent of start-to-end eccentricity, saccadic eye movements and therefore overt attention are not. Instead there is an approximately linear relationship between saccade amplitude and saccade duration (Abrams, Meyer & Kornblum, 1989).

The benefits of minimising eccentricity between virtual and real information can be seen in applied studies for drivers in which heads up displays (HUDs), where virtual information is overlaid over top of the environment, were compared against head down displays, where information is displayed in a separate window or region of space from the environment. In these studies, HUDs allowed for more stable speed maintenance (Liu & Wen 2004), faster responses to unexpected road events (Horrey & Wickens, 2004; Liu & Wen 2004), and shorter fixations on the devices (Ablaβmeier et al., 2007 & Rigoll, 2007) compared with dashboard HDDs during periods of high attention demand. Yoo et al (1999) investigated the optimal position for windscreen projected HUD elements when participants were sitting in a car interior replica and watching a video tape of a real road scene. Participants were asked to detect and respond to several types of road events as well as amber coloured warning triangles in the HUD. On average, response time to HUD warnings increased with distance of the position of the HUD element from the center of the screen. The exception to this

pattern was when the HUD warnings appeared in the direct center of the participant's visual field and blocked view of the road ahead. Current generation HMDs are unable to display information in the center of a participant's vision, but the potential of overlaid virtual information to block the view of objects in the environment will also need to be considered as the technology develops.

In summary, there are many risks for wearables at the perception stage of Wicken's Multiple Resource Model. Increasing the amount of visual clutter within the user's perceptual space can narrow the spread of their attention, making users less likely to notice objects of interest that fall outside the current field of attention. Additionally, more visual information increases the demand on visual resources and makes it less likely HMD users will passively scan the other objects within their field of view. This puts them at risk of failing to notice unexpected events in their environment, even if the event is unusual or potentially dangerous. When dividing attention between two perceptual objects, even when they share a location in space, each reallocation of attention requires a shift of covert attention. This creates an avoidable cost to task performance with each shift. These costs can be minimised by reducing the number of attention shifts required. A large number of HMD programs will also feature detailed information which requires overt attention. Saccades of the eye follow covert attention shifts and are affected by eccentricity between the previous and current target attention locations. Because overt attention shifts take a longer time than covert attention shifts, performance costs will be higher. Covert attention shifts set the minimum cost of an attention shift while overt attention shifts set the maximum. The costs created by overt attention shifts can be minimised by accurate and effective cues and by matching the positions of virtual information to the positions of objects in the environment for which that information is relative. Overt attention shift time is minimised by making the position of virtual information conformal to the environment, but virtual information should not be positioned directly on top of the real-world objects if possible as doing so will obscure the real-world objects and undercut task performance.

Competition for Cognitive Resources

HMDs affect the second stage of Wicken's Multiple Resource Model, the cognition stage, in at least three ways; overloading cognitive resources needed for the primary task, causing cognitive tunnelling, and increasing demands on central executive decision making by complicating the users' internal resource allocation policy. There is some debate whether cognitive resources are shared in parallel between two concurrently performed tasks, as described in the resource models of Kahneman (1973) and Wickens (2002), or instead limited to a single task at a time, creating a bottleneck leading to serial responding to the two tasks (Pashler, 1994a, Pashler, 1994b). The details of this debate are beyond the scope of this thesis because in an applied setting the outcome is the same; if two tasks are performed at once at least one task will have reduced or delayed access to cognitive resources. The more resources that are shared between the tasks, the greater the impairment will be. For example, a study by Sims and Hegarty (1997) involved a task in which a participant must predict the motion of an object in a diagram after force had been applied to an attached pulley system. As a secondary task, participants were asked to also remember a set of 3 dot positions on a 4x4 grid and report at a later time if a second set matched the memory set (task adapted from Kruley et al., 1994), or to remember a string of 6 consonant letters and at a later time report if the set contained a particular letter (task adapted from Sternberg 1969). Attempting to hold the spatial information of the dot patterns in memory caused greater interference with the force prediction task than the letter memory task as measured by the proportion of errors in both the primary and secondary tasks. This is because two spatial tasks share more resources than a spatial and verbal task.

One solution to overloading particular cognitive resources is to redesign concurrently performed tasks such that there is less overlap in demand. This is not a practical solution when augmenting behaviour, however. This is because to integrate task-related information from a virtual space with real-world information the information must, at least in its final stages, be processed in the same form. Furthermore, guidance through instructions for a task is more effective when the description of the procedure is in the same form as the result. For example, a spatial reasoning task

such as folding origami shapes is completed more accurately when the steps are also displayed spatially as images rather than given linguistically as written instructions (Novick & Morse, 2000). Before the written instructions can be used they must be converted into a spatial understanding of the action and this conversion increases mental load and adds opportunities for errors. This means to best aid a task, any supporting information should be displayed in the same form as the task. This means an overlap in the demand for cognitive resources is unavoidable. A second solution to the overload problem is to decrease the load of the primary, real-world task or the secondary, virtual space task. Ideally the augmenting device will reduce the cognitive load of the primary task by simplifying decisions by adding difficult to detect or otherwise invisible information or performing measurements that would otherwise require processing by the user. The amount of load reduction to the primary task will depend on the amount and quality of information displayed on the virtual device but this comes at a trade-off of complexity on the secondary task. The denser and more complex the virtual information given, the more resources will be needed to process it, and the more decisions about which information should be used will need to be made. The base mental load and the value of augmenting information will differ for each task to be augmented. This means that the balance between the improvements added by increased amounts of virtual information and the costs to performance caused by the need to process the virtual information will need to be asses individually for each task.

Beyond overloading a particular cognitive resource, a general increase in cognitive demand caused by an increased number of decisions about which source of information to attend to can cause cognitive tunnelling. Cognitive tunnelling is a phenomenon where participants lock their attention on one object or goal longer than is needed and to the detriment of overall performance (Thomas, & Wickens, 2004; Wickens & Alexander, 2009). Cognitive tunnelling is a result of increased decision-related stress and can be predicted using physiological measures such as a participant's pulse rate (Dirkin, 1983; Wickens, 1996). The increased fixation time on individual goals or objects can be measured in some cases by the number and type of eye movements. Users suffering from stress and

consequently cognitive tunnelling make fewer, smaller eye movements compared to when they would otherwise undertake the task with a lower load (Reimer, 2009; Sohdi et al., 2002). While the lack of attention flexibility can be measured by assessing eye movements, cognitive tunnelling is not a form of perceptual interference. Even when the number of visual stimuli presented to the participant is unchanged, cognitive tunnelling can be induced and its effects on eye movements monitored. For example, Reimer (2009) investigated the threshold for the onset on cognitive tunnelling while driving by manipulating the complexity of a secondary audio task without adjusting the clarity of the audial presentation. Without significant changes to the amount of visual or audial stimuli delivered to the participant these changes in the difficulty of the audial task induced cognitive tunnelling symptoms in the participants once the mental workload exceeds the participant's ability.

The effects of cognitive tunnelling can be reduced or eliminated by lowering the mental workload of participants. When designing augmentation software, any display that decreases the number of decisions or simplifies otherwise complex decisions should already accomplish this and help offset the cost in task complexity added by the introduction of the supporting device. In a study of HUDs for pilots conducted by Ververs & Wickens (2000), a reduction in the effects of cognitive tunnelling has also been found when the augmented display is conformal and anchored to real-world objects compared to when the display is only partially conformal with the depth of the virtual information untied to the real-world environment. The authors argued the partially conformal display suffered because it gave no opportunity for a preview of upcoming course changes. Therefore, additional reduction of cognitive tunnelling effects may be applied by allowing the participant access to upcoming events and consequently reducing the number of in-the-moment decisions that need to be made by the participant. Adding future event or predictive information will come at a trade-off cost with the other types of interference discussed so far, however.

The addition of virtual information into the user space also creates an increased demand on the central executive functions of a user's cognition. The central executive functions apply to the resource allocation and performance evaluation loop of the capacity models of mental resources

(Kahneman, 1973). The human mind constantly compares current performance against momentary priorities and by adding additional channels of information the user must make more choices about which sources of information to attend to and how much perceptual and cognitive resources should be devoted to each task. The greater the number of options available the greater the number of opportunities for error exist. Beyond the unintentional effects of more information described already, users may intentionally and voluntarily apply more attention to a device than is optimal for task performance. Users who rely heavily on an assisting device to augment their behaviour suffer significant performance losses the first time their device suffers a failure or an unexpected event is not accounted for by their device (Wickens, 2017). This is because participants overestimate the reliability of augmenting systems and become complacent (Wickens, Sebok, Li, Sarter, & Gacy, 2015). In some tasks, the effects of complacency and overreliance on assisting devices can be mitigated by giving multiple response suggestions and then forcing the user to make the final assessment, or by displaying information on the current situation rather giving a specific response recommendation to the user or taking action automatically (Onnasch et al., 2014; Wickens, 2017). For HMDs, augmentation programs will often be used in the real world where the environment cannot be controlled. In these situations, the augmenting program can never be completely reliable and some unexpected events will have the potential to be physically harmful to the user. For these reasons preventing over-reliance and complacency with an assisting device will be important.

Competition for Response Resources

In the final stage of Wicken's Multiple resource model there are two types of response conflict; conflicts within the use of the device and conflicts between responding to the device and to the environment. Current generation HMDs (Google, 2017; Hachman, 2016; Recon Instruments, 2015) have two primary methods of inputting information back in to the device. Users can use voice commands which can be searched for keywords and parsed into text or numbers, or the user can make manual inputs using the small touchpad on the side of the HMD's frame. For the current generation

HMDs, these touchpads only accept a limited range of inputs; horizontal and vertical swipes, taps, squeezes and stretches. Both the verbal and manual input for these devices are limited to one action at a time making all input to the device serial for each of the two independent channels. Actions will also be performed serially when the user must make a response on the device and the environment through the same channel. Humans can only vocalise one word or phrase at a time and each hand can only be in a single location at once. Additionally, there is some interference between response channels when users must make an independent response with each hand, although this can be mitigated with practice and by making the responses required for each hand match natural or trained behavioural affordances (Duncan, 1979). When responses are made on separate channels, for example manually and vocally, there are delays caused by competition in the cognitive response selection stages but the actual responses themselves can be made in parallel.

Thesis Scope and Goals

Because the response stage conflicts are primarily caused by physical rather than mental limitations for the user and hardware rather than software limitations for the device, improvements for this stage of the multiple resources model are beyond the scope of this PhD. Of the two remaining stages, recent research has suggested it is the conflicts at the cognitive stage that impose the greater impact to user dual-task performance in applied settings (He, McCarley, & Kramer, 2013; Sawyer et al., 2014a). If the goal is to improve the performance of users of augmentation software for HMDs, investigating design tools to minimise conflicts at this stage would appear to have the most value. The issue with designing for the cognitive stage conflicts is that they are highly task dependent. Every task will have a different level of base complexity and consequently a different amount of information that can be added before a user becomes overloaded. Likewise, each situation will have a different trade-off between the amount of information that can be given vs the value of that information for actually making task related decisions. Tools for reducing perceptual conflicts are broader in their application. For example, a type of cue which can lead attention from the device to the environment for the

purposes of helping draw attention to an unexpected real-world event more quickly could be used regardless of the context. While perceptual conflict reducing design rules may give less advantage to performance than methods which focus on cognitive cost reductions, their wide application makes them still a valuable focus of research. This body of work will look at a small subset of tools which can be used to mitigate perception stage conflicts. The focus is on efficient methods of guiding attention to the correct locations such that the device to environment switching costs and the effects of visual clutter and near field capture of attention are minimised.

The first experiment will establish that perception stage conflicts do exist for the Google Glass when used concurrently with a primary task requiring sustained visual attention. The second experiment will confirm that the users of HMD devices do treat the virtual display as a separate perceptual surface to their environment. This will mean subjects encounter unavoidable performance costs every time attention is moved to and from the device. Experiment three will investigate the possibility of using a new type of reflexive orientation cue to replace the exogenous attention directing cues (Posner, 1980) which are unavailable in situations where the software designer has no control over the environment. The results of this study will find that accessing a natural visual reflex is neither sufficient nor necessary for creating an effective attention directing visual cue. Experiment four will then test several non-reflexive replacements for exogenous cues delivered via HMD in a simple search task. The best of these cues will then be adapted to account for environment depth and free user movement in the fifth experiment. This experiment will involve a highly realistic situation where a participant must navigate a real-world environment, avoiding physical obstacles while making their search. A final sixth experiment will repeat the parameters of the fifth with addition conditions comparing participant performance with the augmenting device to situations in which a subtle and a highly salient environment-based cue are available.

Chapter 2: The effect of delivery mode on the dual-task interference of receiving information from a Google Glass.

Preamble

The central idea behind this PhD thesis was to explore how the Google Glass and other HMDs could be used to aid performance, rather than just act as a distractor. Regardless of this however, the distraction created by the device and costs of dual tasking are inevitable. For the device to aid performance on a real-world task, the user will need to attend to both the device and the environment. Given the previous research on dual tasking and heads up displays presented in the previous chapter, this means we expect both perceptual and cognitive interference when a user operates the device, regardless of whether the device is aid or distracting from the primary task.

For our first experiment, we choose to focus on the perceptual costs of using a HMD. The reason for this is that the end goal of the PhD is to find information that is generally helpful to designers for HMDs. The cognitive costs of divided attention are task specific, rather than device specific. For example, a method to make a body of text more easily parsable to a reader would lower the time and effort needed to do so. This would reduce the cognitive costs of a task but would do so regardless of whether the text was delivered through an HMD, a regular screen, or written on a page. The same method would have no use on in a different HMD program which delivered no text. The effectiveness of the design tool is dependent on the task rather than the device.

By contrast, the perceptual costs of dual tasking are determined by the method of information delivery rather than the type of information itself. Visual information delivered by the Glass, regardless of content, will require users to shift their attention to the device, expand their volume of attention in space, or process it peripherally. Audio information will require users to filter relevant information from the HMD out from any noise in the environment. Spatialised audio will require participants to estimate the direction and position of the sound source in the environment regardless of why they need to locate that sound source. The limitations of the HMD device create the perceptual costs, and

so consequently, methods to reduce the impact of these costs can be used broadly in programs that look to augment human performance with these devices.

This purpose and main goal of this PhD thesis therefore requires a perceptual cost of attending the device to exist. These first two experiments work to show that a perceptual cost, one separable from task-dependent cognitive costs, of divided attention exists. To do so we chose a task which would require constant attention from the participant and then measured how long an unexpected piece of simple information distracted the participant from the main task. We had an additional theory of which piece of the perceptual costs of HMD use we wished to improve upon over the course of the PhD thesis: the cost of shifting from the device screen on the depth plane close to the eye out to the environment on a more distant depth plane. We expected this shift of visual attention to take longer than the inward shift because of the asymmetry in how attention drops off in depth (Andersen, & Kramer, 1993; Finlayson, & Grove, 2014; Parks, & Corballis, 2006). We added a warning sound to a condition in the experiment to allow the participant to pre-emptively move their attention to the device, and thus split the measurement of the inward and outward attention shifts. If the outward attention shift is indeed delayed, it provides a target with greater potential for improvement through design principles.

Introduction

Augmenting a person's behaviour with virtual information is not a new idea. Augmented reality (AR) heads up displays, where virtual information is projected on to a transparent screen, have been successfully used in the field of aviation for decades (Fischer & Haines, 1980; Fischer, Haines & Price, 1980; Ververs, & Wickens, 1998). Consumer vehicle manufacturers have also investigated using HUDs to provide AR support to drivers and such systems have been available in cars as early as 1988 (Weihrauch, Meloeny & Goesch, 1989). Recently, interest in a wearable heads up display for consumers peaked with the announcement of the Google Glass, a consumer level wearable computer system featuring a transparent display prism (Starner, 2013). A fully transparent screen allows the

head mounted wearable computer (HMD) to provide AR experiences by overlaying virtual images and information on top of the real world without occluding it.

Adding information to a user's visual space comes at a cost. Previous investigation into the impact of augmented reality systems or other heads up displays has found both cognitive and perceptual impairment caused by near-field virtual displays overlaid on a far-field environment (Sawyer, et al, 2014). A user who tries to process information from the environment and their display simultaneously makes more errors and responds slower to the information displayed by both fields (Horrey & Wickens, 2004; Liu & Wen 2004).

These cognitive costs can be reduced by decreasing the demands on mental resources created by either the virtual space or the environment (Navon & Gopher, 1979; Wickens, 1980). For example, an arrow pointing to the left or right may be processed more quickly and with a lower demand on mental resources than an instruction to "turn East on to main street" (Guthrie, Bennett, & Weber, 1991; LeFevre, & Dixon, 1986). Every task will have different cognitive costs and different opportunities for improvement often making any solution unique to the task at hand. The arrow symbol example would be inappropriate for texting or social media apps or for a display which gave numeric information. This makes general applied solutions to the cognitive costs of the addition of visual information elusive, although some general design guidelines do exist (Magers, 1983; Smith, & Mosier, 1986, Section 4).

The issues around perceptual costs are more general. Overt attention, the orientation of the eyes and head relative to the environment, can only be physically aligned to a singular region in space at a time. The human eye has greater resolution and sensitivity within the central foveal region (Anderson, Mullen, & Hess, 1991). Therefore, overt attention must be moved in series from object to object to make the most precise discriminations regarding the features of multiple visual objects. This is true for any environment and display in which complex feature discriminations must be made.

Like overt attention, covert attention, the movement of the mind's eye or spotlight of attention, generally focuses on a single volume of space (Andersen, & Kramer, 1993; Atchley, Kramer,

Andersen, & Theeuwes, 1997; Finlayson, & Grove, 2014; Parks, & Corballis, 2006; Theeuwes, Atchley, & Kramer, 1998). This volume is malleable. It can expand or retract separately on the horizontal and vertical axes, as well as in depth. Rates of detection and gradual loss of sensitivity occurs outside of this zone (Andersen, & Kramer, 1993; Eriksen & St. James, 1986; Henderson & Macquistan, 1993; Mangun & Hillyard, 1988). This sensitivity gradient is not always equal in all direction. Instead, several studies have shown the covert visual attention volume is user-centric; The sensitivity gradient is less steep when extending back to the user from the center of the volume (Andersen, & Kramer, 1993; Finlayson, & Grove, 2014; Parks, & Corballis, 2006).

Because wearable HMDs are affixed to the head it is unavoidable that the virtual display will be physically positioned between the user and the environment. A shallow user-direction visual attention gradient leads to the prediction changes on the near-position display will be detected with greater sensitivity than changes on the far-positioned environment. Consequently, attention will be pulled more easily and quickly to the device from the environment than from the device to the environment. The present study seeks to test this prediction in an applied setting with a current generation HMD and make a measure of the size of the effect of this asynchrony as it would apply to real-world situations with this device.

We used a compensatory tracking task for the far-domain environmental task. Compensatory tracking tasks require constant visual attention to detect changes in the tracking object's momentum. Responding to the task requires manual hand input. A task requiring constant visual attention for the far-domain was chosen because in most real-world settings the environment is physical and tangible and therefore maintains a constant presence. Additionally, the design of the compensatory track provides a continuous performance measure. When participants have lapses in their engagement with the far domain they will lose tracking accuracy and will fail to detect changes in the tracking object's momentum.

We used a simple word memory task for the near-domain HMD task. This task requires occasional focal attention for the participant to read the new word and in doing so creates discrete

moments in which the participant must make an inward then outward shift of attention. The word list contains verbal information and will be stored in the phonological loop (Baddeley, 1995) and therefore should have only minimal interference with the visuo-motor tracking task (which presumably places little direct demands on verbal memory). Reponses to the task are made verbally when a word is detected, and then manually after each block of trials when the participant is asked to recall as many of the words as possible. This allows both a measure of the rates of detection for near-field information changes as well as the effect of the dual-task situation on memory of the incoming information. Because the two tasks use different resources for analysis and different means for responding, their conflicts outside of perceptual interference should be minimised (Wickens, 1980; Wickens, 2002). To measure the asynchrony of attention shifts inwards and outwards, one condition will have words presentations preceded by a tone, directing participants to pre-emptively shift their attention to the device. This will allow the measurement of the attention shift inwards and outwards in the unsignalled condition and the outwards shift alone in the signalled condition. My prediction is that any increase to the tracking task reaction time caused by the inwards and outwards attention shift following an uncued presentation will be smaller than twice the increase caused by the outwards shift alone following a cued presentation.

Experiment 1

Methods

Participants

Participants were 19 (14 men; 5 women) graduate students from the University of Canterbury. All of the participants were recruited from a graduate course. They were not monetarily compensated. An unexpected hardware fault caused one of the female participants' data to be only partially recorded. The remaining 18 participants had a mean age of 23.12 years (SD = 2.91).

Stimuli and Apparatus

The experiment took place in a closed office. The tracking task was presented on a 22-in. Phillips 241B4LPYCB/75 LCD, LED backlight monitor (1,680 × 1,050 pixels). All stimuli for this task appeared within a 1024 x 768 pixel window at a viewing distance of approximately 70 cm. In the center of the window was a target comprised of five red concentric circles of radius 20, 40, 60, 80 and 100 pixels on top of a light grey background. The target was a white circle with a radius of 16 pixels with a 4 pixel black border. Figure 2.1 shows the experiment screen layout. Participants responded to the tracking task with a Saitek ST90 joystick. Conditions were started using the joystick trigger and during the tasks all input was made using the joystick. Task presentation and data recording were run on an Intel Core i5 processor with a Windows 7 enterprise platform using custom code created in the Processing language (Processing.org, 2015) with the BluetoothDesktop library (Extrapixel.ch, 2015).

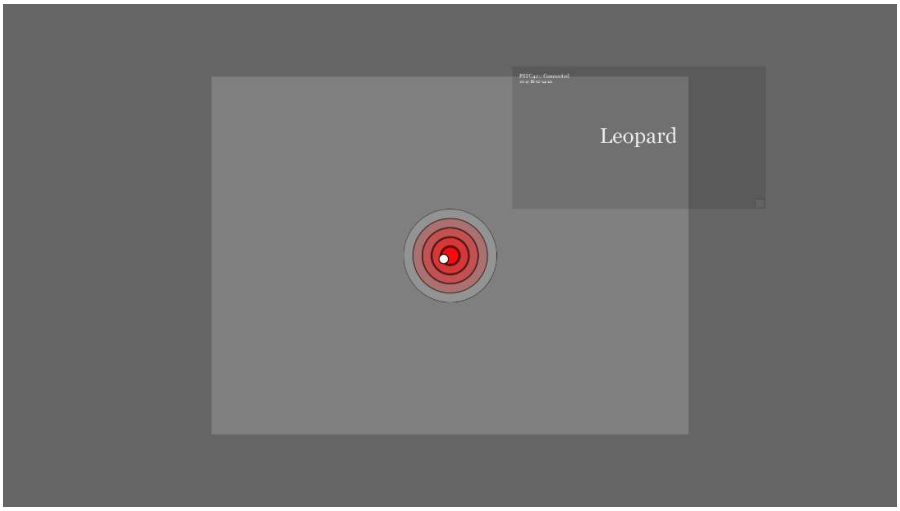


Figure 2.1: The Google Glass image layered on top of the tracking task.

The word memory task was presented on a Google Glass optical head-mounted display, with a 640 by 360 pixel resolution and 14 degree field of view. The device was resting within 5cm of the participant's eye, elevated between 25 and 40 degrees above the horizontal medial plane of the participant's visual field. The variance in elevation was due to the differences in participants' facial

structure which was not individually measured. Four lists of 20 words, one list for each condition, were created using the online Paivio et al. Word List Generator (Paivio, Yuille & Madigan 1968). Responses to this task were made verbally and scored by the experimenter. Input to the Glass, where necessary, was made using the touch pad on the side of the display. The Glass was synchronised with the desktop machine via a Bluetooth connection and the task was run using custom code created in the Processing language (Processing.org, 2015) with the native Android bluetooth library.

Procedure

Prior to the experiment, each participant came in for two practice sessions spaced a week apart. In each of these practice sessions, the participants performed the compensatory tracking task for 8 minutes. Participants quickly reached a performance asymptote and did not improve across practice sessions. Word memory is an ability practiced in normal day to day life so there were no training sessions for the secondary task.

On arrival at the office, participants were given an explanation of the experiment, its purpose and the approximate amount of time required, after which consent to their participation was gained. The participant was then given an opportunity to familiarize themselves with Google Glass, adjust to the weight and sensation of wearing Glass and adjust the position and focus of the display prism so they could clearly see displayed words and numbers on the screen. A short navigation task on Glass familiarised each participant with the audio cues associated with Glass. When participants were comfortable with the operation, the experiment proceeded to the first condition.

Each participant was expected to complete all five conditions; (1) a tracking alone condition, (2) a word memory alone condition, (3) a tracking and word memory condition, (4) a word memory preceded by audial signals condition, and (5) a tracking and word memory preceded by audial signals condition. Instructions were repeated before each task verbally and on the computer screen while the desktop and glass synchronised. Each condition ran for 4.5 minutes. An extended Latin square design was used so that each condition was preceded and followed by each other condition once in each

position. This design has 20 possible row/combinations so one was chosen at random to be removed to match the sample size of 19.

The tracking task program ran at 30 frames per second and on each frame, two separate forces were calculated and applied to the cursor. The first was the force generated by the program which pulled the cursor in a random direction at a maximum speed of 1.5 pixels per frame (45 pixels/s; 12.7mm/s; 2.5% screen width/s). Each program-generated force lasted between one and nine seconds with both a half-second fade in after a force change and half-second fade out time before the next force change. During these fade in periods the program-generated force accelerated at a uniform rate from zero pixels per second to the maximum speed and this was reversed for the fade out period. Once a program-generated force had been completed, a new program-generated force was chosen by selecting a random point from within the area of the program window and calculating the time it would take to travel to that point at the maximum speed, adjusted for acceleration.

The second motion was the force input by the participant via the joystick. To hold the cursor steady the participant would need to apply a force equal and opposite to the the program-generated force. The Saitek ST90 joystick's rectangular range of motion was mapped to a circular space to make the maximum velocity 5 pixels per frame (150 pixels/s; 42.3mm/s; 9% screen width/s) in any direction. Measurements of the velocity of both motions (in pixels per frame), the magnitude of the composite vector created by the combination of the two forces (in pixels per frame) and the displacement of the cursor from the center of the target (in pixels) was sampled and recorded on every 6^{th} frame (200 milliseconds). In conditions without the tracking task, the window showed only the background neutral grey colour with no cursor or target.

Word presentations were displayed in the center of the Glass' prism in size 80 Georgia typeface. The text was white on a transparent background. The task began after a 15 second pause in all conditions, the purpose of this pause was to allow the participant to stabilize in the tracking task before presentations began. After this pause, the next 4 minutes were split into 20 blocks, each 8 seconds long with a 4 second spacing between them. Within each of these blocks, one word was

presented at a random time drawn from a uniform distribution. Word presentations lasted 60 frames (2 seconds) and, in the conditions with audial warning signals, were preceded by a monotone beep which started 30 frames (1 second) before the presentation and lasted 15 frames (0.5 seconds).

In all conditions, each word presentation had a 50% chance of triggering a change in the program-generated force on the cursor based on a pseudorandom binary string. These force changes had a minimum duration of six seconds, but otherwise shared all the same properties as a regular force change. For consistency across conditions, the tracking task alone condition simulated these force changes using the same hidden background timer even when words could not be displayed to keep the behaviour identical across conditions. The Google Glass was worn during the tracking alone condition but no images were displayed on the screen.

After all 20-word presentation blocks had passed, another 15 second pause was given, and then the task ended. The participant was notified to the end of each condition by a message on the desktop computer screen. At this time, the participants were asked to write as many of the words as they could remember on a response form. Participants were given 4 minutes for responses to each condition, but the full time was rarely used with the majority of participants noting that they could not remember any further words. A minute rest period was given between conditions.

Results

Post-event Tracking Performance

Reaction times were measured as the amount of time between when a program-generated force change occurred and when the participant made a correction to their own force large enough to overcome the program-generated force and start to return the cursor to the center of the target. Three types of events occurred across all tracking conditions in the experiment: word presentations on Google Glass, changes of program-generated force which was being applied to the cursor, and paired word presentations/program-generated force changes. Because the paired presentations created force changes under slightly different rules than the regular force changes, word presentation

times were generated for the tracking alone condition even though the Google Glass remained blank. Each participant had between nine and eleven word presentations, twenty-four and thirty-one drag changes, and nine and eleven paired presentations. Figures 2.2a to 2.2c show the average post-event effects on participants' tracking.

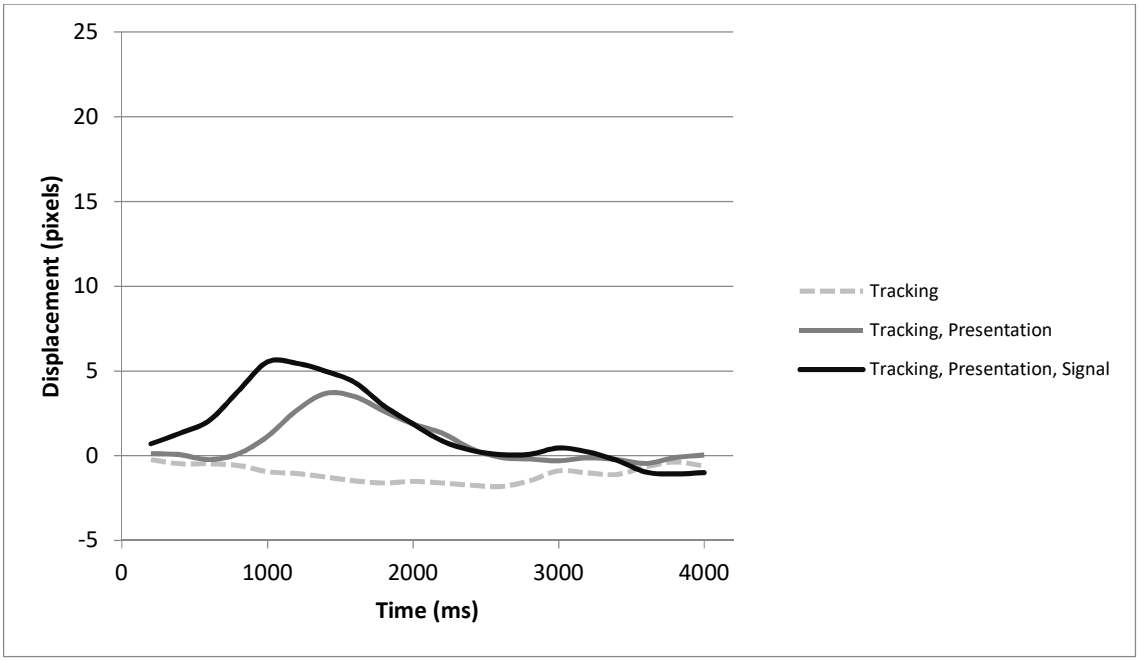


Figure 2.2a: Effect of word presentations on displacement over time averaged across participant. Displacement is measured from the pre-event value. No event occurred in the tracking alone condition.

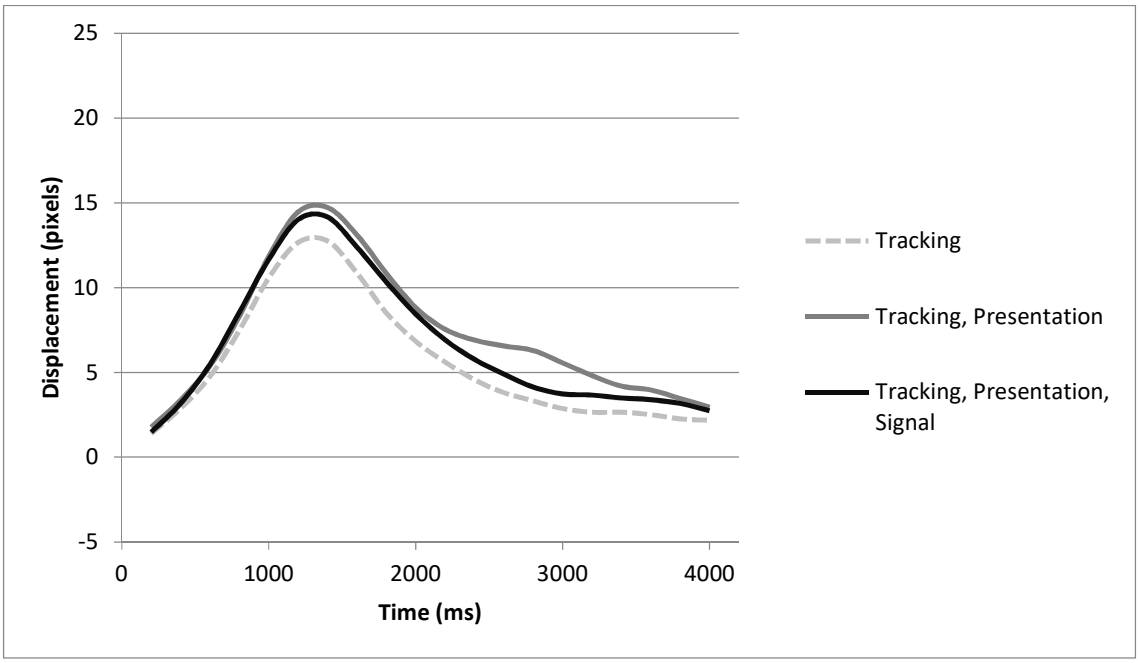


Figure 2.2b: Effect of program-generated force changes on displacement over time averaged across participant. Displacement is measured from the pre-event value.

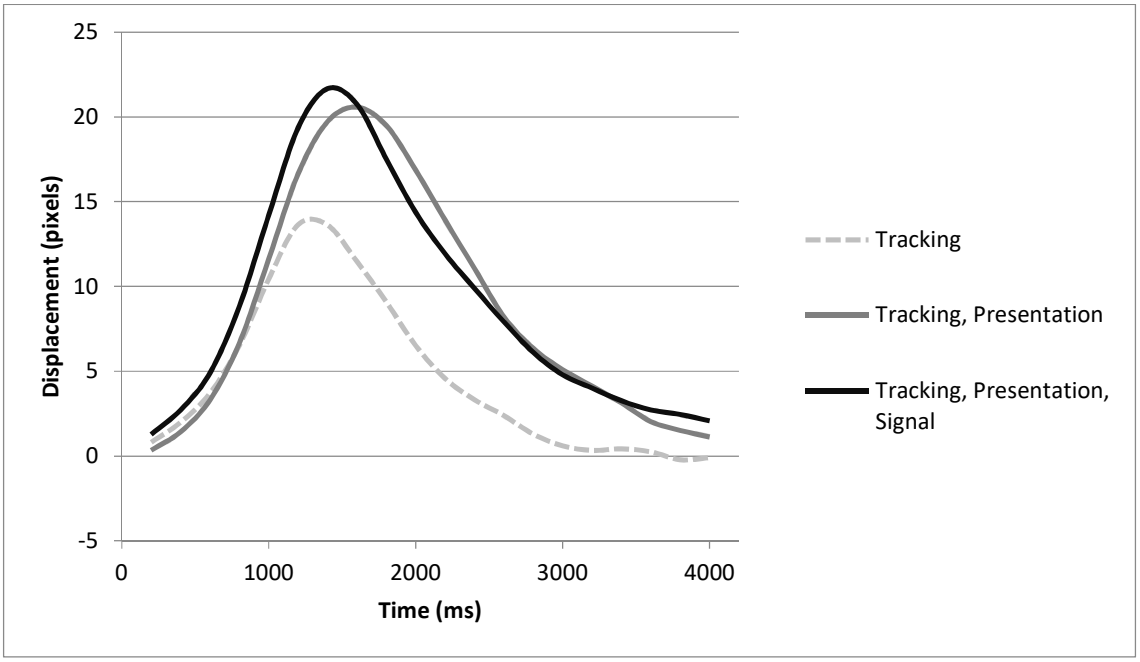


Figure 2.2c: Effect of paired word presentations and program-generated force changes on displacement over time averaged across participant. Displacement is measured from the pre-event value. A drag force change alone was made in the tracking alone condition.

Participants took on average 1315 milliseconds to respond to unpaired drag force change events. Their reaction times did not vary across condition (m_T = 1293ms, m_{TP} = 1339ms, m_{TPS} = 1308ms, $F(2,34) = 0.84$, $p = .411$, $\eta_p^2 = .047$). Average reaction times were calculated by fitting a quadratic curve to the five points around the local maximum of the first post-event peak to better estimate reactions times between the 200ms measurement intervals. In cases where there were two equal local maxima, the earlier of the two was chosen as the center. In cases where participants did not move detectably away from the center target, a reaction time of 0 was assigned.

In the dual task condition without a warning signal, participants took an average of 1586 ms to respond to paired presentations, significantly longer than for force changes alone ($t(17) = 6.16$, $p < .001$, Cohen's $d = 1.45$). The addition of the warning signal reduced this delay to 1470ms, shorter than the condition without this signal ($t(17) = 2.72$, $p = .015$, Cohen's $d = 0.64$), but still significantly longer than drag force changes not paired with a word presentation ($t(17) = 3.73$, $p = .002$, Cohen's $d = 0.88$). Yet, despite the faster reaction times, participants did not return to baseline any faster when the warning signal was present.

In the 'Tracking Alone' condition, participants took an average of 1317ms to respond to paired force changes. In contrast to the dual task conditions, this was not different to force changes alone ($t(17) = -0.75$, $p = .465$, Cohen's $d = 0.18$). The lack of an effect on reaction times when a paired presentation is assigned but the word is not shown means that the effects in the dual-task conditions are due to the word presentation itself rather than simply being an artefact of the program's presentation and force change timing.

In the dual-task conditions there was a noticeable off-center drift following word presentations (Figure 2.2a). This drift varied greatly between participants (m_{TP+TPS} = 1295ms, s_{TP+TPS} = 589ms), with five participants showing no drift at all in one of the two conditions. Because of this high variability between participants there is no difference between the signal and no signal dual task conditions either in terms of average reaction time ($t(17) = 0.04$, $p = .917$, Cohen's $d = 0.009$) or average displacement over time ($F(3.00, 102.07) = 1.368$, $p = .257$, $\eta_p^2 = .039$).

Sustained Tracking Performance

Table 2.1 shows a summary of each participants tracking performance across the full duration of each of the relevant conditions. The introduction of the word memory task was found to cause an increase in the participants' overall tracking error ($t(17) = 4.39$, $p < .001$, Cohen's $d = 1.03$). The further addition of the warning to the dual task conditions had no effect on overall tracking performance, neither significantly improving nor weakening performance ($t(17) = -0.31$, $p = .761$, Cohen's $d = 0.07$). There was no effect of task order on tracking performance across conditions ($F_{Linear}(1, 49) = 0.08$, $p = .777$, $F_{Quadratic}(1, 49) = 0.40$, $p = .530$).

Table 2.1:

Mean Squared Displacement Error for Each Participant Across Tracking Task Condition. Distances Are Measured in Pixels

Participant ID	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Tracking Alone	121	311	162	155	172	166	527	371	300	199
Tracking with Presentation	181	377	283	227	254	214	392	371	567	317
Tracking with Presentation and Signal	268	491	181	421	227	179	470	372	500	270
Average	190	393	209	268	218	187	463	371	456	262

Participant ID	p12	p13	p14	p15	p16	p17	p18	p19	Average
Tracking Alone	223	151	320	777	105	317	342	187	272
Tracking with Presentation	287	384	666	1107	186	395	369	169	375
Tracking with Presentation and Signal	399	272	638	723	131	421	859	147	387
Average	303	269	541	869	141	378	523	168	345

Word Memory

Table 2.2 shows the number of words remembered by each participant across the four conditions where the word memory task was present. An average of 1.5 fewer words were remembered in the conditions where the tracking task was present ($F_{tracking}(1, 17) = 4.92$, $p = .040$, η_p^2

= .224) but the presence or absence of the warning signal had no effect on the number of words remembered ($F_{signal}(1, 17) = 0.40$, $p = .538$, $\eta_p^2 = .023$; $F_{interaction}(1, 17) = 0.17$, $p = .688$, $\eta_p^2 = .010$). No effect of task order on the number of words remembered was found across the four conditions ($F_{Linear}(1, 67) = 0.88$, $p = .352$, $F_{Quadratic}(1, 67) = 0.09$, $p = .768$).

There were no differences in the number of words detected by the participants across conditions ($F_{tracking}(1, 17) = 1.00$, $p = .331$, $\eta_p^2 = .056$; $F_{signal}(1, 17) = 1.00$, $p = .331$, $\eta_p^2 = .056$; $F_{interaction}(1, 17) = 1.00$, $p = .331$, $\eta_p^2 = .056$), although this was because only a single participant missed any words at all. Incorrectly recalled words were relatively common, being made by 13 of the 18 participants, but their distribution was random with no differences across conditions ($F_{tracking}(1, 17) = 1.13$, $p = .302$, $\eta_p^2 = .062$; $F_{signal}(1, 17) = .239$, $p = .631$, $\eta_p^2 = .014$; $F_{interaction}(1, 17) = 1.55$, $p = .231$, $\eta_p^2 = .083$). There was no relationship between tracking performance and the number of words remembered ($F(1,16) = 1.90$, $p = .187$, $\eta_p^2 = .106$).

Table 2.2:
Number of Words Remembered by Each Participant

	No Tracking		Tracking	
ID	No Signal	Signal	No Signal	Signal
P1	16	16	12	12
P2	6	9	4	7
P3	18	20	19	17
P4	19	19	20	8
P5	12	12	10	12
P6	9	11	6	6
P7	9	8	8	9
P8	11	7	7	14
P9	16	13	9	9
P10	5	12	8	13
P12	11	12	14	8
P13	17	13	20	16
P14	11	10	7	12
P15	14	15	7	7
P16	8	7	6	6
P17	6	12	7	13
P18	19	18	17	14
P19	8	5	8	8
Average	11.8	12.1	10.3	10.6

Discussion

The results of this experiment support a difference in the speed of attention shifts depending on whether attention is being drawn towards the user or pushed away from the user. This is not, however, the only finding from these data. We sought to measure the asynchrony of attention shifts inwards and outwards in depth by exposing participants to two experimental conditions. In the first, the presentation of a new word was unexpected and therefore required at onset the participant to shift their attention to the device, absorb the new word, and then shift back to the screen to continue tracking performance. In the second, 500ms before the presentation of a new word participants were given an auditory signal so that they could center their attention on the device before presentation of the new word leaving them only to encode the new word and make the return shift of attention. The length of the inward shift can then be calculated by taking the difference in the delays caused by paired word presentations for the cued and uncued conditions.

For this calculation to be valid, participants must have actually made pre-emptive shifts of attention to the device following the signal and also must have been unable to detect tracking task events while attending to the device. If participants were able to process word presentations on the device and changes in the tracking task simultaneously it would be expected that paired presentations would be detected and responded to no slower than unpaired presentations. This was not the case. Further evidence of the participants' serial responding can be seen in Figure 2.2a; while the participants' attention was moving away from the device, their tracking began to veer off course even when a momentum change was absent. On returning their attention to the screen, participants then took a short time to determine a corrective course before implementing it. Evidence that the signal drew attention to the device earlier can also be drawn from this figure. The off-target drift begins approximately 500ms earlier for the signalled vs unsignalled conditions. This is consistent with signal condition drawing attention to the device when the signal tone ended.

In the unsignalled dual-task condition, a shift of attention inwards to the device and then back out to the screens' depth added 247ms on average to the participants' tracking response time. In the signalled dual-task condition, the return movement alone added 162ms on average. From this an average inward attention shift time of 85ms (SE= 48ms) can be calculated. The inward shift of attention takes only approximately 34% of time taken for the full attention movement required to absorb a presented word. This supports the existence of different perceptual costs when users need to move their attention to, or away from HMD devices. Additionally, in accordance with the predictions of a user-centric attentional volume, new stimuli on the near-field device captured attention very easily. Only a single participant failed to detect any of the word presentations.

An alternate explanation for the results is that both inward and outward attention shifts take the same amount of time and the extra delay is instead caused by the participant fixating on the device while they process the newly presented word and add it to their phonological loop. The experiment was designed to minimise the overlap in non-perceptual, mental resources required but there was still evidence of cognitive interference between the two tasks. In the dual-task conditions participants performed more poorly in both of the tasks. Fewer words were recalled by participants in the word memory task and the participants were less accurate with their tracking.

Experiment 2

To clarify the source of the increased delay, we conducted a second experiment similar to the first but with two new conditions. In one condition, the word was displayed on the same screen as the tracking task. Human actors are able to perform multiple discrimination tasks when the features to be discriminated share a surface (Ernst, Palmer, & Boynton, 2012) and by removing the displacement in depth between the word memory task and the tracking task there should be no requirement for participants to make an attention shift in this condition. Without an attention shift, any present delay caused by cognitive cost of processing and adding a new word to the phonological loop should be isolated. To ensure that the participant is not making a shift of visual attention, the second condition

will present the words aurally. If the delay in responding to the tracking task is solely caused by the attention shifts required to attend to information displayed on the Glass then there will be no difference in response times for tracking task events when they are paired or unpaired with a word presentation.

Methods

Participants

Participants were 5 men and 7 women recruited on the University of Canterbury campus and via a local participant recruitment website. They were compensated $10NZ for their participation. One participant refused to complete all five trials; they found wearing the glasses uncomfortable. A second participant did not respond consistently enough to direction changes in the tracking task to give an accurate representation of their response time. The remaining 10 participants (5 men, 5 women) had a mean age of 24.8 years (SD = 1.69).

Procedure

The procedure was identical to experiment 1 except the practice sessions were discontinued due to the lack of improvement and replaced by a 2-minute tracking practice session preceding the experiment. Two conditions were shared with experiment 1; (1) tracking alone, and (2) tracking with the word memory condition presented on the Glass. The remaining three conditions were; (3) the tracking task with the word memory condition presented on the same screen, (4) the tracking task with the word memory condition presented aurally, and (5) the word memory task alone presented on the computer screen. The experiment took 52 minutes to complete in total.

Results

Table 2.3 shows a summary of each participant's performance across the 5 conditions of experiment 2. The introduction of the word memory task caused an increase in the participants' overall tracking error in only the Glass display condition ($F_{Glass}(1,9) = 20.50$, $p = .001$, $\eta_p^2 = 0.67$).

Table 2.3:

Averaged Memory and Tracking Results for Experiment 2. Condition 1: Tracking; Condition 2: Word Memory on Screen; Condition 3: Tracking and Word Memory on Glass; Condition 4: Tracking and Aural Word Memory; Condition 5: Tracking and Word Memory on Screen

	Words Detected **M (SD)**	Words Remembered **M (SD)**	Overall Tracking Error (mm) **M (SD)**	Unpaired Event Response Time (ms) **M (SD)**	Paired Event Response Time (ms) **M (SD)**
1.Tracking			5.29 (1.72)	1196 (344)	
2.Memory	20.0 (0.0)	12.7 (4.5)			
3.Glass	19.9 (0.3)	10.8 (4.6)	6.35 (1.77)	1252 (267)	1596 (335)
4.Aural	19.8 (0.4)	11.1 (4.6)	5.80 (1.21)	1210 (219)	1454 (403)
5.Screen	20.0 (0.0)	12.3 (4.2)	5.66 (1.41)	1278 (248)	1305 (290)

The number of words remembered did not vary between conditions ($F(3,27) = 0.86$, $p = .481$, $\eta_p^2 = 0.21$). This is different to the finding of the first experiment, but is likely only reflective of the smaller sample size as when memory is compared across experiments there is no differences in the number of words remembered ($F_{Experiment}(1,26) = 0.15$, $p = .698$, $\eta_p^2 = 0.07$; $F_{Interaction}(1,26) = 0.06$, $p = .802$, $\eta_p^2 = 0.06$) while the difference between memory alone and dual task conditions remains ($F_{Dual\ Task}(1,26) = 4.63$, $p = .041$, $\eta_p^2 = 0.55$). Again, there was no effect of task order on the number of words remembered ($F_{Linear}(1, 39) = 0.35$, $p = .560$, $F_{Quadratic}(1, 39) = 0.83$, $p = .368$).

Reaction times to unpaired force changes did not differ between experiment 2 conditions ($F(3,27) = 0.67$, $p = .577$, $\eta_p^2 = 0.17$). Concurrent word memory tasks on the Glass increased reaction times to force changes regardless of whether the information was given visually ($F_{Glass}(1,9) = 35.59$, $p < .001$, $\eta_p^2 = 1.00$), or aurally ($F_{Audio}(1,9) = 6.70$, $p = .029$, $\eta_p^2 = 0.64$). In contrast, there was no evidence of an increase in reaction times when the word memory task was presented visually on the same visual plane as the tracking task ($F_{Screen}(1,9) = 0.23$, $p = .645$, $\eta_p^2 = 0.07$). This finding supports the hypothesis

that the increase in reaction times to far domain events is due to the need to shift attention between depth planes rather than interference from central executive processes.

Discussion

This experiment sought to confirm that the delay in responding to paired events the compensatory tracking and word memory tasks was due to competition for focal attention rather than cognitive processing resources. To test this, the word memory task was presented on the Glass and also through two methods which were expected to remove the need for a shift of focal visual attention. One method was to have the presentation of the words and the tracking task share the same perceptual surface. Previous research into dividing attention between superimposed surfaces show that participants have the ability to process multiple features of the same surface simultaneously even when detecting the same number and type of features across separate surfaces imposes a performance cost (Ernst, Palmer, & Boynton, 2012). A similar pattern was found here; when the need to shift focal attention in depth was removed, participants were no slower to respond to momentum changes when the tracking task was paired with a word than when the tracking task was performed alone. The other dual tasking costs detected in experiment 1, an overall decrease in tracking accuracy and a reduced ability to recall words for the list, were also absent in this condition. This could be because the ability to process both tasks in parallel removed the need for the participant to make higher-level performance evaluation decisions regarding the division of attention, reducing the overall cognitive load. This result eliminates the alternative explanation of the first experiment's results that both inward and outward attention shifts take the same amount of time and the extra delay is instead caused by the participant fixating on the device.

The second method of removing the need for a perceptual attention shift in depth was to deliver the word memory task aurally. In contradiction to the hypothesis of this experiment, the removal of the need for an attention shift did not remove the delay in responding for participants during this condition. The reason for this delay is unclear as under the Wickens Multiple Resource

Model, this pair of tasks should have had the least overlap in required mental resources (Wickens, 1980; Wickens, 2002). One explanation could be that some participants were compelled to visually search for the speaker on hearing a word read to them. This shift in orientation may be unconscious or bottom-up as there are strong cross-modal links in the orientation of spatial attention (Driver & Spence, 1998). This would explain the relatively large variability in paired reaction times for the aural presentation trials compared to the other dual task conditions, as the auditory presentation may disorient visual attention.

Combining the two experiments confirms that switching visual attention to and from HMD devices imposes a perceptual cost that is separate from the cognitive demands of the tasks. It also confirms the primary hypothesis of this study, that it takes less time to shift attention away from tracking to the Glass than to return it from the Glass to tracking. This poses a problem for many situations in which a developer may want to improve real-world task performance with the new HMD technology. Threats to a user's physical safety exist primarily within the real-world environment but users' attention is drawn more easily to the near-field virtual display. This may make apps which deliver warnings to a user about their environment counterproductive, as a highly salient warning signal may draw the user's attention inwards in depth and the delay in returning it to the environment may slow rather than improve the user's response to the hazard. Future research into the best methods of directing a user's attention already centred on the device back to the environment would be of value to developers of apps which include components that monitor the users' environment or equipment.

Chapter 3: Visual Cues to Reorient Attention from Head Mounted Displays

Preamble

The previous experiment finds evidence for a response time cost when the users need to shift their attention between the environment and HMD. This shift of attention takes longer to perform when the user is shifting their attention outwards, from the device to the environment. A method to help the user make this outwards attention shift more easily or more quickly would be a valuable tool for designers of HMD software. Visual attention orienting cues have been investigated in many contexts before, and these cues are usually divided into two classes: endogenous cues and exogenous cues (Jonides, 1981; Müller & Rabbitt, 1989; Posner, 1980). Exogenous cues are the faster of the two classes and therefore seemed a logical choice for driving the outward attention shifts. Exogenous cues, however, require the designer to have the ability to create changes in the environment in order to direct attention. This is not feasible for software designers who only have control over the user's device.

The purpose of this next experiment was to search for a HMD viable replacement for the typical form of exogenous cues: the abrupt onset style cue. This led to what was to be the central theory of the PhD thesis, that a visual cue could use the reflexive response to pursuit motion to generate a reaction with similar speed and reliability to using the reflexive orienting response to abrupt environmental changes. Posner (1980), Jonides (1981), and Müller and Rabbitt (1989) all attribute the response speed difference between the exogenous and endogenous classes to stemming from a difference between reflexive and voluntary control. According to their theory, reflexive control of attention happens immediately, while voluntary control requires the user to interpret the cue and then make a decision about whether to follow the cue. This additional two stage process is what makes endogenous cues, as a class, slower than ones which tap into a reflex. Orienting to, and matching speed with, an abruptly moving object within the user's field of view is a reflexive response. Therefore, we expect HMD-delivered visual cues utilising this pursuit-tracking reflex to fall into the exogenous

category and exhibit similar properties to the other forms of exogenous cues. Because pursuit motion tracking can begin anywhere in the user's field of view, it can start within the region of the HMDs screen and thus is an appropriate cue type for designing with the device in mind.

Attribution

This chapter is a reproduction of Ward, M., Barde, A., Russell, P. N., Billinghurst, M., & Helton, W. S. (2016, September). Visual cues to reorient attention from head mounted displays. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*(Vol. 60, No. 1, pp. 1574-1578). Sage CA: Los Angeles, CA: SAGE Publications.

Introduction

Improvements in Head Mounted Display (HMD) technology open exciting design opportunities for improving human productivity. At the same time, the widespread employment of HMDs in real-world settings requires users to divide their attention across three-dimensional space. HMDs require the user to split their visual attention between the display and the surrounding world. Users cannot process all of this visual information and instead must allocate their attention to particular regions in space (Arnott & Shedden, 2000; Atchley, Kramer, Andersen & Theeuwes, 1997; Jonides, 1981; Posner, 1980).

To ensure the detection of an event or object and enable rapid user responses, a user's attention can be assisted with a cue. Cues embedded in the HMD could be used to reorient the user to events in the surrounding world. Common visual cues include centrally positioned arrow cues which point towards the desired location for the user's attention and peripheral onset cue where a new object appears in the user's peripheral vision or an existing object changes in luminance or hue on the display (Theeuwes, Atchley, & Kramer, 1998). Posner (1980) found these two types of visual cues elicited different reactions, and proposed two classes of cues. Voluntary cues, such as the arrow cues, require processing before they are followed while reflexive cues tap in to an orientation reflex and are

followed more quickly (Jonides, 1981). Orientation in the direction of voluntary cues can be suppressed by a user if such a behaviour is desired (Jonides, 1981) and reach peak accuracy 150ms later than orientation to reflexive cues (Müller & Findlay, 1988; Müller & Rabbitt, 1989).

Current commercial HMDs such as the Recon Jet or Google Glass, create constraints for the kinds of cues used to redirect attention back to the surrounding world. The display space is relatively small. Also, when real-world objects or events are the target the user may need to be able to shift attention to anywhere in a full 360° field of rotation. This is in contrast to standard cueing experiments conducted with video terminals where targets appear all within one screen or stereoscopic image and rarely outside the participants' peripheral vision (or off the display itself).

The narrow digital display visual angle range and the wide possible target appearance visual angle range mean that peripheral onset cues are unusable with many currently marketed HMDs. For certain types of targets, however, the inability to suppress the reorientation of attention and faster reaction times are important. Warnings about real-world hazards in particular benefit from these properties.

One alternate automatic visual reflex that originates from a central visual location is the pursuit tracking reflex. When an object begins smooth motion within a user's field of vision, the user detects the object's momentum within 150ms before making a catch-up saccade to the object and beginning precise tracking (Lisberger, 2010; Rashbass, 1961; Wilmer & Nakayama, 2007).

In this experiment, we will compare pursuit motion visual cues with centrally positioned arrow cues on a Recon Jet HMD for reorienting the user to an event in the surrounding world. Using the cues participants will be required to find targets inside (±50°) and outside (±100°) their peripheral vision range and mark them with a virtual paintball gun. We predict that participants will begin head motion towards the targets and mark the targets more quickly when directed by pursuit motion cues than central arrow cues and a no visual cue control condition. To best represent the range of design possibilities, three conditions of visual cues; no cue, arrow cue, pursuit motion cue; were paired with analogous auditory cues; no auditory cue, a static non-spatial audio cue, and a spatialized audio cue

which moved from the user towards targets as they appeared. In this paper, we focused on the visual cues; see Barde, et al. (2016) for an analysis of the audio cues.

Method

Participants

Thirty participants (20 male) were recruited on campus and using social media. Participants were all students between the ages of 18 and 34 years (M = 23.9; SD = 4.2). All participants were required to be able to see clearly at distances up to 3 meters without corrective lenses as glasses could not be worn beneath the HMD. Informed consent was collected from all participants prior to their participation and the study was approved by the University of Canterbury's Human Ethics Committee.

Apparatus

Tracking data were collected using the Dtrack software (Ar-tracking.com, 2015) and a four ARTRACK2 camera array. Tracking data were transferred using the VRPN software (GitHub, 2015). Positional trackers were 3D printed on site and marked with reflective material. Central computation for the experiment was run using custom code written in the Unity 3D engine (Unity3d.com, 2015). Far domain stimuli were presented on three 2440mm (74.65° visual angle) by 1830mm (59.53°) screens connected at 60 degree angles mounted at 600mm above the floor. Images were projected on to the screens by three NEC LT265 projectors. Near domain stimuli were presented on a Recon Jet (Recon Instruments, 2015) connected wirelessly. Responses were made using a modified Steradian S-7X laser gun. The connections between these devices are shown in Figure 3.1.

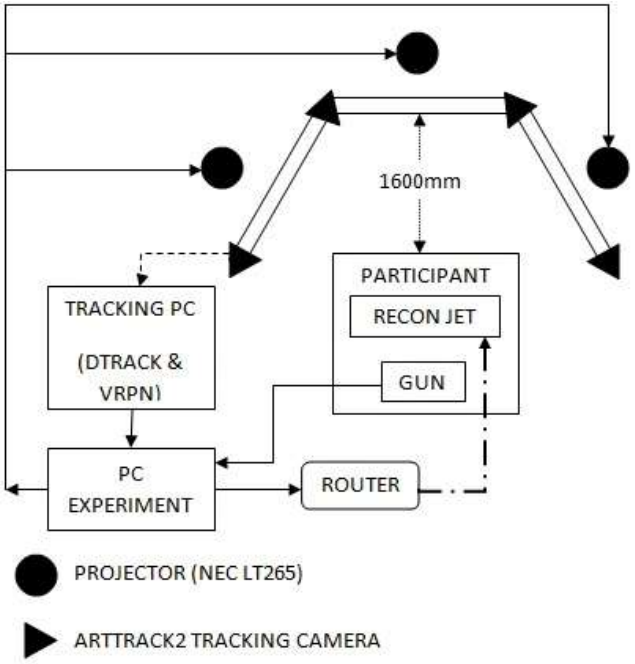


Figure 3.1: Block diagram for the layout of experiment components.

Stimuli

Onscreen tracking points were 8mm radius disks coloured yellow ($FFED00) for head position and blue ($0096F3) for gun position. Blue ($0096F3) paint splotches created by pulling the trigger were scaled to a radius of 120mm and began to fade linearly after 0.5 seconds over an additional 0.5 seconds. Targets were 58mm radius disks that changed from yellow to blue after being hit by a shot within a radius of 116mm from the central point.

Text on the center screen was in size 40 black Arial typeface (65mm, 2.3°) positioned in the horizontal center of the screen, 722mm (25.4°) below the vertical mid-mark. Incoming text on the Recon Jet was size 22 white ($FFFFFF) Arial type face (0.5°) positioned in the center of the horizontal center of the screen with previous message listed above in grey ($646464). Visual interrupt signals included white arrows 1.3° in width and 6.5° in length, angled at 40° for targets at a 50° horizontal offset and angled at 80° for targets at 100° horizontal offset. Pursuit visual stimuli caused all objects

on the Jet screen; current text, previous text and a white bounding box, to move in the direction of the target at a speed of 16.2° per second.

Procedure

Participants were seated on a rotating chair 1600mm back on the normal line extending from the center of the middle screen such that targets could be displayed anywhere on a 200° horizontal arc. Participants were allowed to adjust the seat height for their comfort. Participants were then given the Recon Jet with a 3D position tracker and then given help to properly adjust the devices. A short calibration process ensured both trackers and the gun trigger were functioning properly, that the participant could read from both the screen and the Jet, and that the participant had a full 200° range of motion.

Next participants were given six practice trials to show some the different types of audio and visual cue combinations that they could expect during the experiment including two of each of the visual cues and two of each of the audio cues. The remainder of the experiment was separated into 3 blocks separated by two 5 minutes breaks.

During each block participants would alternate between reading aloud strings of eight digits from the center screen and messages on the Recon Jet following one of three structure chosen at random; [Alpha] team entered site [C][37] at time [1407], [Alpha] cleared floor [2] at time [1407], or [Alpha] team exited site [C][37] at time [1407]. During one third of the periods when the message was shown on the Recon Jet, a cue would interrupt the participant one second after the message's onset and a target would appear at one of the four possible locations: -100°, -50°, +50° and +100° horizontal rotation. Participants were then required to mark the targets with their gun as quickly as possible, turning the target from yellow to blue.

Alternation between central screen number strings and recon Jet messages resumed once the target was marked and the participant had returned their head to facing the middle 28° arc of the central screen. Within each block there was one target event trial for each combination of visual and

audio cues for each position for a total of 36 target events and 72 non-event messages. The experiment took on average 65 minutes to complete and the participants were compensated $20NZ for their time in shopping vouchers.

Results

One male participant was excluded due to technical issues with the audio signals and a second male participant was excluded for failure to comply with directions. Due to additional technical difficulties (mainly power management issues with the Recon Jet), six of the remaining participants lost some of their final block event data (Mean = 4.33 events). These later participants, however, where not excluded. Instead to compensate for some lost events, the average of the participants' second and third blocks results were used for all analyses. There were no differences across the second and third blocks for the two measures of interest: milliseconds until head began motion, $F(1,21)= 0.01$, $p= .927$, and milliseconds until the gun tracker reached a target, $F(1,18)= 1.98$, $p= .176$.

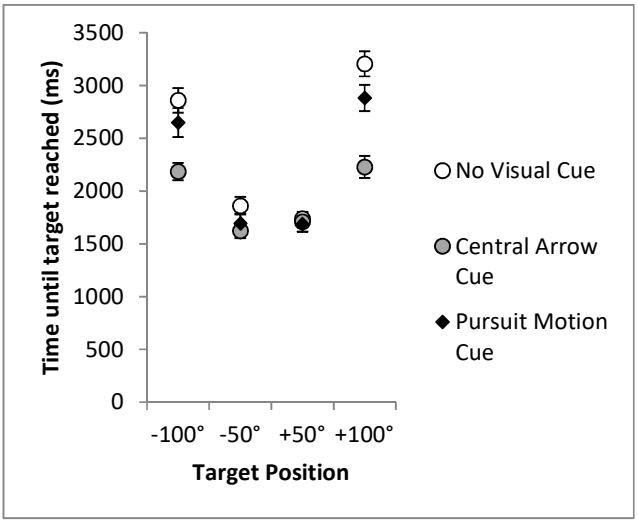


Figure 3.2: Average time until the participant's gun was pointed at the target for each visual cue type at each position. Error bars represent 1 standard error.

First, we examined time taken by participants to aim their gun on the target using a 2x2x3 (Near/Far: Left/Right: Visual Cue) repeated measure analysis of variance (ANOVA). There was a significant difference in the time to reach targets at 100° rotation compared to 50° rotation, $F(1,27)= 278.94$, $p< .001$, $\eta^2_p= .912$ (Figure 3.2). Participants reached left hand targets slightly faster than right hand targets, $F(1,27)= 2.66$, $p= .114$, $\eta^2_p= .090$, and this difference increased for far positioned targets, $F(1,27)= 6.06$, $p= .020$, $\eta^2_p= .183$. This pattern of results is likely caused by all of the participants holding the rear handle with their right hands.

Because of the strong interaction between target distance and visual cue type, $F(2,54)= 18.13$, $p< .001$, $\eta^2_p= .402$, we performed separate 2x3 (Left/Right: Visual Cue) repeated measure ANOVAs for near and far targets to determine the effects of the different types of visual cues at each distance. For near cues, it took participants longer to reach the target following no-visual-cue events, $F(1,27)= 10.19$, $p= .004$, $\eta^2_p= .274$, than other cue types. In particular -50° no-visual-cue events were completed more slowly on the left than on the right, $F(1,27)= 6.29$, $p< .018$, $\eta^2_p= .189$. There were no other significant differences between the two cued event types (arrow and pursuit motion).

For the far targets, participants reached the targets faster with a pursuit motion cue than no visual cue, $F(1,27)= 4.43$, $p=.045$, $\eta^2_p= .141$. However, participants reached the targets significantly faster with the central arrow visual cue than pursuit motion, $F(1,27)= 30.59$, $p<.001$, $\eta^2_p= .531$. These differences in speed were not affected by the side on which the target was placed, $F(2,54)= 2.86$, $p=.066$, $\eta^2_p= .096$.

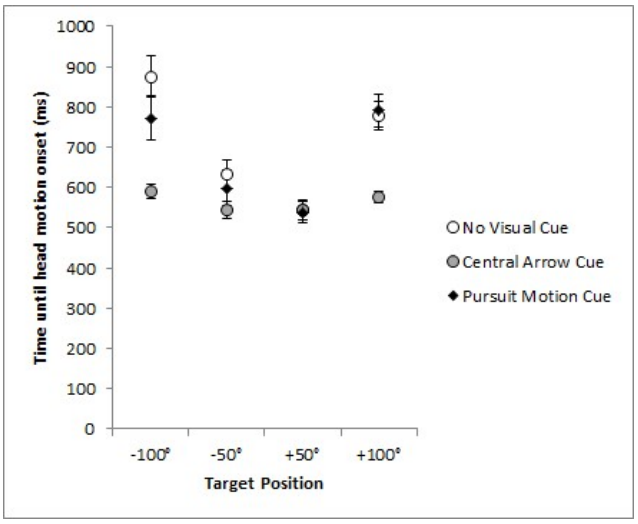


Figure 3.3: Average time until the participant began head motion for each visual cue. Error bars represent 1 standard error.

Turning now to head motion onset latencies we performed second a 2x2x3 (Near/Far: Left/Right: Visual Cue) repeated measures. Participants took longer to begin head motion for far position targets than for near targets, $F(1,27)= 49.68$, $p<.001$, $\eta^2_p= .648$. This increase in reaction times was smaller for central arrow cues than other cue types, $F(2,54)= 15.18$, $p<.001$, $\eta^2_p= .360$ (Figure 3.3).

Long reactions times for the onset of head movement may indicate participants failed to notice a cue or failed to determine the correct direction to rotate from the cue. To test for these cases, we analysed the number of times participants made an initial head movement in the incorrect direction. The total number of times participants missed or failed to process a cue will be approximately double this number as, on average, participants will guess the right direction to move 50% of the time.

For no visual cue events participants were forced to choose their initial head direction at random for targets outside their peripheral vision. 24 out of 28 participants made direction errors for these events, a rate not significantly different from the expected rate if the participants made direction choices for far-position un-cued events randomly, $\chi^2(1) = 3.09$, $p= .079$). Overall, the

participants made 51 (45.5%) direction errors to un-cued far targets and 8 (7.1%) errors to un-cued near targets.

Participants made no direction errors for arrow-cued events, but 11 out of 28 did so for pursuit motion cues. This rate is statistically unlikely if participants were making direction decisions randomly, $\chi^2(1)= 52.15$, $p< .001$, and is significantly smaller than the rate of participants who made incorrect direction decisions for far-position un-cued events, $\chi^2(1)= 4.46$, $p= .020$. For participants who had difficulty detecting or interpreting direction from the pursuit motion cue, there was no difference in the average number of direction errors made between un-cued ($M= 2.13$) and pursuit motion cued ($M= 2.27$) far-position events ($t(33)= -0.51$, $p= .613$), indicating that the pursuit motion cues were completely undetected by this subgroup of participants.

Discussion

Contrary to the primary hypothesis of this study, pursuit motion cues did not cause participants to react more quickly or reach targets faster than centrally-positioned arrow cues. This was due in some part to a subgroup of the participants failing to detect the pursuit motion cues. The pursuit motion cues in this experiment were transient, lasting 133ms (8 frames at 60 fps), whereas the central arrow cues persisted until the target was marked. It is possible that arrow cues might have caused similar numbers of direction errors if they had also been limited to 133ms screen time. However, for future design of software for augmented reality devices, purposely limiting the efficacy of a cue is not a practical or a desirable goal. Instead, it is more likely to beneficial to increase the onscreen time on any pursuit motion cues, adding repetition if an increase in screen time would too significantly slow the motion of the visual cue object.

While the pursuit motion cue may be able to be improved in this way, there is still no evidence that they would perform in a way superior to the central arrow cues. When the subgroup that experienced direction confusion in response to the pursuit cues was removed, a 150ms difference in reaction time still persisted between the two visual cue types. This means that the pursuit motion

cues of the current experiment showed neither of the expected properties of reflexive cues (Posner 1980; Jonides, 1981); some participants showed the ability to suppress the pursuit motion reflex by prioritizing the completion of the reading task and missing the cue while the remainder showed no benefit in terms of reaction or response times.

If pursuit motion cues do not show the same properties as abrupt onset cues, it may be because they both belong to the reflexive visual cue category but the properties of that category may be misattributed. Jonides and Yantis (1988) and Theeuwes (1990) found evidence that the abrupt onset of stimuli captured attention while properties such as brightness and hue did not when their onset was gradual. Therefore, the properties that Jonides (1981) proposed as the defining traits of reflexive visual cues; capacity, resistance to suppression, expectancy and increased costs/benefits, may instead be properties unique to the abrupt onset reflex rather than reflexive visual cues in general. Additional support for this possibility is provided by Hillstrom and Yantis (1994) who found that motion captures attention only when it creates a new visual object by differentiating an object from the background: a peripheral onset cue (see also Abrams & Christ, 2003).

It is possible that some property of the pursuit motion cues as used in this experiment impaired their value as a visual cue. Lovejoy, Fowler & Krauzlis (2009) investigated the distribution of attention during pursuit motion and found that a participant's attention is focused on the center of a moving object with less focus given to other components of the object the greater their eccentricity. In this experiment, the pursuit motion cue lasted 133ms, longer than is needed for the participant to begin eye motion (Wilmer & Nakayama, 2007) and be an effective visual cue (Ansorge, Carbone, Becker & Turatto, 2010). However, as the pursuit target filled a large portion of the Recon Jet's screen, the center of the target, and therefore the center of the participant's attention, left the screen within 83ms which may be too short a duration for the participant to have fully processed the pursuit target's momentum.

Alternately, it is possible that pursuit motion does not automatically translate from eye motion to head motion when the target does not reach the edge of the participant's peripheral vision

as it is unable to do when limited to the screen of a current head mounted wearable. In this case, a participant might have to use some kind of mental process to convert their reflexive eye motion into a voluntary head motion. This would cause the theoretically reflexive pursuit motion cue to act as a voluntary cue; a voluntary cue that is less trained and therefore followed more slowly than the arrow cues. Future research might investigate this possibility by combining head tracking with eye tracking although doing so without occluding the participant's view may be a challenge for current technology.

Pursuit motion cues were only successful for 17 out of 28 (61%) participants or fewer. It may be possible to fix the issue of direction errors by extending the duration of the pursuit motion cue however even in this case there is no evidence that these cues would then out-perform central arrow cues. Arrow cues were responded to only 40ms slower than peripheral onset targets, the traditional reflexive visual cue, and therefore may represent the best option for visual cues for most HMDs given current technological constraints.

Chapter 4: Assisting Visual Search with Head Mounted Displays

Preamble

The core theory of the PhD thesis was to be that a visual cue could use the reflexive response to pursuit motion to generate a reaction with similar speed and reliability to using the reflexive orienting response to abrupt environmental changes. The previous experiment did not support this theory. Typical practice for the failure of a new cue type would be to attempt to refine or improve upon the design of the cue, and indeed there were design elements of the pursuit motion cue which could have been improved upon. The previous experiment yielded an additional, unexpected finding however. When removing the effect of distance to the target but looking at when the major acceleration in the direction of the target began, endogenous class cues were significantly slower than exogenous cues but the magnitude of this difference was small (~40ms).

HMD delivered cues require participants to shift their attention in depth and this process takes time. If users are able to process the information and make the judgements required by endogenous/voluntary visual cues during the delay created by the attention shift in parallel, then it is possible that typical reaction speed difference between endogenous and exogenous cues are reduced when directing attention from a HMD device. This means that the search for a replacement exogenous class cue for HMD software design may be unnecessary. Even a small improvement on the endogenous cue provided in the previous experiment would have closed the gap between the two classes. To see if and how endogenous cues could be improved to direct a user's attention from the HMD to the environment, we developed the next experiment to test different types of endogenous cues in a search task more complicated than the previous one.

Introduction

Augmented reality, the layering of virtual objects and information over top of real-world space, is a valuable design tool for a variety of tasks. In professional settings, augmented reality is used

for the treatment of many types of phobias (Botella, et. al., 2010; Garcia-Palacios, et al., 2002) and to enable doctors and other highly skilled professionals to share their expertise remotely (Borgmann et al., 2016; Chai, Babu, & Boyer, 2015). In more common every-day settings, augmented reality can be used to give users feedback while exercising (O'Hear, 2017), help users navigate new environments by displaying maps and trails (Strange, 2017), provide information about locations and objects of interest (NianticLabs, 2015), and make new types of interactive mobile games such as Pokémon Go (Niantic, Inc., 2016).

Wearable computers, particularly ones with transparent screens, are uniquely positioned for the use with augmented reality programs. Head mounted wearable devices (herein HMDs) specifically enable users to attend to visual virtual information without restricting use of their hands or arms. Devices such as the Google Glass (herein "Glass"; Google, 2017) and the AltoTech Cool Glass One (Coolglass.com, 2016) are the first commercial attempts at filling this emerging niche in the ubiquitous computing frontier.

For augmented reality programs to function, a user must retrieve information from both their device and the real-world but humans cannot process all the visual information they receive simultaneously. Instead they must shift their focal attention between particular regions in space (Arnott & Shedden, 2000; Atchley, Kramer, Andersen & Theeuwes, 1997; Jonides, 1981; Posner, 1980). This is further compounded by the design of current HMDs which feature only a monocular display, restricting visual virtual information to a single fixed depth plane. Shifting attention between depth planes has been shown to cause delays in target detection when searching a 3D environment (Theeuwes, Atchley & Kramer, 1998; Finlayson & Grove, 2014). This means shifting attention between the fixed depth plane on which virtual information is displayed and the variable depth planes on which relevant information may come from in the environment will create further performance impairments above and beyond the cognitive costs of merely increasing the information load.

To maximise the utility of HMDs for the purposes of augmenting user performance on tasks, it is important to minimise costs and maximise the value of the information given. Attending to the

device moves focal attention away from the real-world task. Furthermore, from the previous research reported in Chapter 2, it takes an estimated 160ms to move their attention from the device's virtual space to the physical space. During the time taken to shift both the eyes and the internal locus of attention neither of the spaces are properly attended. Therefore, to minimise the costs to the user's attention it is important to decrease the amount of time needed looking at the device to process information while also reducing the number of attention shifts a user is required to make between physical and virtual surfaces. Maximising the value of given information can be achieved by increasing information quality or information quantity.

The four design outcomes; time taken to process the virtual information, number of attention shifts required, information quality, and information quantity, are all important to optimally assist the user with virtual information. These outcomes are not necessarily equally important, however, and unavoidably trade-offs must be made amongst them. Increasing the quantity of information displayed will also increase the amount of time needed to absorb the information and consequently the amount of time with focal attention removed from the real-world surfaces of the environment. A larger amount of information also means the most important information may be missed among temporarily irrelevant information decreasing average information quality and requiring attention to be shifted back to the device more times to retrieve missed value.

One fundamental trade-off is between the use of static and dynamic virtual information. Dynamic information is updated in real time, improving the quality of information by remaining current throughout the user's performance. This is particularly beneficial when the user needs to correct an initially incorrect response. Dynamic information comes at the cost of requiring multiple attention switches to and from the device, and risks causing additional unneeded attention movements when participants automatically attend to changes in virtual objects within their peripheral vision. In contrast, static information does not change during the task. This means that all the information displayed can be retrieved in a single initial attention shift provided the information

quantity is sufficiently small enough to be stored in memory. Static information comes at the costs of the increased information quality inherent to up-to-date information.

In this study, we explore design outcome trade-offs within the context of a real-world task archetype: the visual search. Visual search tasks are common in daily life. Looking for the right shirt from the closet, finding the milk in the fridge, hunting down where on the desk keys had been left; These are just a few of the searches one might make before even leaving the house. In an experimental visual search task, a participant will typically be asked to find a particular target within a field of distractors and to report "yes" or "no" if the target was present. Search time or accuracy (in a limited search time window) is recorded. (Theeuwes, 1994; Wang, Cavanagh, & Green, 1994; Wolfe, 1994).

Response time to visual searches depends on the properties of the target relative to the distractors. When the target is highly salient and differs in one or more basic features such as colour, size, angle or shape the target is found quickly and the speed of the search is close to independent from the number of distractors (Carrasco, & Yeshurun, 1998; Neisser, 1963; Treisman, 1985; Wolfe, 1994). Alternatively, when distractors share properties with the target or the difference in features between the target and the distractors are closer to the detection threshold, search speeds decrease and begin to take on an approximately increasing linear relationship with the number of present distractors (Wang, Cavanagh, & Green, 1994; Wolfe, Cave, & Franzel, 1989). In other words, as the saliency of the target relative to the distractors decreases, participants shift from a parallel search strategy where large sections of the environment are scanned and the target "pops out" to the participant, to a serial search strategy where the participant inspects each item until the target is located (Treisman, 1986).

Augmentation is best suited for the tasks where the target has low salience and the participant must adopt the serial search strategy. For augmentation to be successful, the benefits in terms of speed and accuracy must outweigh the costs created dividing the user's attention. Serial search is slower which means there is more opportunity for gains through the addition of virtual information.

In the current experiment, participants were asked to find an “E” or a “3” made from five line segments from among a large field of “8”s made from seven line segments, the same style of targets and distractors used by Yantis and Jonides (1984). These stimuli result in serial search behaviour.

To assist device users in the visual search task, software designers must have some method of directing attention to a particular region in space. Previous research divides attention directing cues into two classes (Posner, 1980). Cues which direct attention automatically and involuntarily are labelled ‘exogenous’. Exogenous cues include any cue which causes the creation of a new perceptual object within a user’s field of vision (Yantis, & Jonides, 1996) or changes an existing object’s properties provided the user’s is aware of the property’s predictive value. For example, changing the colour (Theeuwes, 1994) or luminance (Müller & Rabbitt, 1989) of a target will automatically draw attention if a participant is instructed to search for it, even if colour or luminance fail to capture attention when no incentive is given to attend to them (Jonides & Yantis, 1988). Cues which do not draw attention automatically and must be processed before they are followed are labelled ‘endogenous’. Endogenous cues include symbolic representations of cues such as grid references, numbered target positions or arrows pointing towards a target (Jonides, 1981). Exogenous cues have been shown to be faster at directing attention than endogenous cues (Müller & Findlay, 1988; Müller & Rabbitt, 1989). Additionally, the reflex to attend towards changes in the visual field is difficult to suppress making exogenous cues a highly reliable method of directing attention (Jonides, 1981; Yantis, & Jonides, 1996).

Typical exogenous cues are not suited for current HMDs, however, because of two major limitations. First, this type of cue only works when the target location falls within the participants’ peripheral vision. When working in real-world environment, there will be many times when targets of interest might appear anywhere around the user, especially when the user is given free motion to navigate the environment. Secondly, and most importantly, this type of cue only works when a cue can be created at the point in the environment to which attention is to be oriented. Current HMDs have screens covering only a small region of the user’s peripheral vision and therefore abrupt onset

peripheral cues could only be overlaid onto the environment within this domain. This makes abrupt onset peripheral cues unviable for directing visual attention.

Our previous research attempted to replace the fast reflexive response generated by peripheral abrupt-onset cues with a centralised cue using the smooth motion pursuit-tracking reflex (Ward et al., 2016). Instead we found that for simple directing of attention, the arrows, a centralised endogenous cue, enabled participants to begin head-motion towards targets outside of their peripheral vision almost as quickly as they began motion towards highly-salient abrupt-onset cues appearing within their peripheral vision. Given the success of arrow-based voluntary attention, we decided to apply similar cues to augmenting user performance in visual search tasks while also investigating other voluntary methods of attention redirection.

We divided our cues for the current experiment into three classes based on the properties we expected in terms of the design trade-offs; time taken to process the virtual information, number of attention shifts required, information quality, and information quantity. Our first class contained two static information cues. The first of these was a simple arrow which pointed in the direction of the target from the center of the search space. The second static cue was a simplified map of the space, which marked in red one of 12 segments of the search space which would contain the target. We expect participants to be able to process both types of these cues quickly as arrow cues are overlearned stimuli (Gibson, Scheutz, & Davis, 2009), and the simplified map cue has only a small discrete number of possible modes. As the displayed information is static, participants are not expected to need to make many shifts of attention to or from the assisting device once the initial direction has been established. The fast processing and a low number of attention shifts is expected to come at the cost of information quality.

The second class contains two dynamic arrow cues. The first dynamic arrow maximised information quantity by giving vertical direction and horizontal direction using its orientation and distance to the target using size. Pilot testing of this cue found participants preferred to center the device's display over the target but doing so obscured the target with the shaft of the arrow. To avoid

fighting against the natural affordance of the cue, this version of the cue displayed only the arrowhead, leaving the center of the display clear. The second dynamic arrow traded information quantity for additional clarity by replacing the vertical direction dimension with the distance to target dimension, removing the need for participants to interpret varying arrow sizes. This resulted in an arrow that pointed in the horizontal direction of the target from the center of the participant's vision and gradually stood more upright as the participant moved closer to the target. If the target was in the top 5 rows, the arrow rotated until it was pointing up. If the target was in the bottom 5 rows it rotated such that it was pointing down. Because these cues continued to update during the trial participants are expected to make multiple shifts of attention between searching the space and the virtual display. Our primary interest in conducting this study is to determine whether the additional information afforded by dynamic updates to the cues during the trial outweighs the costs of multiple attention shifts encouraged by the presence of updating information.

The final class of cues contains two cues which do not need to be directly attended to. These cues use luminance and colour to encode the dimension of distance from the participant's center of view to the target. Luminance and colour are two features which can be detected using peripheral vision alone provided a target is of sufficient size (Hansen, Pracejus, & Gegenfurtner, 2009; Johnson, 1986). This means that participants may not need to make any shifts of focal attention to retrieve information from these cues. This comes at a cost of information quantity, only simple properties can be used to encode information in this way, and there is a time cost associated with the need to take multiple samples and refer to memory to establish whether movement is toward the target. The first of these cues uses a full screen height white circle which transitions from completely transparent at a 60-degree rotational displacement from the target, to maximum Glass luminance when the target is at the center of the participant's field of vision. The second uses the same circle but instead transitions from green to red as the participant converges on the target. Our secondary interest in conducting this study is to determine if peripherally displayed information can augment search behaviour and if

so whether minimising the demand for the participant's focal attention is worth the trade-off for increasing the difficult of processing the received information.

A seventh condition provided the participant with no assistance in searching for the target. We hypothesise that all three classes of cues will augment the participant's basic search behaviour and improve search times. However, there is a possibility that the addition of virtual information to the task and the increase in distraction and visual clutter may instead reduce participant performance. This condition allows quantification of the improvement of the various cues compared to an unassisted control.

Methods

Participants

Twenty-eight volunteers (13 Male, 15 Female) completed all 7 trial blocks of the study. All of the participants were naïve to the use of the Google Glass device prior to the experiment. All participants reported normal vision, or corrected to normal vision using contact lenses. Corrective eye-glasses could not be worn beneath the Google Glass. The average age of participants was 26.4 years (SD = 6.2, Range = 17 - 45). Participants were recruited from the University of Canterbury Campus and were compensated with a $10NZ in shopping voucher.

Stimuli and Apparatus

Tracking data were collected using the Dtrack software (Ar-tracking.com, 2015) and a four ARTRACK2 camera array. Tracking data were transferred using the VRPN software (GitHub, 2015). Positional trackers were 3D printed on site and marked with reflective material. Central computation for the experiment was run using custom code written in the Unity 3D engine (Unity3d.com, 2015). Far domain stimuli were presented on three 2440mm (60.3° visual angle) by 1830mm (47.1°) screens connected at 60 degree angles mounted at 600mm above the floor. Images were projected on to the screens by three NEC LT265 projectors. Near domain stimuli were presented on a Google Glass

(Google, 2017; Starner, 2013) connected via a Nexus 5 android smartphone as a bridge. The viewing angle covered by the Glass varies based in the participant's eye to nose bridge distance, but has been estimated to be at 13° by 7.3° of the user's visual angle (Hwang & Peli, 2014). The bridging program converts a wireless network connection from the experiment PC to a Bluetooth connection with the Glass. Responses were made using a handheld trigger topped with a positional tracker. The connections between these devices are shown in Figure 4.1.

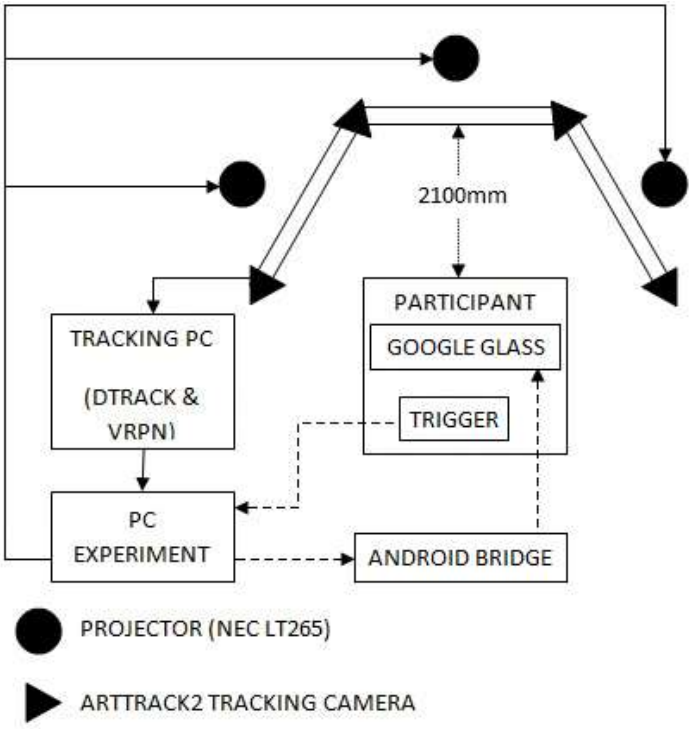


Figure 4.1: Block diagram for the layout of experiment components.

The projections out from the participant's head tracker and hand-held trigger tracker were marked on the projector screens by a pair of disks 33mm (0.8°) in diameter marked yellow and blue, respectively. Distracter stimuli were figure 8s constructed from 7 straight line segments. Target stimuli had either the left most or right most vertical segments removed to create a 3 or an E shape

respectively. The order of 3's and E's was chosen pseudo-randomly to ensure an equal number of each was presented in each block.

Guide markers on the Glass screen included an arrow, approximately 3.0° by 6.3° of visual angle, used in conditions 1 and 5; a disk, 7.3° by 7.3°, used in conditions 3 and 4; an arrowhead, 2.1° by 2.1°, used in condition 6; and three connected squares, 7.0° by 2.4°, with a thickness of 0.2°, and with a quadrant of one of the three squares marked in red, used in condition 2. Figures 4.2a-d show the implementation of these cues on the Glass Screen. Participants were also fitted with a palm sensor which measured heart rate and skin conductance. Data from this device is not used in the current analysis.

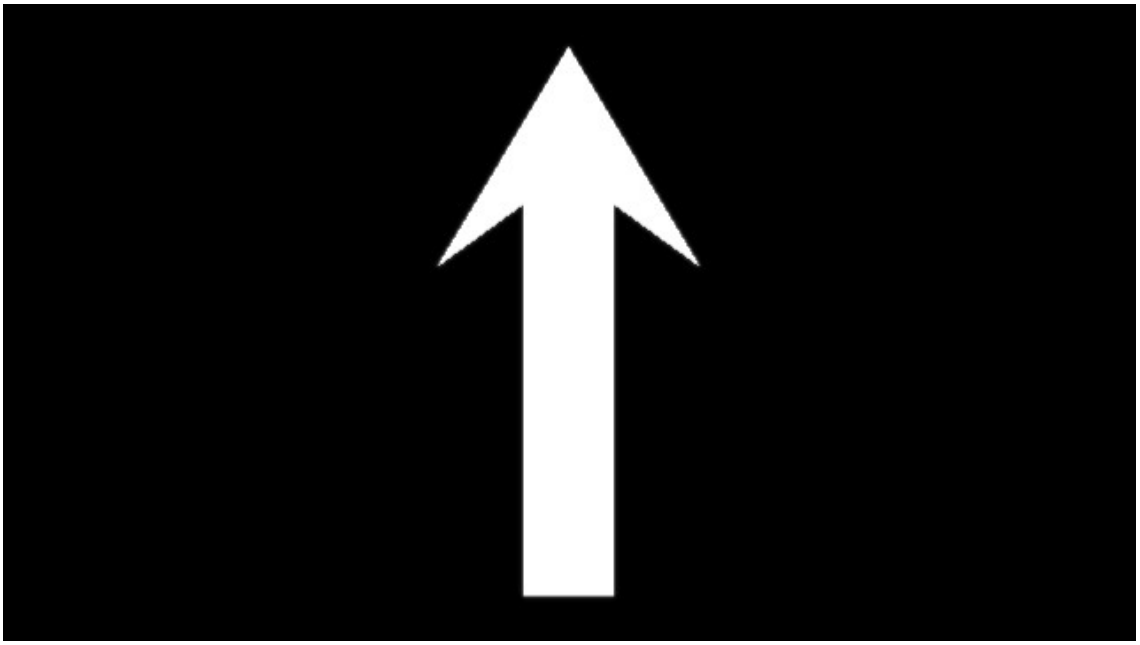


Figure 4.2a: An arrow cue as displayed on the Google Glass. Black regions are transparent. The arrow rotated around a central axis point.

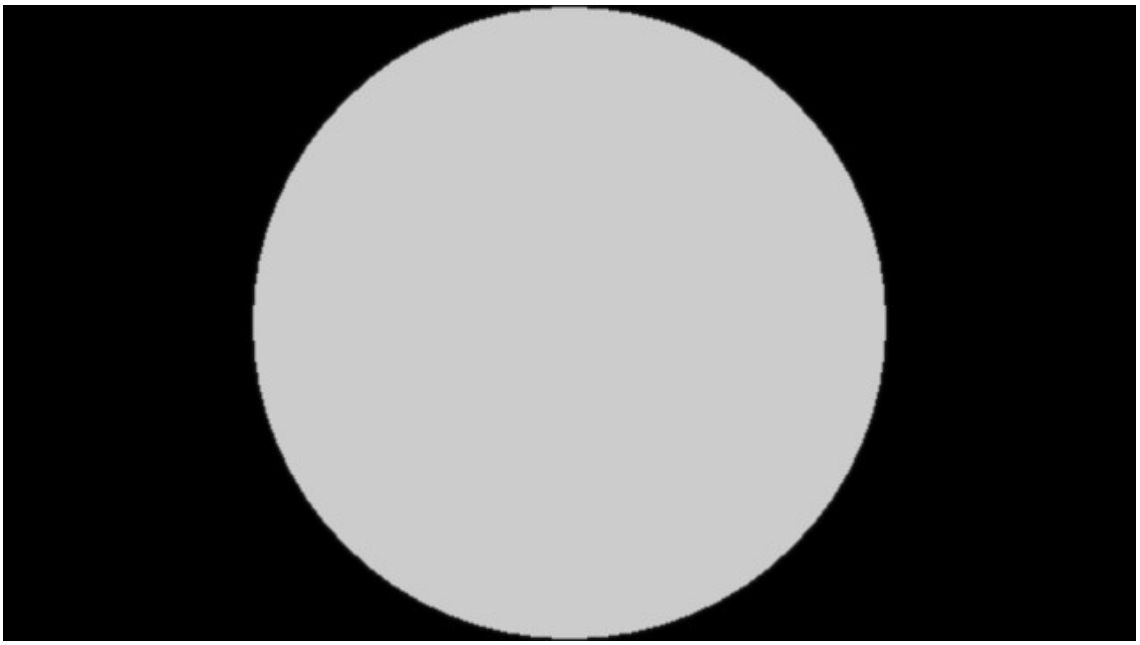


Figure 4.2b: A luminance or hue cue as displayed on the Google Glass. Black regions are transparent. The circle varied in opacity (0%-80%) or in hue (0-108 on a 360-point scale).

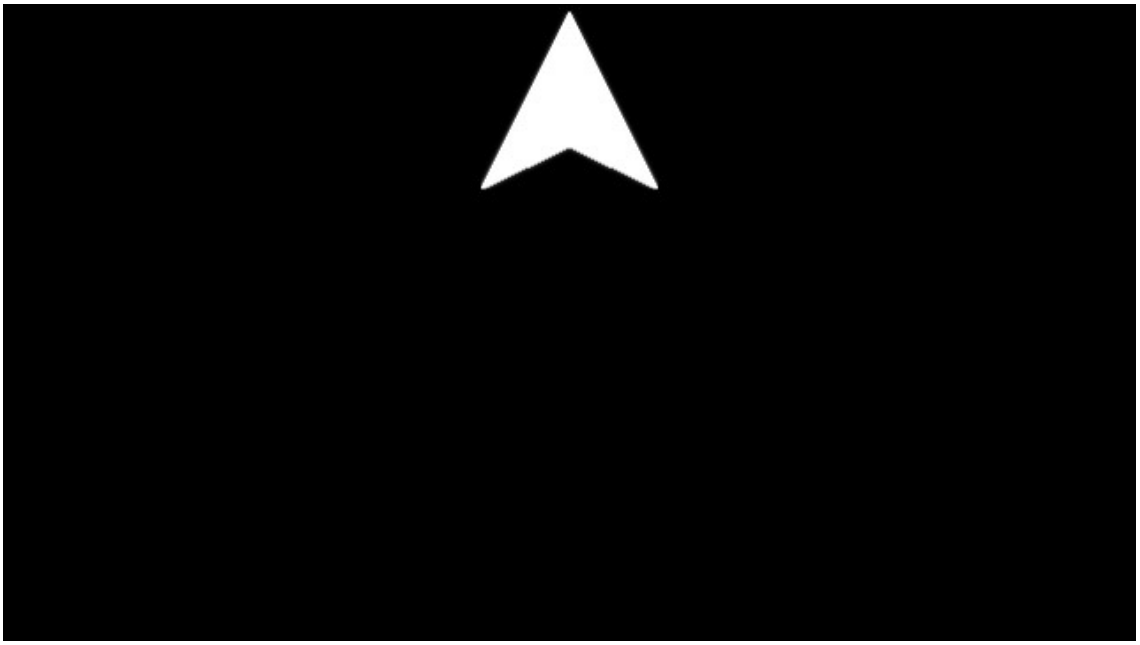


Figure 4.2c: An arrowhead cue as displayed on the Google Glass. Black regions are transparent. The arrowhead rotated the edge of the screen and its size varied based on the distance to the target.

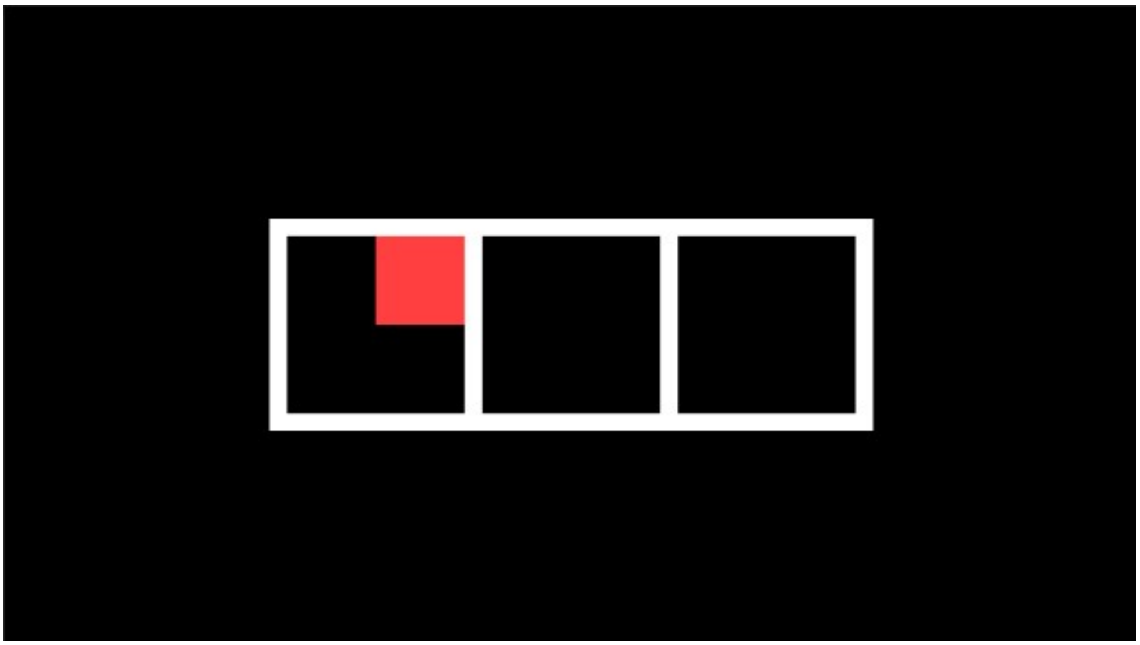


Figure 4.2d: A simplified map cue as displayed on the Google Glass. Black regions are transparent. The target is within the marked screen quadrant.

Procedure

Participants were seated on a rotating chair 2100mm back on the normal line extending from the center of the middle screen such that targets could be displayed anywhere on a 180° horizontal arc. Participants were allowed to adjust the seat height for their comfort before being fitted with the Google Glass, a cap with a reflective position tracker, the palm sensor and the handheld trigger. A short calibration process ensured the projection from the head tracker aligned with the center point of the participant's vision.

Participants were required to complete 7 conditions presented in a counterbalanced order to ensure each block preceded and followed each other block an equal number of times (rotation size: 14). Each block was 5 minutes in length to ensure the entire task did not take longer to complete than the Google Glass' expected battery life (45-50 minutes in test conditions). In all 7 conditions, the participant was required to locate the single unique target in a field of 299 distracter stimuli (10x10 on each of 3 screens) and then pull the hand trigger while pointing it at the unique target. A one

second break would then occur, replacing all stimuli with black dots, and then the task would continue with a new unique target placed pseudo-randomly such that in every 30 trials, 10 are placed on each of the 3 screens.

In condition 1, the Glass screen displayed a static arrow pointing in the direction of the target from the center of the search space (Figure 4.2a). In condition 2, the Glass screen displayed the three connected white squares with the red quadrant corresponding to the quadrant of the screen which contained the unique target (Figure 4.2d).

In condition 3, the Glass screen displayed the disk coloured white which increased from 0% luminance when the head tracker projection was at distances greater than 50° from the unique target, to 100% luminance when the head tracker projection was directly over top of the target (Figure 4.2b). The luminance increased following a squared power relationship to offset the diminishing sensitivity to brightness documented by Fechner (1966), Stevens (1957), and many others. In condition 4, the Glass screen displayed the disk which changed in hue from green when the head tracker projection was more than 50° from the target, to red when the head tracker projection was directly over top of the target.

In condition 5, the Glass screen displayed a dynamic arrow which pointed either left or right in the direction of the target if the horizontal angle between the head tracker projection and the unique target was greater than 50° and the began to stand more upright as the horizontal angular difference decreased. If the target was in the 5 rows the arrow would point directly upwards once the head projection was in line with the unique target, otherwise the arrow would point directly down (Figure 4.2a). In condition 6, the Glass screen displayed a dynamic arrow head which rotated around the edge of the screen such that it always pointed in the direction of the unique target from the head tracker projection (Figure 4.2c). Once the direct angular difference from the target to the projection was less than 25°, the arrowhead began to shrink down until it was invisible.

Condition 7 was a control condition. The Glass screen remained blank during the block.

Post experiment, participants were debriefed and given a short questionnaire on which they were asked to rank their preference for the 7 difference aid types and to note whether they referred back and forth to the glass, looked once or twice and then focussed on the search, or ignored the cue entirely for each block.

Results

Response time distributions for each participant had a strong right skew which is common with reaction times. The following analysis was conducted using the participants' median response times to reduce the impact of outlying extreme values. Inter-condition analysis was conducted using repeated measures ANOVA and planned linear contrasts. Analysis of the distribution of pooled individual response times was conducted with linear regression and the slopes and intercepts of the resultant lines were compared through the addition of dummy-coded interactions and main effects. Speed and preference rankings were analysed with Spearman's rank order correlation coefficients.

Within the static cue types, condition 1, the static arrow, provided an initial direction for the participants to follow while condition 2, the simplified map, provided a position to orient towards. The median times for the participants to find the unique target was faster when the cue oriented the user to a position than in a direction, $F(1,27) = 38.01$, $p < .001$, $\eta^2_p = .585$, although this difference was smaller when the target appeared within the participants approximate binocular peripheral vision, $F(1,27) = 19.59$, $p < .001$, $\eta^2_p = .420$.

Within the peripheral cue types, angular distance between projected head position and the unique targets was marked with luminance in condition 3 and hue in condition 4. There was no difference between the median search times for the two peripheral cue types, $F(1,27) = 0.73$, $p = .399$, $\eta^2_p = .026$. This was unaffected by whether the targets appear inside or outside the participants' binocular peripheral vision, $F(1,27) = 0.77$, $p = .386$, $\eta^2_p = .028$.

Within the dynamic cue types, the condition 5 arrow used the two dimensions provided by rotation to encode horizontal direction and angular distance to the target for a total of two task-

relevant dimensions. The condition 6 arrowheads encoded horizontal and vertical direction into position and rotation and encoded the distance to the target into the size of the arrowhead for a total of three task-relevant dimensions. Participants' median search time was shorter for the two-dimension condition, $F(1,27) = 11.98$, $p = .002$, $\eta^2_p = .307$. Similar to the static cues, the difference in search times was smaller between condition when the target appeared within the participants' peripheral vision, $F(1,27) = 9.35$, $p = .005$, $\eta^2_p = .257$.

Comparing between cue types, the simplified map cue enabled faster median search times than the second fastest cue: the 2-dimension dynamic arrow, $F(1,27) = 6.03$, $p = .021$, $\eta^2_p = .183$. This difference was independent of whether the target appeared inside or outside peripheral vision, $F(1,27) = 1.99$, $p = .170$, $\eta^2_p = .069$. There was no difference in the median search speeds of the 2-dimension dynamic arrow and the static arrow, $F(1,27) = 2.04$, $p = .164$, $\eta^2_p = .070$, although a small difference appears when the comparing targets that appeared outside the approximate range of the participants' peripheral vision, $F(1,27) = 3.57$, $p = .069$, $\eta^2_p = .117$. All cues outperformed the control condition, although for the peripheral cues this was only true when the targets appeared outside the participants' peripheral vision, $F_{7\,v\,3,4}(1,27) = 12.37$, $p = .002$, $\eta^2_p = .314$. The average median response times and subjective preference rankings for each condition are displayed on the left side of Table 4.1.

Participant preference for the cue types only had a medium correlation with their ranked speeds, $\rho_{CI} = [0.33,0.61]$, with participants overrating the value of the peripheral cues (C3 average preference ranking = 3.6, average speed ranking = 2.7; C4 average preference ranking = 4.1, average speed ranking = 2.1) and underrating the two static cues (C1 average preference ranking = 3.6, average speed ranking = 4.6; C2 average preference ranking = 5.9, average speed ranking = 6.7).

Table 4.1:

Subjective Participant Rank Preferences (1 = Least Preferred, 7 = Most Preferred) and Averages and Standard Errors for Participants' Median Search Times Overall and When Appearing Inside and Outside Binocular Peripheral Vision.

	Average Participant Preference Rank	Average Median Response Times (ms)	Average Median Response Times to Targets Inside Peripheral Vision	Average Median Response Times to Targets Outside Peripheral Vision	Increase in milliseconds for each degree distant from the target at trial onset	Estimate of average search time for targets directly in front of participant
C1: Static arrow	3.52	3888 (368)	3178 (221)	4941 (466)	26.1	1666
C2: Simplified map	5.89	2862 (183)	2641 (165)	3454 (402)	11.8	1895
C3: Luminance P.	3.56	4984 (346)	4484 (325)	5712 (404)	21.9	2802
C4: Hue Peripheral	4.11	5012 (242)	4542 (260)	6280 (581)	30.2	3023
C5: 2trD Arrow	4.81	3463 (137)	3010 (130)	4219 (188)	22.7	2082
C6: 3trD Arrow	4.11	4063 (293)	3399 (240)	5294 (394)	27.4	2007
C7: No Cue	2.00	5375 (330)	4358 (360)	7320 (413)	58.6	1734

Discussion

Users must divide their attention between virtual information and the real world when being assisted on tasks with augmented reality programs. To overcome the costs of dividing attention programs can be designed to minimise the number of attention shifts required, increase the value of the information displayed for completing the task, or reduce the amount of time it takes to process the information. Three classes of centrally positioned cues were tested in this study; static cues, which only required one attention reorientation to use; dynamic cues, which gave more precise, up-to-date information at the cost of requiring multiple attention shifts; and peripheral cues, which gave less precise information to allow the cue to be interpreted using only peripheral vision.

Overall, the static cues performed best with the simplified map cue resulting in faster search times than any other cue and with the static arrow cue coming in third while not being significantly slower than the second place overall and best dynamic cue: the two-dimension encoded dynamic arrow. The static cues also have an additional advantage in that they require less precise tracking, if at all, of the user within the search space as the cues are relative to the environment rather than the user's current position.

Experimental design factors contribute to the success of the two static cues. The targets and users did not move relative to one another over time reducing the amount of value added by having up-to-date information, the expected advantage of dynamic cues, while not adjusting the time taken to shift attention or the clarity of the information. Additionally, the targets had high salience, as shown by the similar reaction times across static cued, dynamic cued and no cue conditions when responding to a target that appeared directly in the centre of participants' view. This means participants are unlikely to overshoot or fail to detect a target and need to restart their search, again reducing the added value of up to date information. The two-dimension dynamic arrow had fewer extreme search times and a slightly lower median search time for targets appearing outside peripheral vision than the static arrow, showing it did gain some advantage from this property, however.

Comparing the dynamic cues, the two-dimension dynamic arrow was faster than the three-dimension dynamic arrow despite lacking information on the dimension of vertical displacement. This means the information quality added by vertical displacement information did not overcome the trade-off cost in time needed to parse the additional information. The success of the simplified map cue shows that participant's do not necessarily need to be guided to the exact position of the target. Instead it is sufficient, and in cases similar to this, advantageous, to instead use the cue to restrict the area of the participant's search and let their unaugmented search behaviour make the final detection. This is a promising result for augmented reality assistance as the goal of the field is to improve and assist, rather than completely replace, the user's default learned behaviours.

Reducing the number of distractors has been consistently shown to reduce response times in visual search tasks (Wang, Cavanagh, & Green, 1994; Wolfe; 1994; Wolfe, Cave, & Franzel, 1989) and the three fastest cue types in this experiment all functioned by restricting the search area and excluding distractors rather than leading to a unique point. Too much precision can cause the user to over-rely on the device. When the Glass led to the precise position, users reported attending only to the device until they had directly centred their field of view on the target. They were reluctant to stop using the device until all the possible information on it had been delivered. By attending solely to the

device until no more information could be received, participants deprived themselves of any benefits from the passive scanning while they conducted their primary initial search. The negative impacts of over-reliance on an assisting device can be reduced by forcing the user to make the final decisions or actions (Onnasch et al., 2014; Wickens, 2017). The most successful cue types in this study did this by restricting the search area and then having the user make the final search actions alone.

The peripheral cues were unsuccessful. These two cues were conceptualized as a method of giving a single dimension of information to the device user using luminance or hue, two features of an object that can be detected using peripheral vision. Participants did not use the cue in this way however. Interviews and discussions with participants' post-experiment revealed that the majority of participants instead focussed their full attention on the device until the glass was lit sufficiently white or red for conditions 3 and 4 respectively, and then shifted their attention to the screens. This relatively inefficient double search, once of the virtual space, and then once in the real-world space, along with the absence of an initial direction for their search made the participants slow to find targets even when they appeared within their approximate binocular peripheral vision range. In this way, the peripheral cues functioned as dynamic rather than peripheral cues and had both reduced information value and information clarity as compared to the designated dynamic cues.

The simplified map cue was the most successful centralised cue out of the six tested in this study. Participants found the unique targets when guided by this cue faster than any other, regardless of whether the target fell inside or outside their peripheral vision at the time of onset. Subjectively, it was the top ranked cue type for 15 out of the 28 subjects and second place for an additional 6. The simplified map did not direct attention precisely to the target's location, but in trade-off for information quality it had advantages in the number of attention shifts needed to use the information it provided and the amount of time it took to process the information. Because the information was static, all the information could be obtained with a single attention shift after which attention could be focussed on the search environment. Also, processing the information was quick because there were only 12 possible outcome cases and the response to each case was identical each time. An arrow

leading to the right while attending to the left screen could require the participant to attend to the center screen or the right screen; the termination point for the required movement is at first unknown, or at least uncertain. For the simplified map, the termination point was always clear meaning the participant could make confident movement to the region containing the target, skipping the intervening space without risking overshooting the target's position. Because of this, the simplified map cue was least affected by the distance to the target at onset and was less affected by targets appearing outside the participant's field of vision.

Typical exogenous cues move attention by first capturing covert attention with a discontinuity in the participant's visual field at the desired attention location (Posner, 1980; Posner, Nissen and Ogden, 1978). Once the termination point has been set with a covert attention shift, overt attention the follows with a saccade of the eye (Shepherd, Findlay & Hockey, 2007). In this way, the simplified map cue causes a similar response to exogenous attention directing cues while still being subject to an endogenous internal locus of control.

In our previous paper (Ward, Barde, Russell, Billinghurst, & Helton, 2016), we argued that it is not necessarily the degree of autonomy of the response elicited by a cue that predicts a participant's performance. With the current results, we now add it may be that cues which direct to a point restrict and focus the subject's attention to a particular space faster or more precisely than directional cues; and also that abrupt onset cues are just particularly precise and easily parsed point-directed cues. Some support for the theory that the destination is more important than the journey when it comes to orienting attention comes indirectly from the domain of human-computer interaction user input research. In this field research consistently shows participants are faster and more precise when aiming with control systems that use displacement/point-mapping such as a computer mouse or stylus rather than velocity/direction-plus-impetus mapping such as a joysticks and trackballs (Card, English, & Burr, 1978; Klochek & MacKenzie, 2006; MacKenzie, Kauppinen & Silfverberg, 2001). MacKenzie, Kauppinen & Silfverberg (2001) attributed a decreased speed in aiming tasks with the joystick and trackball velocity-based inputs to an increased likelihood that a user veers off the optimal track

towards the target and over shoots the position of the target more frequently, among other variables, than with point-targeting devices. For conditions in which the participant was given both distance and direction information but no marked stopping point it is possible analogous errors were made; letting their search path drift up or down a row, or overshooting the target before noticing that the cue on their device has reversed direction.

It should be noted, however, that while it is the finding that the point-directing simplified map second condition outperforms the other cues that leads to the idea that it is the destination versus direction properties of a cue that predicts performance, it does not directly test the idea. Condition 2 is the only destination-orienting rather than direction-orienting cue in this study and abrupt onset cues are not tested directly. With only one type of destination orienting cue it would be an overgeneralization to extend the success of this one type of cue, dominant as it may have been, to an entire category of attention directing cues.

Additionally, despite the relatively poor performance of all the arrow cues, conditions 1, 5 and 6, further research is still required into optimising the design of centralised, direction-orienting cue. This study was conducted because the previously established optimal attention-orienting cue, the abrupt onset cues (Jonides & Yantis, 1988; Müller & Findlay, 1988; Müller & Rabbitt, 1989; Theeuwes, 1994) could not be used as designed on current generation wearables such as the Google Glass due to the restricted region of control, i.e. the 13° by 7.3° transparent display, available to software designers. Similarly, there will be times that cues similar to the simplified map cues in condition 2 will not be viable; situations where the environment may be too complex or layered to display on a small monocular screen, or where there is simply no available method to map the environment in real time. In situations destination-orienting cues would lose the environmental reference points necessary to narrow the users' focus on the correct section of the search domain. Future research should test if destination-orienting cues like the simplified map continue to function on more complex search environments. When searching for an object within a more complex environment, dynamic information will likely have more value. When targets are more likely to be missed, the user will be

more likely to need redirection. Studying a more complex environment will help develop a clearer picture of the situations in which static or dynamic arrows cues should be used.

When the symbolic representation of the search space is unavailable, a developer needs to choose between a static or dynamic direction-orienting cue. There was no statistically significant difference in speed between the direction-orienting static cue in condition 1 and the best dynamic cue tested: the two-dimension encoded dynamic arrow in condition 5. Both the dynamic cues were preferred by the participants despite the similar performance to the static cue. Therefore, our recommendation to designers of AR programs for HMDs is to use a symbolic representation of the search space to guide users when one is available. Failing that, if a user's position and orientation can be tracked within the space, use a dynamic arrow which encodes the horizontal direction and distance to the target. And finally, if neither tool is available, use a static arrow cue from a fixed point in the environment.

Chapter 5: Extending HMD Delivered Visual Cues for the Augmentation of Visual Search to a Real-world Environment

Preamble

HMDs are designed for use even while the wearer is active. Our experiments up until this point have had a stationary, sitting participant searching for targets on a singular depth plane. This is not a strong analogue to the real-world tasks that HMD designers will be building their software to assist with. Our last two experiments test the two most successful endogenous class cues from the previous experiment in a more naturalistic setting. Participants will be required to walk through and avoid obstacles in a 3D environment and the cues were expanded to account for target depth within the environment.

The second of the two remaining experiments goes one step further and compares the successful, HMD-delivered, endogenous cues against a traditional, manually-placed, exogenous cue which taps into the orientation reflex. The primary goal was to find a replacement to the traditional exogenous cue for HMDs, as HMDs do not give the software designer the control of the environment needed to create this style of cue. If the HMD-delivered cues can direct participants through the space as quickly, or at least close to as quickly, as the exogenous cues then they can be recommended as a replacement for when a designer needs to quickly direct the user's attention from the HMD out to a particular region in real-world space.

Introduction

Head mounted wearable computer devices (HMDs) such as the Google Glass (Google, 2017) move a part of the functionality of a smart phone out of the user's hands and on to a transparent screen that overlays the user's field of vision. This move offers multiple advantages; first and foremost, it leaves both of the user's hands free to interact with their environment. This opens up the possibility of a wide number of tasks in which the user must use virtual information from their device to complete

a real-world task. These situations include cases where the users' needs to assemble an object based on diagrams delivered by their device (Tang, et al., 2003) or telemedical surgery, where a surgeon performs a procedure with the guidance of a remote specialist (Armstrong, et al., 2014). In both these cases and many more, the ability to access virtual information without a device occupying the hands is essential.

An additional advantage of HMDs is that an overlaid transparent screen means that virtual information can be added to the user's visual field when necessary without requiring action from the user. An unexpected, task relevant, text message presented by smartphone requires the device to be retrieved before it can be used. When the use of a device is unexpected, the majority of users keep their smartphones in either a pocket or a bag, unavailable for immediate use (Cui, Chipchase, & Ichikawa, 2007). HMDs, by comparison, can be woken by the incoming message and the information is then displayed to the user immediately.

The major limitation of HMDs compared to handheld devices is their reduced range of input options. Both HMDs and handheld devices share access to voice command inputs, but these are not suitable in many situations such as in noisy or public environments. Interactive touchscreens like those found on smartphones and tablets allow programs with complex button-based responses while current generation HMDs have only a small trackpad with a limited range of possible inputs. HMDs can replace this type of input with gesture controls (Lv, et al., 2015), but these remove the major advantage of the HMD of freeing up the hands. These features and limitations together give HMDs an advantage in situations where the user's primary attention and efforts center on the real-world environment, while the device provides helpful or necessary assisting information and requires only limited input from the user.

There are additional limitations to adding the use of any device to a task. Adding new information or objects to the user's visual space, whether virtual or physical, increases the number of regions in the space the user must attend to. Visual attention consists of two components; covert attention, the internal locus of attention, which determines which volume in 3D space gets the largest

share of visual processing resources; and overt attention, the bearing and depth focus of the eyes, which determines with region in space has the most precise level of visual discrimination (Intriligator & Cavanagh, 2001; Posner 1980). Both components of visual attention need to be deployed in series. The eyes cannot be pointed at two locations at once and while the region of covert attention can be expanded to allow parallel processing of multiple proximal features of a surface, it cannot be split between two separate regions (McCormic & Klein, 1990; McCormick, Klein & Johnston, 1998) and suffers performance costs when split between surfaces or objects that share the same space (Ernst, Palmer & Boynton, 2012; Rodrıguez, Valdes-Sosa & Freiwald, 2002). This means that any time spend by the user attending to information presented on devices comes at a cost of time spent attending to the primary real-world task. While it is possible that the value of the virtual information exceeds the costs, it is important to ensure this is the case for any software design for augmented performance via HMDs.

An increase in the amount of visual information provided to the user can also cause the volume of visual attention to narrow. Tightening the focus of visual attention can help the mind filter out proximal distracting information (Eriksen & Hoffman, 1972). Adding new information to the user's visual field increases the density of information and makes this narrowing more likely. This narrowing of attention, and the consequent increase in feature selectivity, leads to the phenomena of inattentional blindness (Simons, & Chabris, 1999). Hyman, et al. (2010) performed a study in which participants were asked to travel through a real-world space while using electronic devices such as cell phones and MP3 players. Participants using cell phones were more likely to have near collisions with other pedestrians, less likely to acknowledge the presence of other people and were the least likely of any group in the research to detect highly unusual salient events. This decrease in environmental awareness while traversing real-world spaces creates a real and present danger for device users (Stavrinos, Byington & Schwebel, 2011). An analysis of hospitalization rates from 2004 through to 2010 has found that use of a mobile device is becoming an increasingly common factor in injuries for pedestrians (Nasar & Troyer, 2013).

The light weight and small size of the HMDs encourage users to continue to wear them while active. This means any program developed for an HMD will likely be used by a moving person. There will also be times where the costs of attending to a HMD while moving will be unavoidable. For example, museum tours, offered through an HMD, will necessarily require the user to navigate a real-world environment while receiving direction and exhibit information through the device. As a museum is a public place, the environment will often be too noisy for audio direction alone and therefore visual guidance and information is required for optimal functioning of the program. For these reasons, it is important for researchers to investigate the best practices for minimizing the distraction created by HMD use for users who need to concurrently navigate a real-world environment.

One common task which can be aided by an HMD and creates the need to navigate an environment is searching for an item. The most effective method for guiding a user's attention to the location of an item of interest is to use an exogenous cue (Müller & Findlay, 1988; Müller & Rabbitt, 1989; Posner, 1980). Exogenous cues are highly salient events which occur at the location of the target, such as a sudden change in colour or flash of light. These events create an orientation reflex which quickly and almost unavoidably draws attention to the desired location. These exogenous cues are unsuitable for current generation HMDs, however. The currently available HMDs offer only a small display window. The display is also monocular, removing the opportunity to use parallax to simulate varying depths. This leaves only a small area of control within the environment available to a software designer. This means unless a participant is already orientated towards the target item and has sufficient contextual information to determine the object's depth, it is unlikely an exogenous visual cue will draw the user's attention.

Our previous studies (Chapter 3, Chapter 4) investigated alternative cueing methods for HMDs, first by trying to find a reflexive cue to replace the common form of the exogenous cue and then later testing various endogenous cues for guiding the user's attention. These studies were completed in a laboratory setting with a seated participant. We found that a reflexive response to a HMD-displayed cue was not sufficient, nor necessary, to efficiently guide attention. Instead centrally

controlled and voluntarily followed endogenous cues started the participant's search a negligible amount more slowly than traditional exogenous cues (Chapter 3).

When comparing various types of endogenous cues, we found that it was not necessary to guide a participant to the exact location. Instead it was sufficient to move the participant's attention to the region containing the target and let their practiced natural visual search behaviours find the exact position of the item. Doing so led to faster search times than when a participant needed to make multiple attention shifts to and from the device to determine how close they were to the target. The most successful of the cues was a simplified map of the environment despite the cue also being the least precise in terms of leading to the exact location of the target. The two most successful cues, the simplified map and an arrow which encoded horizontal direction and distance, gave the user a measure of the distance to the target which was available as soon as the cue appeared. For the map, this distance estimate was clear and locked to the environment, and for the less successful horizontal distance arrow the participant needed to infer the distance to the target based on the slope of the arrow. The three most successful cues, adding the environment static arrow which pointed from the center of the screens out in the direction of the target, shared a second property in that their information was front-loaded. All of the information from the three cues could be retrieved on the first attention shift to the device and held in memory. The remaining three cues tested all required subsequent attention shifts to the device to check for relative changes in the cues brightness, hue or size to ensure the participant was moving in the correct direction or had not overshot the target. These additional attention shifts slowed down participants and drew attention and time away from their natural, practiced and relatively successful search strategies (Chapter 4).

In the current study, participants were required to perform a search task while navigating a real-world environment. To adapt the most successful map cue to account for target depth within the space, the map was reoriented to a top down display. This was chosen because the height dimension is the least likely spatial dimension to encode information in most real applied settings. Most items a user might be searching for will fall somewhere up and down within arm's reach. As found in the

Chapter 4, it is sufficient to lead the user to the correct area and have them perform the final identification using an unassisted search strategy. Therefore, leading the user to the correct height is unnecessary in most contexts.

To adapt the next most successful cue, the horizontal rotation arrow, to account for target depth, we first simplified the rotation arrow to use stretch rather than orientation to indicate distance to the target. This freed the vertical dimension of space on the Glass' display to display a second arrow in a different style to mark depth. Preliminary testing found this secondary arrow as not effective at conveying a particular end point for the user's travel as it required participants to make a mental conversion between line length and distance travelled. For rotation, an arrow pointing halfway in one direction across the screen was easily interpreted as a half turn in that same direction, but for distance the conversion was less clear. The vertical arrow was replaced by the horizontal distance to the target, written as a number in meters above the arrow guiding rotation.

We predict the same pattern of results for the two visual cues as those found in Chapter 4; The map cue, with a clear end point will be the fastest cue to lead participants through the space to the target. This is because the map cue only requires the participant to make a single attention shift to and from the device and gives the participant a clear end point at which to terminate their movement when planning their route through the space. We expect the arrow based cue to lead participants though a space significantly faster than making a hidden target search with no cue. We also expect the arrow cue to lead participants more efficiently than any audio based cues which require participants to make extra conversions and judgments regarding distance and are therefore more likely to overshoot the target. Participants are expected to need only to check the dives a couple of times when presented with the arrow cue: once to determine the initial heading and give an approximate end point to begin manual search; and the second time when close to their expected end point to check for extra clarity and to ensure they have not overshot their target.

One factor that may influence performance and may equalize or even reverse the pattern of performance for the two visual cues is that route planning requires working memory (Oulasvirta, et

al., 2005). Furthermore, even when route planning is unnecessary and users are moving across a flat, obstacle free space, the physical exertion of moving also impacts working memory performance (Epling, et al, 2016). This competition in working memory resources may mean the advantage of the map cue that derives from requiring only a single attention shift may be reduced. Participants may find they need to make additional checks of the device to refresh their memory. A participant may remember the table they need to travel to but once they arrive find they need a second check to determine where on the table they need to search for the target. The search space is also more complex than the space presented in Chapter 4. This means the participants may take longer at the beginning of the experiment to map the top-down cue on to the real-world environment and they may be direct to the wrong search region more frequently.

In summary, the current experiment seeks to extend the two most successful endogenous cues from Chapter 4 to account for depth and then apply them to a search task in a real navigable world space. Our hypotheses are that the map-based cue which requires only a single attention shift and provides a clear travel end point, will lead the participants to the hidden target faster than any other cue. The arrow cue, with a travel end point which must be estimated by the participant, will lead to the target more slowly, but still faster than an audio cue which requires more estimation and interpretation to determine direction and distance to the target. Search times are expected to be faster in all cue conditions than a no cue control condition.

Experiment 1 Method

Participants

Twenty male and 16 female volunteers recruited from the University of Canterbury participated in the experiment. The participants had an average age of 25.7 years, ranging from 21-38 years. Their vision was normal or corrected to normal and all participants reported normal hearing. Participants were compensated for their time with a $10NZ shopping voucher. One female participant

was excluded due to a network connection failure which terminated the experiment after the second test block.

Environment and Stimuli

The testing region was a 4.2m by 6.2m rectangle. In this space were positioned 6 tables of various sizes covered with between 4 and 15 laminated cards for a total of 67 cards. Figure 5.1 shows the layout of the tables within the space. Each card was 12.7cm by 7.6cm and blank on both sides excluding a single target card which had a 6cm diameter green circle on a single face.

Visual cues for the experiment consisted of a map (Figure 5.2a) and an arrow compass (Figure 5.2b). The map cue represented the six tables as white rectangles on a transparent background and marked the position of the target card with a red dot. The arrow compass cue had an arrow that extended out to the left or right based on how much the participant needed to rotate to be looking towards the target card's location. Above the central dot of the compass was written the distance in meters from the participant to target, ignoring vertical displacement.

Sound cues for the experiment were two binaurally spatialized pings that lasted one second. Spatialization was achieved using the 3Dception plugin for Unity developed by Two Big Ears. One was a static sound source that was positioned at the target location and constantly pinged; the other was a dynamic stimulus that shot out towards the target from the participant's position. For the static sound cue, the volume of the cue increased as the participant approached the target similar to sounds created by real-world objects. The dynamic sound cue traced a parabolic trajectory to the target and was identical to the one used in chapter 3. For this cue, the length of the fall off period was varied to create the illusion of the sound travelling different distances and to match the distance from the participant to the target. The auditory cues were designed in accordance with the recommendations of Walker and Kramer (2006).

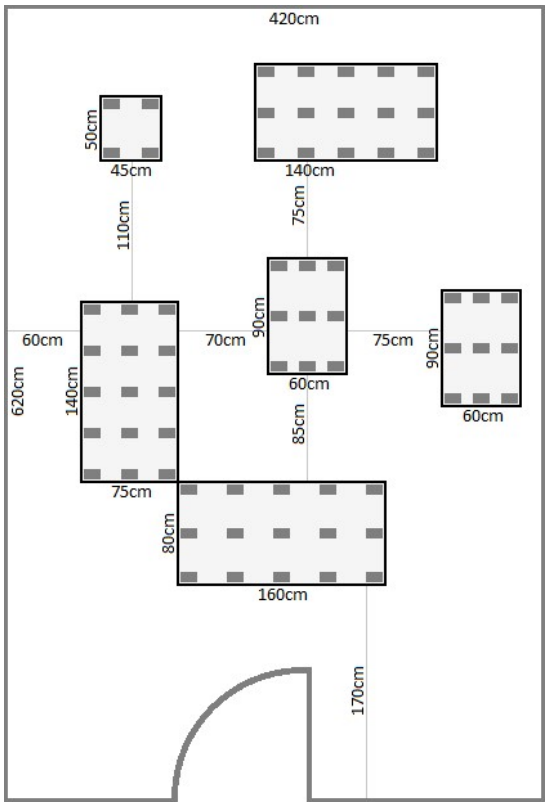


Figure 5.1: Environment layout to scale. Area includes 1 door, 6 tables and 67 blank cards.

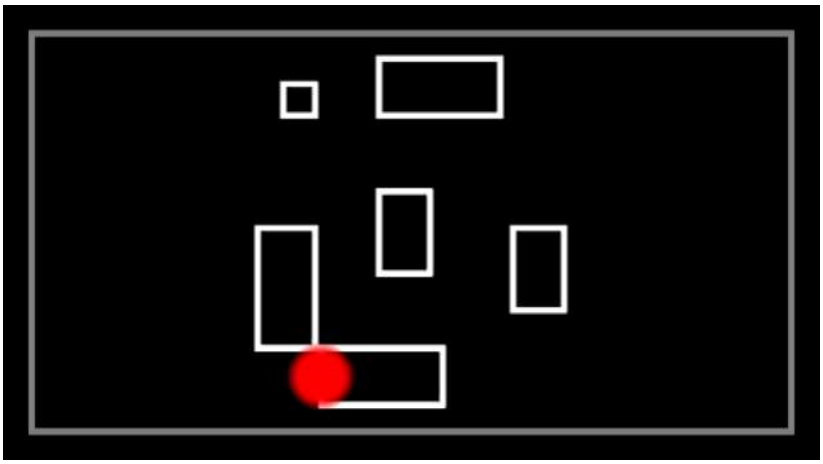


Figure 5.2a: Map visual cue as displayed on the Google Glass. Black regions are displayed as completely transparent.

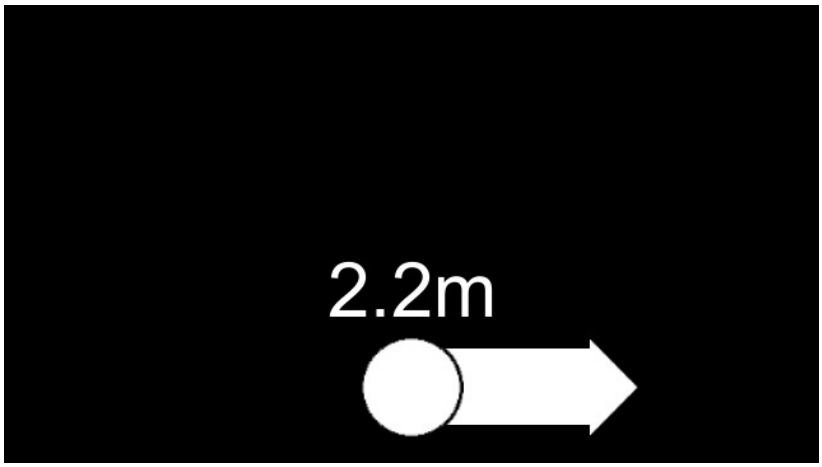


Figure 5.2b: Arrow compass visual cue as displayed on the Google Glass. Black regions are displayed as completely transparent.

Apparatus

The hardware setup for the experiment consisted of a central desktop computer, a Nexus 5 handheld smartphone, a Google Glass: Explorer Edition, six Optitrack Flex 13 cameras and an Aftershokz Sportz3 bone conduction headset connected to the desktop through a Wi AudioStream Pro AV wireless transmission device. The desktop computer housed an Intel Core i7-2600 @3.4Ghz CPU with 8gb of RAM and ran a 64-bit Windows operating system. During the experiment, the desktop computer ran two programs; the Motive motion capture software, and a custom Unity program which managed the core processes of the experiment. The Unity program controlled the virtual target locations, cue order randomization, sound cues, and passed calculated angle and distance data through to the smartphone via wireless connection. The Unity program also allowed input to move between trials, prime trials to start when a participant entered the testing region, and stop the trial once a target card had been flipped over. The Nexus 5 smartphone acted as a wireless hotspot and bridge between the desktop computer and the Google Glass, passing on incoming data to the Glass via a Bluetooth connection. The smartphone app also had buttons to allow a second experimenter to record the number of incorrect card flips. Visual and audio cues were presented to the participants via the Google Glass and bone conduction headset respectively. To monitor movement within the

environment, the participant was required to wear a cap with eight reflective motion capture beads sewn on top. The six Optitrack cameras were placed in an 180^{o} arc around the environment and calibrated with the Motive software to detect the position and orientation of the marked cap. The position and orientation data were updated and recorded in a log file at a refresh rate of 60Hz.

Procedure

After introductions, participants were given a description of the task including an estimate of the total time and instructions to prioritize speed over accuracy during the task. Consent forms were collected and the participant was given the Google Glass, bone conduction headset and marked cap to wear. The experiment proper consisted of 36 trials separated into 4 blocks of 9. During each block, the participant would encounter each pair of possible visual (no cue; map cue; arrow cue) and audio cue type (no cue; static cue; dynamic cue) once. The order of the cue pairs was randomised in each block. The first of the four blocks was designated a practice block and the participants were encouraged to ask any questions and discuss any issues with the task during this time. Participants were given a second chance to ask questions at the end of this block before experiment condition trials began.

After the practice block, participants completed three additional blocks of the same structure for a total of 27 test trials. To remove the random effect of the distance to the card from the starting point on response times between participants, each participant experienced the same sequence of target locations. The sequence used was randomly generated with an equal likelihood of the target being placed at any one of the 67 possible locations. The order of cue types was randomly assigned for each participant individually to remove order effects and to prevent dependence between the cue type and distance to target effects.

For each individual trial, the participant was asked to leave the room so that the room could be reset target card could be moved to a new location. Once the card was placed and both experimenters had returned to their observation locations, the participant was called in. The trial

timer began as soon as the marked cap was detected by the cameras. At this time, the visual cue appeared on the Glass and the audio cue began to play. Participants where then given up to 40 seconds to move around the environment and search for the target card. Each incorrect flip was recorded by the experimenter operating the smartphone. Once the participant found and flipped the target card, or the trial timer reach 40 seconds, the experimenter operating the desktop computer would mark the trial as finished and then ask the participant to leave the room for the next trial.

Experiment 1 Results

We first performed a preliminary ANOVA for the table on which the target was placed, current participant and the number of times a particular cue had been presented predicting response time. When not accounting for the cue effects, there were no significant differences in the average search times between participants ($F(34,474)=0.90$, $p=.630$, $\eta^2=0.06$). Performance over time was stable ($F(7,474)=1.12$, $p=.346$, $\eta^2=0.02$) with no evidence of a linear ($p=.168$) or quadratic ($p=.278$) increase or decrease in performance as participants progress though the blocks of trials. An effect of distance to the target remained, with the central table and the far tables taking more time to reach and the table directly in front of the entryway taking less time to reach ($F(5,475)=6.66$, $p<.001$, $\eta^2=0.07$). The effect of the target's table did not differ significantly between participants ($F(175,485)=0.86$, $p=.877$, $\eta^2=0.24$) or across blocks ($F(14,474)=0.99$, $p=.465$, $\eta^2=0.03$).

To remove the effect of the target location's table on performance for the analysis of the effect of the cues, the average to find a target on each trial's table was subtracted from the trial response time. This means the following analyses use and report the adjustment caused by each cue from the average time for the current table.

We next performed a repeated measures ANOVA of visual and audio cues types predicting average search time adjustment from table average for each participant. A Mauchly's Test of Sphericity found significant deviations from homogenous difference variances between visual cue conditions ($W=.655$, $\chi^2(2)=13.96$, $p=.001$) and the interaction between visual and audio cue conditions

(W=.318, $\chi^2(9)=37.13$, p<.001). To account for this a Greenhouse-Geisser correction is applied to the following results.

The presentation of visual cues significantly decreased search times when present ($F(1.49,50.56)=348.34$, p<.001, $\eta^2=0.911$). In particular, for no-audio cue trials both the arrow cue ($t(34)=12.75$, p<.001, Cohen's d=2.15) and map cue ($t(34)=23.56$, p<.001, Cohen's d=3.98) produced significantly shorter search times than the no cue control condition. Additionally, between the two visual cue types the map cue led participants to the targets significantly faster than the arrow cue in the absence of audial cues ($t(34)=10.32$, p<.001, Cohen's d=1.75).

There are significant differences in search times between audio cue conditions ($F(1.94,66.01)=38.05$, p<.001, $\eta^2=0.528$) and a significant interaction between visual and audio cues ($F(2.85,96.80)=35.06$, p<.001, $\eta^2=0.508$). These effects are shown on figure 5.3. Both the audio cue main effects and the audio/visual cue interaction result from the decrease in search time associated with the static audio cue when no visual cue is present ($F(1,34)=35.06$, p<.001, $\eta^2=0.763$). When no-visual-cue conditions are excluded, the effect of audio cues ($F(1.98,67.19)=0.64$, p=.530, $\eta^2=0.018$) and the interactions between audio and visual cues ($F(1.90, 64.75)=0.44$, p=.637, $\eta^2=0.013$) on search speed disappear.

Table 5.1 displays the number of incorrect flips made on each trial averaged across participants. The difference in variance between conditions is too extreme to support ANOVA analyses so instead comparisons between condition's incorrect flip rates were made using paired sample t-test, adjusted for the family-wise error rate using the Sidak correction. The corrected significance threshold for the comparisons is p=.0014.

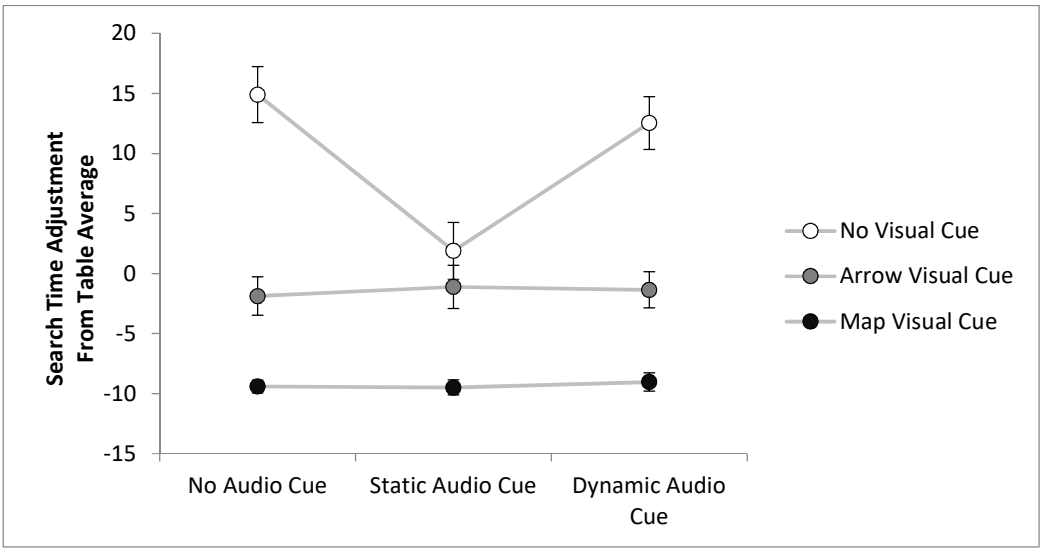


Figure 5.3: Search time adjustments from table average for each combination of visual and audial cues. Error bars represent 95% confidence limits.

During the no-audio/no-visual cue condition, the participants made more incorrect flips than in any other condition except for the dynamic cue/no-visual cue condition ($t(35)=1.88$, $p=.069$, Cohen's $d=0.31$). The comparisons between the number of incorrect flips for the no-audio/no-visual condition and the remaining 7 conditions were each significant ($p<.001$). There were no differences between map cue conditions' effect on incorrect across audio cues nor between arrow cue conditions across audio. Each of these comparisons were not significant ($p>.05$). The map cue alone resulted in fewer incorrect flips than the arrow cue alone ($t(35)=5.20$, $p<.001$, Cohen's $d=0.87$). The static audio cue alone and arrow visual cue alone had similar mean numbers of incorrect flips ($t(35)=3.16$, $p=.003$, Cohen's $d=0.53$).

Table 5.1:

Average number of incorrect card flips for each combination of visual and audio cues.

	No Visual Cue	**Map Cue**	**Arrow Cue**
No Audio Cue	15.9	0.3	1.8
Static Cue	3.6	0.2	2.4
Dynamic Cue	13.3	0.2	1.9

Experiment 1 Discussion

In accordance with the previous findings for depth-irrelevant search, a map based endogenous cue guided participants to disguised targets in a depth-relevant environment search more efficiently than an endogenous arrow based cue (Chapter 4). We hypothesized that the working memory costs of physical movement (Epling et al., 2016) may have decreased the effectiveness of cues which display all of their information at once as the map cue does. This does not appear to be the case. Observationally, participants showed little to no hesitation between reaching their target table and making their initial card flip. Sometimes this flip would be incorrect and at these times participants would look up at the device to confirm the target's location. However, this pattern of behaviour is best explained by the participant making a mistake in their initial interpretation of the map rather than the participant forgetting their target in transit. The lack of interference in this regard is likely because the memory costs of travelling involve planning the route (Oulasvirta, et al., 2005). Route planning requires storing the destination and thus this could reinforce, rather than clash, with the need to remember the target's location.

The map and arrow visual cues encode the same information: both cues gave horizontal rotation and depth information for the targets location; both cues lead to a unique position in space and this position was always the target; both cues continued to be accurate throughout the experiment and gave opportunities for participants to reassess their search pattern if they failed to find their target following an initial attempt; and neither cue encodes vertical displacement to the target as this was not needed for uniquely identifying the target position. Despite receiving identical information from both visual cues, the map cue led the participant to the target in 7.34 seconds on average while the arrow cue took 14.98 seconds on average.

We attribute this large proportionate increase in search time to the difference in the amount of inference and estimation the participant must make in interpreting the cues. The map based cue

gives the user a defined and discrete end point for their travel, while the arrow cue requires participants verify their bearing and estimate a distance for deciding the end point of any planned travel route. The need to estimate an end point increases the likelihood that participants undershoot or overshoot the target. Observationally, an initial overshoot of the target was present on the majority of the arrow cue trails regardless of supporting audio cue. This uncertainty regarding the end point of travel displaces the participant away from the target, requires at least one additional attention shift to correct the participant's course and changes how participants conduct their search of the final location. During the final search procedure for arrow cue trials, the majority of participants focussed on the device and made sure they had extracted all possible information from the device before attending fully to the real world. This occurred even when the arrow cue led the participant to the target's location with clear enough accuracy to enable a manual, unguided search to outpace using the Glass to precisely locate the target's position. Participants were instructed twice that the number of cards flipped was irrelevant and there was no penalty for doing so if it sped up their search. Despite this, participants over-relied on their device to the detriment of their performance. Each of these three features of the participant's search behaviour slows their performance compared with the map cue.

The visual cues used in the experiment dominated the effects of any audio cues present. When a visual cue was present, there was no evidence of any difference caused by the addition of an audio cue. This indicates that audio cues were ignored when participants had access to a visual cue. Multiple Resource Theory (Wickens, 1980) predicts that audio cues should interfere less with a visual task like navigating obstacles than a visual cue. Contrary to this, the audio cues performed poorly compared to the visual cues. Again, the differences in performance appear to be a function of the amount of interpretation and estimation a cue requires from the participant. Much like the visual cues, the static sound cue led a participant to a unique location in the search space and this was always the location of the target. Judging depth from the sound cues was a more difficult interpretation than using a known scale such as numerical distance in meters. Participants needed to compare the current volume of the audio cue against their memory of previous trials and this estimation is less precise the distance

estimates provided by the visual cues. This meant participants made more and larger overshoots and undershoots of the target's location and any corrections were delayed by the need for the participant to detect the sounds had started to weaken rather than strengthen when they passed the correct search zone.

An advantage of the static audio cue over the arrow visual cue was that it encouraged a more successful local search strategy once the target was near. In arrow cue conditions, participants would over rely on the device and continue to attend to it as the precisely located the position of the target even when it would have been faster to flip all of the possible candidate cards in the estimated search zone. Because of the difficulty in precisely identifying the point of maximum volume, participants did not frequently attempt this in static audio cue, no visual cue trials. This led participants in these trials to adopt a strategy where they flipped more a greater number of cards which finds the target optimally.

Overall, this study found that several types of virtual information aids can assist users in guided search tasks, and of the types tested, map based visual cues were the best method for guiding users to a location. What the experiment does not do is compare the map cue to the accepted optimal method of directing attention, an environmentally positioned exogenous cue (Müller & Findlay, 1988; Müller & Rabbitt, 1989). Comparing the device positioned endogenous map cue against an environmentally positioned exogenous cue will allow an estimation of how closely to optimally the HMD-based cue guides attention. To achieve this, the previous experiment was rerun with the audio cue conditions removed and new visual and environment cues added. A no cue condition, the map cue and the arrow cue were retained from the previous experiment and added is a high salience, exogenous environmental cue in the form of a bright, blinking, red light which was placed above the target card. Our hypothesis is that the map cue will lead participants to the target at a similar speed to the exogenous flashing light cue.

Also added were new conditions which are designed to separate the issues that affected the arrow cues in this experiment. When following the arrow cues, participants could end up slightly

displaced from the target when their destination estimate was not accurate. To test the effect of destination uncertainty on search times a second map cue was added. This cue marked a region which contained the target rather than a single accurate point and it was randomly displaced around the target's location. The circle was sized to cover approximately 5 cards in any given position, meaning a participant must flip 2 incorrect cards on average to find the target. This matches the average number of incorrect flips made on arrow cues in the current experiment. Unlike the arrow cue in the current experiment, the ring map cue should lead participants to adopt the optimal local search strategy of rapidly flipping all of the cards in the expected search zone as there is no other method of narrowing down the target options at this point. Our hypothesis is the ring map cue will lead to only slightly slower search times than the point map cue, and that this cue will lead participants to the target faster than the arrow based cue.

The final new addition is a second environmental target cue which was presented alone and in tandem with each of the HMD-delivered cues. This second environmental cue was a target card with a 10% reduced width. It was designed to function as a non-pop out cue (Treisman, 1985; Treisman & Gormican, 1988), which in contrast to the flashing red light cue, would be invisible at a distance but easily detectable up close. This provides a closer representation to the real-world task of searching for keys, where the keys are hidden but not completely obscured by the environment. This will test how resilient the over-reliance on the device behaviour caused by the arrow cue is by offering an alternative local search strategy when the participant has reached the search location. Because this affects the search strategy once movement is complete, our hypothesis is that it will improve the search speeds of trials where participants make inefficient local searches, such as the no cue and arrow cue conditions but it will have no effect on conditions where participants already make an optimal or close to optimal local search such as the map conditions.

Experiment 2 Method

Participants

Thirteen Male and 7 Female volunteers recruited from the University of Canterbury participated in the experiment. The participants had an average age of 25.9 years, and ranging from 19-34 years. Nine of the participants participated in the previous experiment. We do not expect any differences in their performance as there was no evidence of practice effects across blocks in the previous experiment. Their vision was normal or corrected to normal and all participants reported normal hearing. Participants were compensated for their time with a $10 shopping voucher.

Environment and Stimuli

The testing region was a 4.2m by 6.2m rectangle. In this space were positioned 6 tables of various sizes covered with between 4 and 15 laminated cards for a total of 67 cards. Figure 5.1 shows the layout of the tables within the space. Each card was 12.7cm by 7.6cm and blank on both sides excluding the pair of target cards which each had a 6cm diameter green circle on a single face. One of the two target cards were reduced in width by 10% to 11.4cm. A blinking, red, 3 LED array was used for creating an exogenous environmental cue.

Visual cues for the experiment consisted of a map with the target marked as a red dot (Figure 5.2a), a map with a red ring containing the target (Figure 5.2c) and an arrow compass (Figure5.2b). The map cue represented the six tables as white rectangles on a transparent background and marked the position of the target card with a red dot. The arrow compass cue had an arrow that extended out to the left or right based on how much the participant needed to rotate to be looking towards the target card's location. Above the central dot of the compass was written the distance in meters from the participant to target, ignoring vertical displacement.

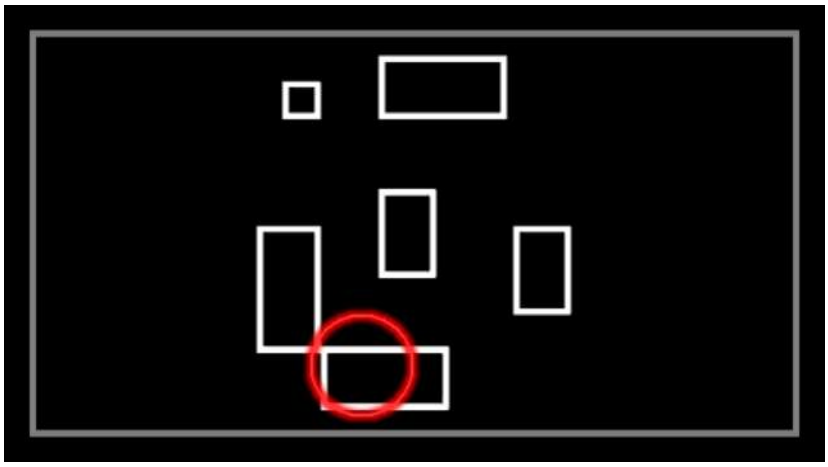


Figure 5.2c: Ring map visual cue as displayed on the Google Glass. Black regions are displayed as completely transparent.

Apparatus

The hardware setup for the experiment consisted of a central desktop computer, a Nexus 5 handheld smartphone, a Google Glass Explorer Edition, six Optitrack Flex 13 cameras and a 5 red LED array bike back light. The desktop computer housed an Intel Core i7-2600 @3.4Ghz CPU with 8gb of RAM and ran a 64-bit Windows operating system. During the experiment, the desktop computer ran two programs; the Motive motion capture software, and a custom Unity program which managed the core processes of the experiment. The Unity program controlled the virtual target locations, cue order randomization, and passed calculated angle and distance data through to the smartphone via wireless connection. The Unity program also allowed input to move between trials, prime trials to start when a participant entered the testing region, and stop the trial once a target card had been flipped over. The Nexus 5 smartphone acted as a wireless hotspot and bridge between the desktop computer and the Google Glass, passing on incoming data to the Glass via a Bluetooth connection. The smartphone app also had buttons to allow a second experimenter to record the number of incorrect card flips. Visual cues were presented to the participants via the Google Glass. To monitor movement within the environment, the participant was required to wear a cap with eight reflective motion capture beads sown on top. The six Optitrack cameras were placed in an 180^{o} arc around the environment and

calibrated with the Motive software to detect the position and orientation of the marked cap. The position and orientation data were updated and recorded in a log file at a refresh rate of 60Hz.

Procedure

After introductions, participants were given a description of the task including an estimate of the total time and instructions to prioritize speed over accuracy during the task. Participants were also asked to look at a table with 9 cards, including the shortened target card, to ensure they were able to detect the difference in size between cards. All 20 participants succeeded at selecting the smaller card on their first attempt. Consent forms were collected and the participant was given the Google Glass, bone conduction headset and marked cap to wear. The experiment proper consisted of 36 trials separated into 4 blocks of 9. During each block, the participant would encounter each pair of possible visual (no cue; dot map cue; ring map cue; arrow cue) and target card size (11.4; 12.7cm) once as well as a single trial in which the red LED light array would be placed behind the regular-sized target card and pointed towards the entry point. The order of the cue pairs was randomized in each block. The first of the four blocks was designated a practice block and the participants were encouraged to ask any questions and discuss any issues with the task during this time. Participants were given a second chance to ask questions at the end of this block before experiment condition trials began.

After the practice block, participants completed three additional blocks of the same structure for a total of 27 test trials. To remove the random effect of the distance to the card from the starting point on response times between participants, each participant experienced the same sequence of target locations. The sequence used was randomly generated with an equal likelihood of the target being placed at any one of the 67 possible locations. The order of cue types was randomly assigned for each participant individually to remove order effects and to prevent dependence between the cue type and distance to target effects.

For each individual trial, the participant was asked to leave the room so that the room could be reset and the target card moved to a new location. Once the card was placed and both

experimenters had returned to their observation locations, the participant returned. The trial timer began as soon as the marked cap was detected by the cameras. At this time the visual cue appeared on the Glass. Participants were then given up to 40 seconds to move around the environment and search for the target card. Each incorrect flip was recorded by the experimenter operating the smartphone. Once the participant found and flipped the target card, or the trial timer reach 40 seconds, the experimenter operating the desktop computer would mark the trial as finished and then ask the participant to leave the room for the next trial.

Experiment 2 Results

We first performed a preliminary ANOVA for the table on which the target was placed, current participant and the block number predicting response time. When not accounting for the cue effects, there were no significant differences in the average search times between participants ($F(19,371)=0.97$, $p=.503$, $\eta^2=0.05$). Performance over time was stable ($F(2,371)=44$, $p=.642$, $\eta^2=0.00$) with no evidence of a linear ($p=.474$) or quadratic ($p=.992$) increase or decrease in performance as participants progressed though the blocks of trials. In contrast with the previous experiment, there was no evidence of a difference in the time taken to find targets across tables ($F(5,371)=1.14$, $p=.340$, $\eta^2=0.02$). In the previous experiment participants made successful searches on the table directly in front of the entryway more quickly than on the central table and the far tables. For consistency and to allow comparisons between the two studies, the following analyses use and report the adjustment caused by each cue from the average time for the current table.

We next performed a repeated measures ANOVA of the 4 visual cue types and the 2 target card widths predicting average search time adjustment from table average for each participant. A Mauchly's Test of Sphericity found significant deviations from homogenous difference variances between visual cue conditions ($W=.277$, $\chi^2(5)=22.76$, $p<.001$) and the interaction between visual cue and target width conditions ($W=.430$, $\chi^2(5)=14.95$, $p=.011$). To account for this a Greenhouse-Geisser correction is applied to the following results.

The presentation of visual cues significantly decreased search times when present ($F(2.05,38.94)=270.15$, $p<.001$, $\eta^2=0.93$). Planned within-subjects Helmert contrasts were then used to isolated these differences. All three cues types led participants to the target faster than no cue at all ($F(1,19)=541.96$, $p<.001$, $\eta^2=0.97$). The two map cues led participants to the targets faster than the arrow cue ($F(1,19)=43.95$, $p<.001$, $\eta^2=0.70$). The dot map cue yielded faster search times than the ring map cue ($F(1,19)=37.17$, $p<.001$, $\eta^2=0.66$).

There are significant differences in search times between target width conditions ($F(1,19)=17.77$, $p<.001$, $\eta^2=0.48$) and a significant interaction between visual cues and target width ($F(2.24,42.46)=19.48$, $p<.001$, $\eta^2=0.51$). These effects are shown on Figure 5.4. The cause of these effects is the difference in search time between the no cue, regular target and no cue, small target conditions ($F(1,19)=35.45$, $p<.001$, $\eta^2=0.65$). When the no cue conditions are excluded the differences across target sizes ($F(1,19)=0.18$, $p=.680$, $\eta^2=0.01$) and the interaction with cue type ($F(1.41,26.80)=0.29$, $p=.931$, $\eta^2=0.00$) disappears. Two participants reported having difficulty detecting the shortened card during the experiment.

Trials in which the LED lights were placed behind the target and the participants made one or more false flips were excluded from this analysis. In these trials the pop-out visual cue had clearly been missed by the participant. This exclusion is justified because the purpose of this condition is to analyse the effects of detecting the environmental cue, not test its saliency. 6 out 60 light cue trials were excluded this way.

Repeated measures ANOVA difference contrasts were used to compare the LED light cued condition against the other regular-sized target conditions. The LED light cue led participants to the target faster than the no cue ($F(1,19)=506.12$, $p<.001$, $\eta^2=0.96$), arrow cue ($F(1,19)=32.83$, $p<.001$, $\eta^2=0.63$) and ring map ($F(1,19)=14.35$, $p=.001$, $\eta^2=0.43$) conditions. There were no differences in search times between the dot map condition trials and LED light condition trials when failures to detect the LED array were excluded ($F(1,19)=0.53$, $p=.482$, $\eta^2=0.03$).

The participants who had participated in the previous experiment did not perform differently to the new participants ($F(8,144)=0.82$, $p=.521$, $\eta^2=0.04$).

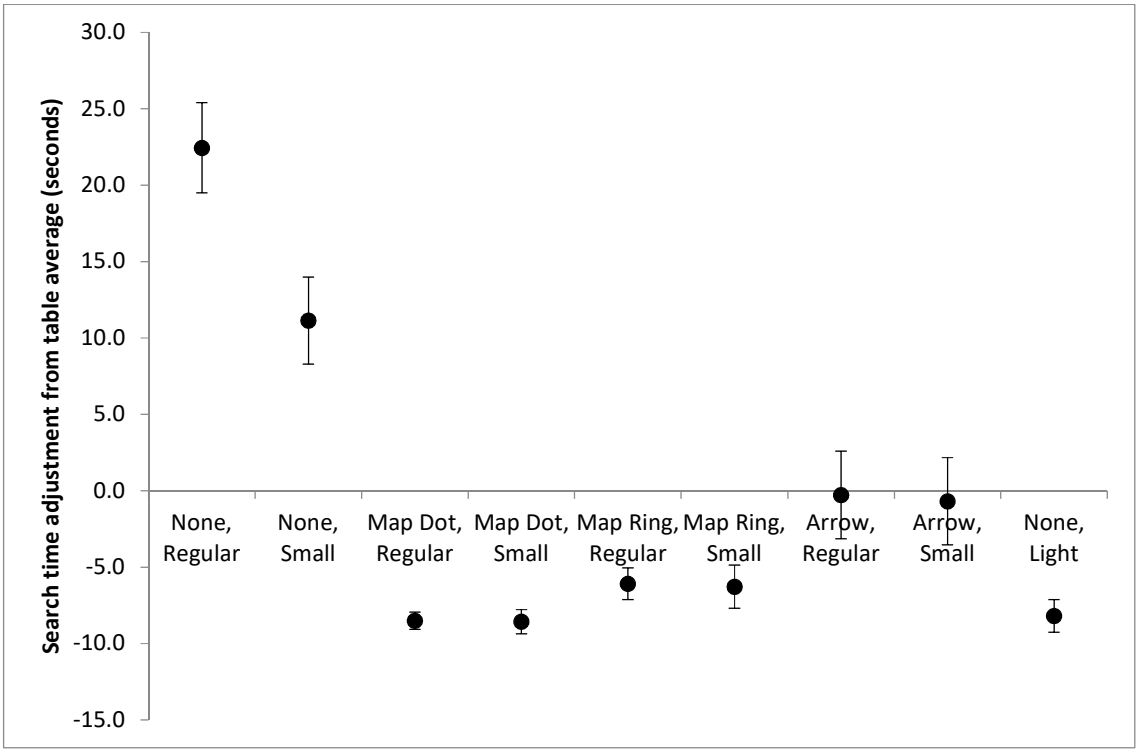


Figure 5.4: Search time adjustments from table average for each combination of Visual cue and target width. Error bars represent 95% confidence limits.

Table 5.2 displays the number of incorrect flips made on each trial averaged across participants. The difference in variance between conditions is too extreme to support ANOVA analyses so instead comparisons between condition's incorrect flip rates were made using paired sample t-test, adjusted for the family-wise error rate using the Sidak correction. The corrected significance threshold for the comparisons is p=.0018.

The smaller target card reduced the average number of flipped cards in the no cue conditions but this difference was not significant at the adjusted level ($t(19)=2.77$, $p=.012$, Cohen's $d=0.62$). The other comparison t-tests for the cued conditions were not significant ($p>.05$), and so in the cued

conditions the smaller card had no effect on the average number of flipped cards. The dot map cue caused participants to flip fewer cards than both the ring map ($t(19)=4.24$, $p<.001$, Cohen's $d=0.95$) and arrow ($t(19)=4.00$, $p=.001$, Cohen's $d=0.89$) cued conditions. There were no significant differences between the number of cards flipped between the ring map and arrow-cued conditions ($t(19)=1.81$, $p=.086$, Cohen's $d=0.41$).

Table 5.2:

Average number of incorrect card flips for each combination of visual and card widths.

	No Visual Cue	**Dot Map Cue**	**Ring Map Cue**	**Arrow Cue**
Normal Width	15.6	0.2	1.2	1.9
Shortened	10.8	0.1	1.2	1.9

Discussion

In agreement with the primary hypothesis of the study and the results of the first experiment, the map based visual cue lead participants to hidden targets within a 3D environment more quickly than the arrow based cue. Participants in this experiment were slightly faster than in the previous experiment, taking 6.21 seconds on average to find a target with the point map cue, 8.46 seconds with the ring map cue, and 14.29 seconds with the arrow cue.

The non-popout environment cue, the shortened card, had no effect on search times when a device based cue was present. We attribute the failure of the non-popout cue to its unreliability. Participants were unaware of which combination of device cue and environment cue they would receive on each trial and so any attempt to use the shortened card to detect the target locally would only be possible 44% of the time. When the shortened card was absent, this same local search behaviour would instead slow responding by requiring the participant to inspect each possible card within the local search area before reverting to their strategy for finding hidden targets. This makes searching for the shortened card an inefficient strategy to adopt. It took a longer time on average for the participants to detect the card than it took to flip all the likely cards within the local search area.

It was common for a participant to notice that the card was shortened only once it was already in their hand.

Adding a level of uncertainty to the map cue by replacing the precise point map cue with the randomly offset ring map cue increased search times, but still led the participants to the target significantly faster than the arrow cue. This shows that the increased search times when using the arrow cue is not solely due to the cue leading the participants to a search zone slightly displaced from the target. Instead participants adopted a less efficient search strategy at their destination when they were guided by the arrow cues. We predicted that the addition of an environmental marker would encourage a more efficient search strategy but this was not the case when the unreliability of the non-popout, environmental cue resulted in it having low utility. It is possible a more reliable environmental cue would still improve arrow based cues.

Alternately, two other factors may have led participants to be overly dependent on their device. First, the novelty factor of using the Google Glass may have motivated participants to attempt to use it as much as possible, even to the detriment of their own performance. Second, the concern over making a mistake by flipping an incorrect card, despite participants being reminded multiple times that card flips were not penalized and they are scored on their search speed, may have caused participants not to take the risk of flipping a card when there was the opportunity for new information. For the map cue, the information displayed was static during the trial and so it would not have prompted the participant to attempt to extract more precise information from the device. The arrow cue, on the other hand, updated with every movement by the participant, providing continuous such opportunities. It is likely a combination of these factors impaired performance in arrow cue conditions.

Both of these effects would be reduced over time as users became more familiar with the device if it was implemented in an applied setting. The novelty of the device would quickly fade with repeated use and aversion to making incorrect flips would extinguish over time with a proper training, feedback and reward structure. The effective local search strategy of rapidly flipping multiple likely cards can be encouraged by disabling constant updates when the user is within close proximity to the

target. A message displayed on the device at this time could remind the participant that false flips are not penalized, although doing so may unduly capture the participants' attention even after the instruction has been clearly understood as new information appearing on the Glass draws attention quickly and reliably (Chapter 2). The appearing message functions as an exogenous cue, a sudden shift in luminance within the participant's peripheral vision (Egeth & Yantis, 1997; Yantis & Jonides, 1990).

To accurately measure how quickly a participant was led through the environment by a salient exogenous cue, any flashing light cue trials in which a participant made an incorrect flip were discarded. Because the identity of the target card was not hidden in these trials, any incorrect flips meant a participant had failed to notice the flashing light. This occurred on 6 (10%) of the flashing light trials. The purpose of this research was to extend replacements for exogenous attention guiding cues into 3D space because exogenous cues only function when the program designer has control of the environment which will not be the case for augmented reality programs for current generation HMDs. Instead the designers will only have access to audio channels and only a restricted range of the user's vision which moves with their head rotation. We found no evidence of a difference in search speeds when participants were guided by endogenous map cues on the device compared with when they were guided by an exogenous flashing light within the environment. There were no trials in which the exogenous flashing light cue was presented in tandem with a device-centered endogenous cue, but previous research into the capture of attention by exogenous cues during a period in which a participant's attention is already being directed by an endogenous cue found that the endogenous cue does not stop the automatic redirection of attention caused by exogenous cues (Seiss, Kiss, & Eimer, 2009). Therefore, in cases where both cue types are present, participants can be expected to respond to the exogenous cue, and at the same speed as the exogenous cue alone.

The equivalence in search speed between map and flashing light cue trials occurred despite the map cue requiring interpretation by the participant and remapping from a top down perspective onto the environment. Because the Glass gave guiding information on a proportionately larger number of trials than lights within the environment, it is likely that participant made their initial

attention shift to and from the Glass even in trails when the flashing light was present. This would have slightly slowed responding to the exogenously guided trials and reduced any advantage gained because the map cues require interpretation. Regardless, the top down map cues were a successful replacement for environmentally centered cues that are deliverable via Google Glass and other HMDs when a program needs to guide a user to a hidden target within a 3D environment. The close performance between the two styles indicate there is little need for further improvement on attention directing cues for HMD design. The relative success of the ring map cue also shows that top down maps are an effective guidance tool even when precise information cannot be obtained by the program. Altogether, the recommendation from this work is that top down simplified map cues be used when directing a user's attention in 3D for augmented reality programs delivered via current generation HMDs.

Chapter 6: Conclusions

Summary

The purpose of this PhD thesis was to explore how programmers and software developers can use head mounted devices to aid performance rather than have the device act as a distractor. To use a HMD to augment performance, a user needs to attend to both their device and their environment at the same time. Dividing attention between the device and their primary task creates a performance cost to the user which the value of the information given by the augmentation needs to exceed. Improving an augmenting program can be done in two ways: the costs of dividing attention can be reduced, or the value of the delivered information can be increased.

Improving the value of the device-delivered information is task specific. For example, a more accurate map will improve a city navigation assistance program but does nothing to improve a translation application. The cognitive costs of divided attention are also task specific. Instead of increasing the maps accuracy, performance could be improved by removing irrelevant information from the map, decreasing the time and mental demand taken to process the information. Methods improving the value of information and decreasing the cognitive costs are also typically medium independent. A more accurate map will improve performance regardless of which device the map is displayed on. A common set of design practices to improve information quality and remove unnecessary complexity would be useful for HMD developers, but because these advantages are common across devices they already have access to existing visual design guides such as Mullet and Sano's '*Designing visual interfaces: Communication oriented techniques*' (1994).

The perceptual costs of dividing attention between a HMD and the environment are device rather than task dependent. Regardless of what is displayed on an HMD's screen, the user will need to move their visual attention from the environment to the device and back again to access the information. Methods to decrease the time cost to access the device would have broad applications for HMD developers and programmers. HMD's are the first-time transparent heads up displays

produced for widespread commercial use by untrained consumers. Overlaying information on top of an environment with a head up display produces different performance than the heads down display of a smartphone, tablet or other system (Ablaβmeier, et. al., 2007; Liu & Wen, 2004). This means developing tools specifically for mitigating the perceptual costs of dividing attention between HMD and the environment has valuable, practical application.

Detailed in chapter 2, the first two experiments tested whether information events displayed on the Google Glass interfered with a far-domain task which required constant visual attention. The primary finding of these two experiments was that moving attention to attend to the worn device did cause participants to react more slowly to a simultaneous event requiring action in the primary task. This delayed reaction was not present when the same secondary word-memory task was displayed on the primary task's screen rather than on the HMD. This means the delay was caused by the perceptual cost of switching between the tracking task screen and the HMD's virtual display rather than the cognitive costs of reading and memorising the displayed word.

Two additional findings were provided by these experiments. First, it was extremely rare for a word presentation event of the Glass to be missed by a participant. This occurred regardless of whether their full attention was on the Glass or dividing their attention between the Glass and the tracking task. Second, when the inwards and outwards shifts of attention were separated by warning a participant of an impending word presentation, the outwards shift of attention from HMD to tracking task took approximately twice as long as the inwards shift. Together these finding support an asymmetry in how attention is divided in depth in agreement with Andersen & Kramer (1993), Finlayson & Grove, (2014), and Parks & Corballis (2006). The reliability and speed of the inward attention shift means there is likely little room for improvement when pulling a user's attention in towards the HMD's display. Pushing attention back out to the environment is more difficult and therefore is a more suitable target for improvement.

One method to increase the speed of returning a user's attention from a wearable device to the environment is to use a cue which can both immediately start a movement in the user's attention

while directing towards the region in which their attention is needed. Visual cues which elicit reflexive and immediate responses are classed as exogenous cues (Posner, 1980). This class of cues direct attention by creating a sudden and salient change within the environment at the position of interest. This taps into the Orientation Reflex and creates a fast and reliable redirection of attention. This form of cue is not an appropriate tool for HMD developers. Current generation HMDs are monocular and had a limited ability to display objects in depth. The current HMD displays also cover only a small portion of the user's visual field. This means the programmer does not have the control of the user's visual space to create an environmentally positioned cue unless the device is already directed at the target. There will also be situations in which a user needs to be directed to areas of their environment outside their peripheral vision and a centralized cue will be necessary.

The class condition for exogenous cues is for the cue to elicit an automatic, reflexive response (Jonides, 1981; Müller & Rabbitt, 1989; Posner, 1980). The limited display region of the device and possibility of the requirement for attention to be directed outside the user's visual space means that developers need a centrally positioned cue. Of the involuntary reflex responses made by the human eye, one candidate which starts from a central location is the Pursuit Motion Reflex (Lisberger, 2010). When an object abruptly begins smooth motion inside the field of vision, the eye takes within 150ms to detect and match the object's momentum (Lisberger, 2010; Rashbass, 1961; Wilmer & Nakayama, 2007). We predicted that a visual cue which utilized the Pursuit Motion Reflex would create responses similar to those of the cues which tapped the Orientation Reflex.

Chapter 3 details an experiment which tests the viability of a pursuit motion cue. It was compared with an archetypal version of the second class of cues, endogenous cues. Endogenous cues require interpretation and action by the participant which causes slower responding (Posner, 1980). In this experiment, the endogenous cue was an arrow. In contradiction of the experiment hypothesis, the pursuit motion cue failed to perform as well as was expected for an exogenous cue and was also outperformed by the arrow cue. There were design features which could have improved the pursuit motion cue but the unexpected level of success of the arrow cue made any further investigations

unattractive. Contrasting experiments which have found larger differences in speed between endogenous and exogenous cues (Müller & Findlay, 1988; Müller & Rabbitt, 1989), the arrow cue started participants in the correct direction of the target only 40ms slower than the exogenous cue. This difference was significant but small in magnitude. This means a small improvement in the effectiveness of endogenous cues could allow them to match the speed of redirection of typical exogenous cues while also being centrally positioned.

Chapter 4 details an experiment in which various endogenous visual cues were tested alongside an alternative solution, cues using features that can be processed with peripheral vision alone. The hypothesis was that the peripheral visual cues would add information to the user's visual space without interrupting their natural and highly practiced, normal search behaviours. Peripheral cues were unsuccessful. They drew attention completely to the device and encouraged an inefficient search strategy.

An arrow pointing from the centre of the space out to the position of the target was used in chapter 3 and was repeated in this experiment. It was outperformed by two other endogenous cues. The second most effective cue was a dynamic arrow which encoded horizontal direction and the distance to the target without encoding vertical direction. The low value of vertical direction was due to targets appearing in a wider horizontal than vertical range. This is a realistic situation, as when a user is moving through an environment, there is a 360^{o} range of horizontal rotation, but only a 180^{o} vertical range. Regularly used objects are usually kept in an even more restricted vertical range. The two-component arrow was more effective than an arrow which encoded all three of the values. This shows that when conducting a visual search, it is sufficient to lead users to the region of the target, overcoming the limitation of the user having no initial information. This is because natural search behaviour is quite efficient over a small region. It is not necessary to completely replace the user's behaviour when creating an augmentation program.

The most successful cue of chapter 4 reinforces this concept. The simplified map cue gave the least precise direction to the target of all the cues, displaying a $1/12^{th}$ section of the full region which

contained the target. A clear end point for their initial head rotation allowed users to travel quickly to the appropriate section of the environment to search as shown by the relatively small time increase caused by initial offset to the target. Once there, the participant had no further clues about the target location, forcing them to use their natural search behaviour which was faster than navigating by the Glass alone.

The map cue gave a discrete destination for movement of attention, and the less successful horizontal direction/distance arrow gave the second most clear distance estimate of the all the cues. The remaining cues require a large amount of interpretation to estimate an endpoint for the motion needed to move attention. This leads to an untested hypothesis that it is the clear endpoint of regular exogenous cues, rather than their reflexive nature, that causes their fast redirection of attention. This matches with the findings of experiment 3 (chapter 3) where the pursuit motion cue required participants to make a judgement regarding the pursuit object's travel angle to determine its end point. In contrast, the arrow cue had only 4, easy to distinguish configurations, and therefore led to a unique end point each time without much interpretation required by the participant.

The final two experiments, detailed in chapter 5, expanded the search task into a more realistic setting. Participants were required to move through a real-world space and navigate obstacles while both searching for targets and attending to the HMD. This task required participants to be able to determine from a cue the target's depth within the environment as well as its bearing. It also created a possibility for the targets to appear outside the user's field of view, meaning the cues needed to cover the full environment.

For this purpose, the two most successful cues from the previous chapter were adapted to account for target depth. Both the extended map and arrow cues led participants through the environment to the target faster than when following an audio cue and faster than searching at random. The visual cues dominated the audio cues when combined. When a visual cue was present, audio cues had no effect on search speed showing that the visual cues alone were sufficient assistance. Matching the results of the 4th experiment (chapter 4), the map cue outperformed the arrow cue. The

size of this discrepancy was larger than in experiment 4 due to the more complex search procedure and the consequent longer average trial time.

Conclusions

The purpose of this PhD thesis was to explore how programmers and software developers can use head mounted devices to aid performance rather than have the device act as a distractor. A common limitation to performance across HMD program designs is the need to swap attention to and from the device. The experiments in chapter 2 showed that the shift from device to environment is more difficult and takes a longer time than the reverse. One way to support a fast and reliable transition of attention back to the environment is a visual cue. The fastest of these cues is the exogenous cue tapping into the orientation reflex. These are not suitable for use with HMDs and so a replacement is needed.

To test if the extended map cue is a suitable replacement for an exogenous cue, the final experiment (chapter 5, experiment 2), compared the HMD delivered cues with an environment centred exogenous cue. There was no significant difference in search times between the map cue and the exogenous cue. In a real-world setting in which the user needs to navigate and search the environment, simplified map based cues which account for depth are an effective replacement for exogenous visual cues. Map cues delivered by the HMD can lead users to areas of an environment and objects of interest as quickly as traditional alternatives. When maps cues are unavailable due to design limitations, arrow cues like those used in chapter 5 are also a reliable, if slower way, to direct your user's attention from the device out to a specific point in the environment.

References

Ablaβmeier, M., Poitschke, T., Bengler, F. W. K., & Rigoll, G. (2007). *Eye gaze studies comparing head-up and head-down displays in vehicles*. In Multimedia and Expo, July 2007 IEEE International Conference (pp. 2250-2252). IEEE.

Abrams, R., & Christ, S. (2003). *Motion Onset Captures Attention*. Psychological Science, 14(5), 427-432. http://dx.doi.org/10.1111/1467-9280.01458

Abrams, R. A., Meyer, D. E., & Kornblum, S. (1989). *Speed and accuracy of saccadic eye movements: characteristics of impulse variability in the oculomotor system*. Journal of Experimental Psychology: Human Perception and Performance, 15(3), 529.

Andersen, G. (1990). *Focused attention in three-dimensional space*. Perception & Psychophysics, 47(2), 112-120. http://dx.doi.org/10.3758/bf03205975

Andersen, G., & Kramer, A. (1993). *Limits of focused attention in three-dimensional space*. Perception & Psychophysics, 53(6), 658-667. http://dx.doi.org/10.3758/bf03211742

Anderson, S. J., Mullen, K. T., & Hess, R. F. (1991). *Human peripheral spatial resolution for achromatic and chromatic stimuli: limits imposed by optical and retinal factors*. *The Journal of Physiology*, *442*(1), 47-64.

Ansorge, U., Carbone, E., Becker, S., & Turatto, M. (2010). *Attentional capture by motion onsets is spatially imprecise*. European Journal of Cognitive Psychology, 22(1), 62-105. http://dx.doi.org/10.1080/09541440902733190

Apple. (2017). Watch. Retrieved 26 January 2017, from http://www.apple.com/watch/

Ar-tracking.com. (2015). DTrack2 - Software - Products - ART Advanced Realtime Tracking. Retrieved 17 December 2015, from http://www.ar-tracking.com/products/software/dtrack2/

Armstrong, D. G., Rankin, T. M., Giovinco, N. A., Mills, J. L., & Matsuoka, Y. (2014). *A heads-up display for diabetic limb salvage surgery: a view through the google looking glass*. Journal of diabetes science and technology, 8(5), 951-956.

Arnott, S., & Shedden, J. (2000). *Attention switching in depth using random-dot autostereograms: Attention gradient asymmetries*. Perception & Psychophysics, 62(7), 1459-1473. http://dx.doi.org/10.3758/bf03212146

Atchley, P., Kramer, A., Andersen, G., & Theeuwes, J. (1997). *Spatial cuing in a stereoscopic display: Evidence for a "depth-aware" attentional focus*. Psychonomic Bulletin & Review, 4(4), 524-529. http://dx.doi.org/10.3758/bf03214343

Baddeley, A. D. (1995). Working Memory. In: *The cognitive neurosciences*. Gazzaniga, M.S. (Ed); Publisher: The MIT Press; pp. 755-764.

Barde, A., Ward, M., Helton, W., Lee, G., & Billinghurst, M. (2016). *Attention Redirection Using Binaurally Spatialised Cues Delivered Over a Bone Conduction Headset*. Manuscript in preparation.

Barnes, B. (2010). Disney Command Center Aims to Keep Lines Moving. Nytimes.com. Retrieved 26 January 2017, from http://www.nytimes.com/2010/12/28/business/media/28disney.html

Borgmann, H., Rodríguez Socarrás, M., Salem, J., Tsaur, I., Gomez Rivas, J., Barret, E., & Tortolero, L. (2016). *Feasibility and safety of augmented reality-assisted urological surgery using smartglass*. World Journal Of Urology. http://dx.doi.org/10.1007/s00345-016-1956-6

Botella, C., Bretón-López, J., Quero, S., Baños, R., & García-Palacios, A. (2010). *Treating Cockroach Phobia With Augmented Reality*. Behaviour Therapy, 41(3), 401-413. http://dx.doi.org/10.1016/j.beth.2009.07.002

Bradshaw, J. L. (1968a). *Pupil size and problem solving*. Quarterly Journal of Experimental Psychology, 20: 116-122.

Bradshaw, J. L. (1968b). *Load and pupillary changes in continuous processing tasks*. British Journal of Psychology, 59: 265-271.

Broadbent (1957). *A mechanical model for human attention and immediate memory*. Psychological Review, 64: 205-215.

Broadbent, D. E. (1958) *Perception and communication*. London, England: Pergamon Press Ltd.

Card, S., English, W., & Burr, B. (1978). *Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT*. Ergonomics, 21(8), 601-613. http://dx.doi.org/10.1080/00140137808931762

Carrasco, M., & McElree, B. (2001). Covert attention accelerates the rate of visual information processing. Proceedings Of The National Academy Of Sciences, 98(9), 5363-5367. http://dx.doi.org/10.1073/pnas.081074098

Carrasco, M., & Yeshurun, Y. (1998). *The contribution of covert attention to the set-size and eccentricity effects in visual search*. Journal Of Experimental Psychology: Human Perception And Performance, 24(2), 673-692. http://dx.doi.org/10.1037/0096-1523.24.2.673

Chai, P., Babu, K., & Boyer, E. (2015). *The Feasibility and Acceptability of Google Glass for Teletoxicology Consults*. Journal Of Medical Toxicology, 11(3), 283-287. http://dx.doi.org/10.1007/s13181-015-0495-7

Colavita, F.B. (1974). *Human sensory dominance*. Perception and Psychophysics, 16: 409–412.

Cui, Y., Chipchase, J., & Ichikawa, F. (2007). A cross culture study on phone carrying and physical personalization. *Usability and Internationalization. HCI and Culture*, 483-492.

Deutsch, J.A.; Deutsch, D. (1963). *Attention: Some theoretical considerations*. Psychological Review 70: 80–90.

Di Fede, D., & Mills, A. (2017). Minim. Retrieved 24 July 2017, from https://github.com/ddf/Minim

Dirkin, G. R. (1983). *Cognitive tunneling: use of visual information under stress*. Perceptual and Motor Skills, 56(1), 191-198.

Downing, C. (1988). *Expectancy and visual-spatial attention: Effects on perceptual quality*. Journal Of Experimental Psychology: Human Perception And Performance, 14(2), 188-202. http://dx.doi.org/10.1037/0096-1523.14.2.188

Driver, J., & Spence, C. (1998). *Cross–modal links in spatial attention*. Philosophical Transactions of the Royal Society B: Biological Sciences, 353(1373), 1319-1331.

Egeth, H. E., & Yantis, S. (1997). *Visual attention: Control, representation, and time course*. Annual review of psychology, 48(1), 269-297.

Epling, S. L., Blakely, M. J., Russell, P. N., & Helton, W. S. (2016). *Free recall and outdoor running: cognitive and physical demand interference*. Experimental brain research, 234(10), 2979-2987.

Eriksen, C. W., & Hoffman, J. E. (1972). *Temporal and spatial characteristics of selective encoding from visual displays*. Attention, Perception, & Psychophysics, 12(2), 201-204.

Eriksen, C.W. & St James, J.D. (1986). *Visual attention within and around the field of focal attention: A zoom lens model*. Perception & Psychophysics, 40(4), 225-240.

Ernst, Z., Palmer, J., & Boynton, G. (2012). *Dividing attention between two transparent motion surfaces results in a failure of selective attention*. Journal of Vision, 12(12), 6-6. http://dx.doi.org/10.1167/12.12.

Fechner, G. (1966). Elements of psychophysics (H. Adler, Trans.) Vol. I. New York: Holt, Rinehart and Winston.

Finlayson, N., & Grove, P. (2014). *Visual search is influenced by 3D spatial layout*. Journal Of Vision, 14(10), 914-914. http://dx.doi.org/10.1167/14.10.914

Fischer, E., & Haines, R. F. (1980). *Cognitive issues in head-up displays*. Retrieved from https://ntrs.nasa.gov/search.jsp?R=19810005125

Garcia-Palacios, A., Hoffman, H., Carlin, A., Furness, T., & Botella, C. (2002). *Virtual reality in the treatment of spider phobia: a controlled study*. Behaviour Research And Therapy, 40(9), 983-993. http://dx.doi.org/10.1016/s0005-7967(01)00068-7

GitHub. (2015). vrpn/vrpn. Retrieved 17 December 2015, from http://vrpn.org

Google. (2017). Google Developers | Glass at Work. Retrieved 7 April 2017, from https://developers.google.com/glass/distribute/glass-at-work

Guthrie, J. T., Bennett, S., & Weber, S. (1991). *Processing procedural documents: A cognitive model for following written directions*. Educational Psychology Review, 3(3), 249-265.

Hachman, M. (2016, Jan 11). Go Meet Alto Tech's 'Cool Glass,' a vastly cheaper alternative to Google Glass. *PCWorld*. Retrieved from http://www.pcworld.com/article/3020573/consumer-electronics/meet-alto-techs-cool-glass-a-vastly-cheaper-alternative-to-google-glass.html

Haines, R. F., Fischer, E., & Price, T. A. (1980). *Head-up transition behaviour of pilots with and without head-up display in simulated low-visibility approaches*. Retrieved from https://ntrs.nasa.gov/search.jsp?R=19810005124

Hansen, T., Pracejus, L., & Gegenfurtner, K. (2009). *Colour perception in the intermediate periphery of the visual field*. Journal Of Vision, 9(4), 26-26. http://dx.doi.org/10.1167/9.4.26

Henderson, J. M., & Macquistan, A. D. (1993). *The spatial distribution of attention following an exogenous cue*. Attention, Perception, & Psychophysics, 53(2), 221-230.

Hillstrom, A., & Yantis, S. (1994). *Visual motion and attentional capture*. Perception & Psychophysics, 55(4), 399-411. http://dx.doi.org/10.3758/bf03205298

Hoffman, J. E., & Nelson, B. (1981). *Spatial selectivity in visual search*. Perception & Psychophysics, 30, 283-290.

Hoffman, J., & Subramaniam, B. (1995). *The role of visual attention in saccadic eye movements*. Perception & Psychophysics, 57(6), 787-795. http://dx.doi.org/10.3758/bf03206794

Hollister, S. (2017, July 6). The rise and not-quite-fall of Pokemon Go. *CNET*. Retrieved from https://www.cnet.com/news/pokemon-go-million-dollar-monthly-active-users/

Horrey, W., & Wickens, C. (2004). *Driving and Side Task Performance: The Effects of Display Clutter, Separation, and Modality*. Human Factors: The Journal Of The Human Factors And Ergonomics Society, 46(4), 611-624. http://dx.doi.org/10.1518/hfes.46.4.611.56805

Hwang, A. D., & Peli, E. (2014). *An augmented-reality edge enhancement application for Google Glass*. Optometry and vision science: official publication of the American Academy of Optometry, 91(8), 1021.

Hyman, I. E., Boss, S. M., Wise, B. M., McKenzie, K. E., & Caggiano, J. M. (2010). *Did you see the unicycling clown? Inattentional blindness while walking and talking on a cell phone*. Applied Cognitive Psychology, 24(5), 597-607.

Intriligator, J., & Cavanagh, P. (2001). *The spatial resolution of visual attention*. Cognitive psychology, 43(3), 171-216.

Johnson, M. (1986). *Colour Vision in the Peripheral Retina*. Optometry And Vision Science, 63(2), 97-103. http://dx.doi.org/10.1097/00006324-198602000-00003

Jonides, J. (1981) *Voluntary and Automatic control over the mind's eye's movement*. J.B. Long, A.D. Baddeley (Eds.), Attention and Performance IX, Lawrence Erlbaum Associates, Hillsdale NJ

Jonides, J. (1983). *Further toward a model of the Mind's eye's movement.* Bulletin Of The Psychonomic Society, 21(4), 247-250. http://dx.doi.org/10.3758/bf03334699

Jonides, J., & Yantis, S. (1988). *Uniqueness of abrupt visual onset in capturing attention*. Perception & Psychophysics, 43(4), 346-354. http://dx.doi.org/10.3758/bf03208805

Kahneman. D. (1973). *Attention and Effort*. Englewood Cliffs, NJ: Prentice-Hall Inc.

Kahneman D. and Beatty, J. (1966). *Pupil diameter and load on memory*. Science, 154: 1583-1585.

Kahneman D. and Beatty, J. (1967). *Pupillary responses in a pitch-discrimination task*. Perception and Psychophysics, 2: 101-105.

Kahneman D. and Peavler, W. S. (1969) *Incentive effects and pupillary changes in association learning*. Journal of Experimental Psychology, 79: 312-318.

Kahneman, D., Peavler, W. S., and Onuska, L. (1968). *Effects of verbalization and incentive on the pupil response to mental activity*. Canadian Journal of Psychology, 22: 186-196.

Kantowitz, B. H. and Knight, J. L. (1976). *Testing tapping timesharing, II: Auditory secondary task*. Acta Psychologica, 40: 343-362.

Keizer, G. (2017). Windows comes up third in OS clash two years early. Computerworld. Retrieved 25 January 2017, from http://www.computerworld.com/article/3050931/microsoft-windows/windows-comes-up-third-in-os-clash-two-years-early.html

Klochek, C., & MacKenzie, I. S. (2006). Performance measures of game controllers in a three-dimensional environment. Proceedings of Graphics Interface 2006, pp. 73-79. Toronto: CIPS.

Kruley, P., Sciama, S. C., & Glenberg, A. M. (1994). *On-line processing of textual illustrations in the visuospatial sketchpad: Evidence from dual-task studies*. Memory & Cognition, 22(3), 261-272.

LaBerge, D. (1983). *Spatial extent of attention to letters and words*. Journal of Experimental Psychology: Human Perception and Performance, 9(3), 371.

Lager, P. (2017). Game Control Plus 1.2.1. Retrieved 24 July 2017, from https://forum.processing.org/two/discussion/5269/game-control-plus-a-new-library

Latham, K., & Whitaker, D. (1996). *A Comparison of Word Recognition and Reading Performance in Foveal and Peripheral Vision*. Vision Research, 36(17), 2665-2674. http://dx.doi.org/10.1016/0042-6989(96)00022-3

Lavie, N., & Tsal, Y. (1994). *Perceptual load as a major determinant of the locus of selection in visual attention*. Attention, Perception, & Psychophysics, 56(2), 183-197.

LeFevre, J. A., & Dixon, P. (1986). *Do written instructions need examples?*. Cognition and Instruction, 3(1), 1-30.

Lisberger, S. (2010). *Visual Guidance of Smooth-Pursuit Eye Movements: Sensation, Action, and What Happens in Between*. Neuron, 66(4), 477-491. http://dx.doi.org/10.1016/j.neuron.2010.03.027

Li, F. (2017). bluetoothDesktop. Retrieved 24 July 2017, from http://www.extrapixel.ch/processing/bluetoothDesktop/

Liu, Y. C., & Wen, M. H. (2004). *Comparison of head-up display (HUD) vs. head-down display (HDD): driving performance of commercial vehicle operators in Taiwan*. International Journal of Human-Computer Studies, 61(5), 679-697.

Lovejoy, L., Fowler, G., & Krauzlis, R. (2009). *Spatial allocation of attention during smooth pursuit eye movements*. Vision Research, 49(10), 1275-1285. http://dx.doi.org/10.1016/j.visres.2009.01.011

Lv, Z., Feng, S., Feng, L., & Li, H. (2015, March). *Extending touch-less interaction on vision based wearable device*. In Virtual Reality (VR), 2015 iEEE (pp. 231-232). IEEE.

MacKenzie, I.S., Kauppinen, T., Silfverberg, M., 2001. Accuracy measures for evaluating computer pointing devices. In: Proceedings of ACM Conference on Human Factors in Computing Systems. ACM Press, New York, NY, pp. 9–16 .

Magers, C. S. (1983, December). An experimental evaluation of on-line HELP for non-programmers. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 277-281). ACM.

Mangun, G., & Hillyard, S. (1988). *Spatial gradients of visual attention: behavioural and electrophysiological evidence*. Electroencephalography And Clinical Neurophysiology, 70(5), 417-428. http://dx.doi.org/10.1016/0013-4694(88)90019-3

McCormick, P. A., & Klein, R. (1990). *The spatial distribution of attention during covert visual orienting*. Acta Psychologica, 75(3), 225-242.

McCormick, P. A., Klein, R. M., & Johnston, S. (1998). *Splitting versus sharing focal attention: Comment on Castiello and Umiltà (1992)*. Journal of Experimental Psychology-Human Perception and Performance, 24(1), 350-357.

Microsoft.com (2017). Visual Basic. Retrieved 24 July 2017, from https://docs.microsoft.com/en-us/dotnet/visual-basic/

Müller, H., & Findlay, J. (1988). *The effect of visual attention of peripheral discrimination thresholds in single and multiple element displays*. Acta Psychologica, 69(2), 129-155. http://dx.doi.org/10.1016/0001-6918(88)90003-0

Müller, H., & Rabbitt, P. (1989). *Reflexive and voluntary orienting of visual attention: Time course of activation and resistance to interruption*. Journal of Experimental Psychology: Human Perception and Performance, 15(2), 315-330. http://dx.doi.org/10.1037//0096-1523.15.2.315

Mullet, K., & Sano, D. (1994). *Designing visual interfaces: Communication oriented techniques.* New Jersey, USA: Prentice Hall.

Nakayama, K., & Mackeben, M. (1989). *Sustained and transient components of focal visual attention*. Vision Research, 29(11), 1631-1647. http://dx.doi.org/10.1016/0042-6989(89)90144-2

Nasar, J. L., & Troyer, D. (2013). *Pedestrian injuries due to mobile phone use in public places*. Accident Analysis & Prevention, 57, 91-95.

Navon, D., & Gopher, D. (1979). *On the economy of the human processing system*. Psychological Review, 86, 254-284.

Neisser, U. (1963). *Decision-Time without Reaction-Time: Experiments in Visual Scanning*. The American Journal Of Psychology, 76(3), 376. http://dx.doi.org/10.2307/1419778

Neisser, U., & Becklen, R. (1975). *Selective looking: Attending to visually specified events*. Cognitive Psychology, 7(4), 480-494. http://dx.doi.org/10.1016/0010-0285(75)90019-5

NianticLabs (2015). fieldtripper.com Retrieved 7 April 2017, from http://www.fieldtripper.com/

Niantic, Inc. (2016). Pokemongo.com. Retrieved 7 April 2017, from http://www.pokemongo.com/

Novick, L. R., & Morse, D. L. (2000). *Folding a fish, making a mushroom: The role of diagrams in executing assembly procedures*. Memory & Cognition, 28(7), 1242-1256.

Nuñez, M. (2015). *Google and Levi's Are Weaving Computers Into Your Clothes*. Popular Science. Retrieved 26 January 2017, from http://www.popsci.com/googles-levi's-computers-clothing-project-jacquard

O'Hear, S. (2017). Could Fitness Be The Killer App For Google Glass? Race Yourself Crowd Funds £100K For GlassFit App. TechCrunch. Retrieved 7 April 2017, from https://techcrunch.com/2013/10/09/race-yourself/

Olivarez-Giles, N. (2017). Which Devices Rule the Workplace?. Wall Street Journal. Retrieved 24 January 2017, from http://www.wsj.com/articles/which-devices-rule-the-workplace-1457921547

Onnasch, L., Wickens, C. D., Li, H., & Manzey, D. (2014). *Human performance consequences of stages and levels of automation: An integrated meta-analysis*. Human Factors, 56(3), 476-488.

OptiTrack. (2017). Motive - Optical motion capture software. Retrieved 25 June 2017, from http://optitrack.com/products/motive/

Oulasvirta, A., Tamminen, S., Roto, V., & Kuorelahti, J. (2005, April). Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 919-928). ACM.

Paivio, A., Yuille, J. C., & Madigan, S. A. (1968). *Concreteness, imagery, and meaningfulness values for 925 nouns.* Journal of experimental psychology, 76, 1-25.

Parks, N., & Corballis, P. (2006). *Attending to depth: electrophysiological evidence for a viewer-centered asymmetry*. Neuroreport, 17(6), 643-647. http://dx.doi.org/10.1097/00001756-200604240-00017

Pashler, H. (1994a). *Dual-task interference in simple tasks: Data and theory*. Psychological Bulletin, 116(2), 220-244. http://dx.doi.org/10.1037/0033-2909.116.2.220

Pashler, H. (1994b). *Graded capacity-sharing in dual-task interference?* Journal Of Experimental Psychology: Human Perception And Performance, 20(2), 330-342. http://dx.doi.org/10.1037/0096-1523.20.2.330

Patel, S., Kientz, J., Hayes, G., Bhat, S., and Abowd, G. (2006) *Farther than you may think: An empirical investigation of the proximity of users to their mobile phones*. Ubiquitous Computing; 123-140

Pitichat, T. (2013). *Smartphones in the workplace: Changing organizational behaviour, transforming the future*. LUX: A Journal of Transdisciplinary Writing and Research from Claremont Graduate University, 3(1), 13.

Posner, M. I. (1980) *Orienting of attention*. Quarterly Journal of Experimental Psychology, 32(1), 3-25, DOI: 10.1080/00335558008248231

Posner, M. I., & Boies, S. J. (1971). *Components of attention*. Psychological Review, 78: 391-408.

Posner, M. I., Nissen, M. J., & Ogden, W. C. (1978). *Attended and unattended processing modes: The role of set for spatial location*. Modes of perceiving and processing information, 137, 158.

Posner, M., Snyder, C., & Davidson, B. (1980). *Attention and the detection of signals*. Journal Of Experimental Psychology: General, 109(2), 160-174. http://dx.doi.org/10.1037/0096-3445.109.2.160

Processing.org. (2015). Processing.org. Retrieved 2 February 2015, from https://processing.org/

Rashbass, C. (1961). *The relationship between saccadic and smooth tracking eye movements*. The Journal Of Physiology, 159(2), 326-338. http://dx.doi.org/10.1113/jphysiol.1961.sp006811

Rasmussen, J. (1983). *Skills, rules, and knowledge: Signals, signs, and symbols, and other distinctions in human performance models*. IEEE Transactions on Systems, Man, and Cybernetics, 13(3): 257-266.

Recon Instruments. (2015). Recon Jet. Smart eyewear for active lifestyles | Recon Instruments. Retrieved 17 December 2015, from http://www.reconinstruments.com/products/jet/

Reimer, B. (2009). *Impact of cognitive task complexity on drivers' visual tunneling*. Transportation Research Record: Journal of the Transportation Research Board, (2138), 13-19.

Remington, R., & Pierce, L. (1984). *Moving attention: Evidence for time-invariant shifts of visual selective attention*. Attention, Perception, & Psychophysics, 35(4), 393-399.

Rodrıguez, V., Valdes-Sosa, M., & Freiwald, W. (2002). *Dividing attention between form and motion during transparent surface perception.* Cognitive Brain Research, 13(2), 187-193.

Samsung. (2017). Smart watches & Wearables Including Gear S2 & Gear VR | Samsung UK. Retrieved 26 January 2017, from http://www.samsung.com/uk/wearables/all-wearables/

Sawyer, B. D., Finomore, V. S., Calvo, A. A., & Hancock, P. A. (2014). *Google Glass: A driver distraction cause or cure?.* Human factors, 56(7), 1307-1321.

Seiss, E., Kiss, M., & Eimer, M. (2009). *Does focused endogenous attention prevent attentional capture in pop-out visual search?* Psychophysiology, 46(4), 703-717.

Shepherd, M., Findlay, J., & Hockey, R. (1986). *The relationship between eye movements and spatial attention*. The Quarterly Journal Of Experimental Psychology Section A, 38(3), 475-491. http://dx.doi.org/10.1080/14640748608401609

Simons, D., & Chabris, C. (1999). *Gorillas in Our Midst: Sustained Inattentional Blindness for Dynamic Events.* Perception, 28(9), 1059-1074. http://dx.doi.org/10.1068/p281059

Simpson H. M. and Paivio, A. (1968) *Effects on pupil size of manual and verbal indicators of cognitive task fulfillment.* Perception and Psychophysics, 3: 185-196.

Sims, V., & Hegarty, M. (1997). *Mental animation in the visuospatial sketchpad: Evidence from dual-task studies.* Memory & Cognition, 25(3), 321-332. http://dx.doi.org/10.3758/bf03211288

Smith, S. L., & Mosier, J. N. (1986). *Guidelines for designing user interface software*. Bedford, MA: Mitre Corporation.

Sodhi, M., Reimer, B., Cohen, J. L., Vastenburg, E., Kaars, R., & Kirschenbaum, S. (2002, March). On-road driver eye movement tracking using head-mounted devices. In Proceedings of the 2002 symposium on Eye tracking research & applications (pp. 61-68). ACM.

Sperling, G., & Weichselgartner, E. (1995). *Episodic theory of the dynamics of spatial attention.* Psychological review, 102(3), 503.

Starner, T. (2013). *Project glass: An extension of the self*. IEEE Pervasive Computing, 12(2), 14-16.

Stavrinos, D., Byington, K. W., & Schwebel, D. C. (2011). *Distracted walking: cell phones increase injury risk for college pedestrians*. Journal of safety research, 42(2), 101-107.

Sternberg, S. (1969). *Memory-scanning: Mental processes revealed by reaction-time experiments*. American scientist, 57(4), 421-457.

Stevens, S. S. (1957). *On the psychophysical law*. Psychological review, 64(3), 153.

Strange, A. (2017). Google Glass Video Shows Off Turn-By-Turn Directions. PCMAG. Retrieved 7 April 2017, from http://www.pcmag.com/article2/0,2817,2423068,00.asp

Tang, A., Owen, C., Biocca, F., & Mou, W. (2003, April). Comparative effectiveness of augmented reality in object assembly. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 73-80). ACM.

Theeuwes, J. (1990). *Perceptual selectivity is task dependent: Evidence from selective search*. Acta Psychologica, 74(1), 81-99. http://dx.doi.org/10.1016/0001-6918(90)90036-f

Theeuwes, J. (1994). *Stimulus-driven capture and attentional set: Selective search for colour and visual abrupt onsets*. Journal Of Experimental Psychology: Human Perception And Performance, 20(4), 799-806. http://dx.doi.org/10.1037//0096-1523.20.4.799

Theeuwes, J., Atchley, P., & Kramer, A. (1998). *Attentional control within 3-D space*. Journal Of Experimental Psychology: Human Perception And Performance, 24(5), 1476-1485. http://dx.doi.org/10.1037/0096-1523.24.5.1476

Thomas, L., & Wickens, C. (2004). Eye-tracking and Individual Differences in off-Normal Event Detection when Flying with a Synthetic Vision System Display. Proceedings Of The Human Factors And Ergonomics Society Annual Meeting, 48(1), 223-227. http://dx.doi.org/10.1177/154193120404800148

Treisman, A. (1985). *Preattentive processing in vision.* Computer Vision, Graphics, And Image Processing, 31(2), 156-177. http://dx.doi.org/10.1016/s0734-189x(85)80004-9

Treisman, A. (1986). *Features and objects in visual processing*. Scientific American, 255(5), 114-125.

Treisman, A., & Gormican, S. (1988). *Feature analysis in early vision: Evidence from search asymmetries*. Psychological review, 95(1), 15.

Unity3d.com. (2015). Unity - Game Engine. Retrieved 17 December 2015, from https://unity3d.com/

Ververs, P. M., & Wickens, C. D. (1998). *Head-up displays: Effect of clutter, display intensity, and display location on pilot performance*. The International Journal of Aviation Psychology, 8(4), 377-403.

Ververs, P., & Wickens, C. (2000). Designing Head-Up Displays (HUDs) to Support Flight Path Guidance While Minimizing Effects of Cognitive Tunneling. Proceedings Of The Human Factors And Ergonomics Society Annual Meeting, 44(13), 45-48. http://dx.doi.org/10.1177/154193120004401312

Walker, B.N. and G. Kramer, *Auditory Displays, Alarms and Auditory Interfaces*, in *International Encyclopedia of Ergonomics and Human Factors*, W.K. Informa Healthcare, Editor. 2006, CRC Press. p. 1021 - 1025.

Wang, D. (2013). *A framework of smartphone use for travel*. Temple University.

Wang, Q., Cavanagh, P., & Green, M. (1994). *Familiarity and pop-out in visual search*. Perception & Psychophysics, 56(5), 495-500. http://dx.doi.org/10.3758/bf03206946

Ward, M., Barde, A., Russell, P., Billinghurst, M., & Helton, W. (2016). Visual Cues to Reorient Attention from Head Mounted Displays. Proceedings Of The Human Factors And Ergonomics Society Annual Meeting, 60(1), 1574-1578. http://dx.doi.org/10.1177/1541931213601363

Weihrauch, M., Meloeny, G. G., & Goesch, T. C. (1989). *The first head up display introduced by general motors* (No. 890288). SAE Technical Paper.

Wickens, C. D. (1976). *The effects of divided attention on information processing in tracking*. Journal of Experimental Psychology: Human Perception and Performance, 2: 1-13.

Wickens, C. D. (1980). *The structure of attentional resources*, in R. Nickerson (ed.). Attention and Performance VIII (Hillsdale, NJ: Lawrence Erlbaum): 239-257.

Wickens, C. D. (1996). Designing for stress. In J. Driskell & E. Salas (Eds.), *Stress and human performance* (pp. 279–295). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Wickens, C. D. (2002). *Multiple resources and performance prediction*. Theoretical issues in ergonomics science, 3(2), 159-177.

Wickens, C. D. (2017). What lumberjacks have to do with human-automation interaction [Video]. Purdue Industrial Engineering Distinguished Lecture Series. Retrieved 28th April 2017 from https://www.youtube.com/watch?v=HkYanYTffWM

Wickens, C. D., & Alexander, A. L. (2009). *Attentional tunneling and task management in synthetic vision displays*. The International Journal of Aviation Psychology, 19(2), 182-199.

Wilmer, J., & Nakayama, K. (2007). *Two Distinct Visual Motion Mechanisms for Smooth Pursuit: Evidence from Individual Differences*. Neuron, 54(6), 987-1000. doi:10.1016 /j.neuron.2007.06.007

Wolfe, J. (1994). *Guided Search 2.0 A revised model of visual search*. Psychonomic Bulletin & Review, 1(2), 202-238. http://dx.doi.org/10.3758/bf03200774

Wolfe, J., Cave, K., & Franzel, S. (1989). *Guided search: An alternative to the feature integration model for visual search*. Journal of Experimental Psychology: Human Perception and Performance, 15(3), 419-433. http://dx.doi.org/10.1037//0096-1523.15.3.419

Wright, P. and Kahneman, D. (1971) *Evidence for alternative strategies of sentence retention*. Quarterly Journal of Experimental Psychology, 23: 197-213.

Yantis, S., & Jonides, J. (1990). *Abrupt visual onsets and selective attention: voluntary versus automatic allocation*. Journal of Experimental Psychology: Human perception and performance, 16(1), 121.

Yantis, S., & Jonides, J. (1984). *Abrupt visual onsets and selective attention: evidence from visual search*. Journal of Experimental Psychology: Human perception and performance, 10(5), 601.

Yantis, S., & Jonides, J. (1996). *Attentional capture by abrupt onsets: New perceptual objects or visual masking?*. Journal Of Experimental Psychology: Human Perception And Performance, 22(6), 1505-1513. http://dx.doi.org/10.1037//0096-1523.22.6.1505

Yoo, H., Tsimhoni, O., Watanabe, H., Green, P., & Shah, R. (1999). *Display of HUD warnings to drivers: Determining an optimal location*. UMTRI Human Factors (report no. UMTRI-99-9).

Appendix 1.0: Programming, Data Analysis and Paper Materials for Experiment 1

Experiment 1 programming consisted of three components. The first of these was a program which would synchronisation and word presentations on the Google Glass. The code for this program is available in Appendix 1.1 with notation. This program was written in the Processing language, android version (Processing.org, 2015). The second component was a set of 5 programs which controlled the timing of the tasks and displayed compensatory tracking program on the desktop computer. One version of the code for these programs is available in Appendix 1.2 with notation. The tracking task programs connected to the Google Glass via Bluetooth and were written in the Processing language (Processing.org, 2015). These programs were also responsible for storing and printing the data for later analysis. An example section of the output file can be found in Appendix 1.2a. The final component was a pair of excel macros programmed in the Visual Basic language (Microsoft.com, 2017). These two macros are available in Appendix 1.3.

Two paper materials were used during the Experiment. These are available in Appendix 1.4. The first of these materials is a page held by the experiments which has the four word lists printed on it. The experimenter uses this sheet to mark off words as they are detected and read aloud. The second page has 4 sets of blank lines and is used by the participant to report the words they remember after each block of the experiment finishes.

Experiment 1 was conducted in co-operation with the 2014 PSYC415 class and under the blanket low-risk ethics approval for class experiments and examples.

Appendix 1.1: Google Glass Word Display Program

```
//Import libraries to access native android Bluetooth functions and java language conversion
 import android.bluetooth.BluetoothAdapter ;
 import android.bluetooth.BluetoothDevice ;
 import android.bluetooth.BluetoothSocket ;
 import android.content.BroadcastReceiver ;
 import android.content.Context ;
 import android.content.Intent ;
 import android.content.IntentFilter ;
 import java.util.ArrayList ;
 import java.io.IOException ;
 import java.io.InputStream ;
 import java.io.OutputStream ;
 import java.lang.reflect.Method ;
 import java.util.UUID;

//Declare miscellaneous global variables
//connection globals
 private static final int REQUEST_ENABLE_BT = 3 ;
 ArrayList devices ;
```

```
 BluetoothAdapter adapter ;
 BluetoothDevice device ;
 BluetoothSocket socket ;
 InputStream ins ;
 OutputStream ons ;
 boolean registered = false ;
//display globals
 PFont f1 ;
 PFont f2 ;
 PFont f3 ;
 int state ;
 String error ;
 byte value = 0;
 String msg = "";
 String msg2 = "";
 int inputCondition = 0;
 int inputDisplay = 0;
 int tickingSquare = 0;
 String testString = "";

//This list contains the names of the devices the Glass should attempt to connect to, this is to
prevent accidental connections to passing open devices.
String[] targetDeviceArray =
{
  "MESSENGER",
  "psyc848",
  "psyc848.",
  "psyc846.",
  "psyc846"
};

//These arrays contain the list of words to be memorized. Internal order is not randomised.
Condition order is randomised by the experimenter.
String[] wordList1 =
{
"slipper",
"glacier",
"pepper",
"settler",
"odour",
"dreamer",
"cigar",
"captive",
"banner",
"poster",
"arrow",
"piston",
"pencil",
```

```
"hardwood",
"beggar",
"invoice",
"robber",
"elbow",
"butcher",
"salute"
};

String[] wordList2 =
{
"candy",
"reptile",
"comrade",
"sunset",
"shotgun",
"profile",
"scarlet",
"harness",
"bandit",
"portrait",
"barrel",
"sunburn",
"warbler",
"lobster",
"rubble",
"bouquet",
"hostage",
"cellar",
"tweezers",
"abyss"
};

String[] wordList3 =
{
"fabric",
"mucus",
"hurdle",
"leopard",
"pianist",
"footwear",
"cuisine",
"hammer",
"trellis",
"garments",
"nectar",
"sultan",
"trumpet",
```

```
"doorman",
"hillside",
"pudding",
"kettle",
"daybreak",
"speaker",
"portal"
};

String[] wordList4 =
{
"costume",
"saloon",
"kerchief",
"python",
"fiord",
"bullet",
"mammal",
"salad",
"lemon",
"twilight",
"abode",
"jelly",
"singer",
"goblet",
"painter",
"wigwam",
"typhoon",
"basement",
"sulphur",
"juggler"
};
```

// This function was dsigned by Mark Billinghurst and serves to keep the Glass awake even during periods of inactivity. Prevents the Glass from going into standby mode during the experiment.

```
// ScreenOn.pde
// Mark Billinghurst - May 2013

import android.view.WindowManager;
import android.view.View;
import android.os.Bundle;

void onCreate(Bundle bundle)
{
  super.onCreate(bundle);
  // fix so screen doesn't go to sleep when app is active
  getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
}
```

```
//This sets up the Bluetooth adapter's receiver
BroadcastReceiver receptor = new BroadcastReceiver ( )
 {
    public void onReceive ( Context context, Intent intent )
    {
       println ( "onReceive" ) ;
       String action = intent. getAction ( ) ;

       if ( BluetoothDevice. ACTION_FOUND . equals ( action ) )
       {
          BluetoothDevice device = intent. getParcelableExtra ( BluetoothDevice. EXTRA_DEVICE ) ;
          println ( device. getName ( ) + " " + device. getAddress ( ) ) ;
          devices. add ( device ) ;
       }
       else if ( BluetoothAdapter. ACTION_DISCOVERY_STARTED . equals ( action ) )
       {
         state = 0 ;
         println ( "Search Begins" ) ;
       }
       else if ( BluetoothAdapter. ACTION_DISCOVERY_FINISHED . equals ( action ) )
       {
         state = 1 ;
         println ( "Search Ends" ) ;
       }
    }
 } ;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//This function runs once at the beginning of the program after the setup and declarations are
complete. It creates the fonts and sets the text display colour to white.
 void setup ( )
 {
   frameRate ( 30 ) ;
   f1 = createFont ( "Arial" , 20 , true ) ;
   f2 = createFont ( "Arial" , 15 , true ) ;
   f3 = createFont ( "Arial" , 80 , true ) ;
   stroke ( 255 ) ;
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//This function runs every frame of the program. It controls the majority of the actions in the
program.
 void draw ( )
{
   background(0,0,0); //Draw a black background

   //Controls the connection setup process. the "msg" string is displayed in small text in the top left
hand corner of the screen and is used to debug connection issues.
   switch ( state )
```

```
 {
   case 0 :
     msg = "Searching...";
     break ;
   case 1 :
     msg = "Device Found";
     listDevices ("test", colour(0, 0, 0) );
     break ;
   case 2 :
     connectDevice ( ) ;
     break ;
   case 3 :
     sampleData ( ) ;
     break ;
   case 4 :
     showError ( ) ;
     break ;
 }

//This creates a small square in the bottom right hand corner of the Glass's display which flashes a dim grey colour every 1/3rd of a second. This is to debug situations in which the Glass side program may have frozen.
 tickingSquare = tickingSquare + 1;
 if (tickingSquare%20 < 10)
 {
  stroke(60,60,60);
  fill(60,60,60);
  rect(width-60,height-60,20,20);
 }

//Display debug text in the top left hand corner ("msg","msg2")
 fill(220,220,220);
 textAlign(LEFT);
 textFont(f1,20);
 text(msg,10,20);
 textFont(f2,15);
 text(msg2,10,40);

 //If a word should be displayed, display the current word from the appropriate list. The word is displayed in the center of the screen in size 80 text.
 textAlign(CENTER);
 textFont(f3,80);
 inputCondition = (int(value)-(int(value)%21))/21;
 inputDisplay = (int(value)%21);
 if (inputDisplay != 0)
 {
  switch(inputCondition)
  {
```

```
    case 0:
        text(wordList1[value%21-1],width/2,height/2-20);
    break;
    case 1:
        text(wordList2[value%21-1],width/2,height/2-20);
    break;
    case 2:
        text(wordList3[value%21-1],width/2,height/2-20);
    break;
    case 3:
        text(wordList4[value%21-1],width/2,height/2-20);
    break;
    default:
        text(str(inputCondition) + " " + str(inputDisplay),width/2,height/2-20);
    break;
   }
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

//This function runs once at the beginning of the program and brings up a pop-up to enable the Glass's Bluetooth component. If Bluetooth is already enabled, it moves to the next function.

```
void onStart ( )
 {
  super . onStart ( ) ;
  println ( "onStart" ) ;
  adapter = BluetoothAdapter. getDefaultAdapter ( ) ;
  if ( adapter != null )
  {
   if ( ! adapter. isEnabled ( ) )
   {
      Intent enableBtIntent = new Intent ( BluetoothAdapter. ACTION_REQUEST_ENABLE ) ;
      startActivityForResult ( enableBtIntent, REQUEST_ENABLE_BT ) ;
   }
   else
   {
    customBegin ( ) ;
   }
  }
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

//Once Bluetooth is enabled, begin the search for nearby devices. Glass debug will update to say that the Glass is searching for a connection. If at least one device is found, the next function is run on the new draw cycle.

```
void customBegin ( )
 {
   devices = new ArrayList ( ) ;
   for ( BluetoothDevice device : adapter. getBondedDevices ( ) )
   {
```

```
      devices. add ( device ) ;
      println("Found a device");
    }
    state = 1 ;
 }
```
//
//Cycles through the devices found in the previous function and checks them against the approved list of devices. If one matches, attempt to make a connection with the next function.
```
 void listDevices ( String customText, colour c )
 {
  if ( devices != null )
  {
    for ( int index = 0 ; index < devices. size ( ) ; index ++ )
    {
      BluetoothDevice device = ( BluetoothDevice ) devices. get ( index ) ;
      println ( "Found the device " + device.getName ( ) + " in position " + index) ;
      for (int i = 0; i < targetDeviceArray.length ; i++)
      {
        if ( device.getName ( ).equals(targetDeviceArray[i]) )
        {
        println ( "Chose the device " + device.getName ( ) + " in position " + index) ;
        verifyChoice ( index ) ;
        break;
        }
      }
    }
  }
 }
```
//
//Verifies the chosen device and updates the debug information on the glass screen. Next draw frame, the connection will be attempted.
```
 void verifyChoice ( int chosen )
 {
  if ( chosen < devices. size ( ) )
  {
    device = ( BluetoothDevice ) devices. get ( chosen ) ;
    println ( device. getName ( ) + " confirmed" ) ;
    state = 2 ;
  }
 }
```
//
//Attempts to connect to the chosen device. If successful, move to polling incoming data for synchronization messages. If it fails, display an error to the screen.
```
 void connectDevice ( )
 {
  msg = "Attempting to Connect";
  msg2 = device. getName ( );
  adapter.cancelDiscovery ();
```

```
  try
  {
  socket = device.createRfcommSocketToServiceRecord(UUID.fromString("00001101-0000-1000-
8000-00805F9B34FB"));

  socket. connect ( ) ;
  ons = socket. getOutputStream ( ) ;
  ins = socket. getInputStream ( ) ;

  state = 3 ;

  msg = device. getName ( );
  msg2 = device. getAddress ( );

  }
  catch ( Exception ex )
  {
   state = 4 ;
   error = ex. toString ( ) ;
   println ( error ) ;
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Once a connection has been made, check each frame to see if a sync. message has come in.
void sampleData ( )
 {
  try
  {
   while ( ins. available ( ) > 0 )
   {
    value = ( byte ) ins. read ( ) ;
   }
  }
  catch ( Exception ex )
  {
   state = 4 ;
   error = ex. toString ( ) ;
   println ( error ) ;
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//If the program is closed, terminate the Bluetooth connection before closing.
 void onStop ( )
 {
  println ( "onStop" ) ;
  /*
   if(registered)
   {
```

```
    unregisterReceiver(receptor);
   }
   */

  if ( socket != null )
  {
   try
   {
    socket. close ( ) ;
   }
   catch ( IOException ex )
   {
    println ( ex ) ;
   }
  }
  super . onStop ( ) ;
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//An alternate method to enable Bluetooth. Triggers if an incoming Bluetooth connection exists. If the Bluetooth connection is interrupted, this function will display an error message to the screen.
void onActivityResult ( int requestCode, int resultCode, Intent data )
 {
  println ( "onActivityResult" ) ;
  if ( resultCode == RESULT_OK )
  {
   println ( "RESULT_OK" ) ;
   customBegin ( ) ;
  }
  else
  {
   println ( "RESULT_CANCELED" ) ;
   state = 4 ;
   error = "Bluetooth is not active" ;
  }
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Used to display a simple error message
void showError ()
 {
  msg = "error";
  msg2 = "error";
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Appendix 1.2: Experiment 1 Compensatory Tracking Task

```
import procontroll.*; //The procontroll library (Lager, 2017) enables access to the joystick information.
```

```
import bluetoothDesktop.*; //The bluetoothDesktop library (Li, 2017) enables access to the bluetooth and connection functions.
import java.util.UUID;
import ddf.minim.*; //The minim library (Di Fede & Mills, 2017) handles the audio player for this program.

//Global variables for the procontroll library
ControllIO controll;
ControllDevice device;
ControllStick stick;
ControllButton button;

//Global variables for the bluetoothDesktop library
Bluetooth bt;
PApplet papplet;
Client currentClient;
String currentClientName = "";

//Global variables for the Minim library
AudioPlayer beep;
Minim beeper;
AudioInput input;

//These variables are settings for the quick adjustment for the difficulty of the task. The current values were used during the experiment. Time is in frames (30 frames per second), speed is in pixels per frame.
//Control Variables
int wordDisplayTime = 60;
float displacementSpeed=15;
float joySpeed = 5.0;
int displayWordList = 0;

//These globals control the movement of the cursor
float joyMouseX = floor(200 + random(624));
float joyMouseY = floor(200 + random(368));
float joyTravelX = 0;
float joyTravelY = 0;
float displacementX = 0;
float displacementY = 0;
int joyClick=0;
int targetZone = floor(random(9));

//These globals control the stages of the program and the display of information to the screen
int startMillis;
int stepState = 1;
int directionChangeHappened = 0;
int wordPresentationHappened = 0;
int stageCounter;
```

```
String typedText = "";
PFont font;
PFont fontSmall;
int[] displayMillisArray = new int[21];
int displayCountdown = 0;
int displayWordCounter = 0;
int[] displayDirectionChange = {0,1,1,0,1,1,0,1,0,1,0,0,0,1,0,1,0,1,0,1};

//These globals calculate and store data values for printing out
float[] arrayDisplacement = new float[0];
float[] arraySpeedDifferential = new float[0];
float[] arrayDragSpeed = new float[0];
float[] arrayUserSpeed = new float[0];
float[] arrayMillis = new float[0];
int[] arrayDirectionChange = new int[0];
int[] arrayWordPresentation = new int[0];
float displacementTargetX;
float displacementTargetY;
float displacementTravelX;
float displacementTravelY;
float displacementCurrentTravelX;
float displacementCurrentTravelY;
float displacementTargetTimeInitial;
float displacementTargetTime;
float measurementCounter = 0;
float currentDisplacement = 0;
float currentDifferential = 0;
float currentDrag = 0;
float averageDisplacement = 0;
float averageDifferential = 0;
int currentDisplacementMillis = 0;
int currentDisplacementMillis2 = 0;
int directionChangeStablizationCounter = 0;
int wordPresentationStablizationCounter = 0;
int directionChangeStablization = 0;
int wordPresentationStablization = 0;
float averageReactionTime = 0;
float[] arrayDirectionChangeStartMillis = new float[0];
float[] arrayDirectionChangeDuration = new float[0];
int[] arrayDirectionChangeStabilized = new int[0];
float[] arrayWordPresentationStartMillis = new float[0];
float[] arrayWordPresentationDuration = new float[0];
int[] arrayWordPresentationStabilized = new int[0];
int saveBlockCounter = 0;
String[] saveBlock = new String[15];

////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

//This function runs once at the beginning of the program. It sets up the screen size and fonts. It selects the first target for the drag force. It adds the titles to each of the rows that will be saved later and then prepares both the audio player, bluetooth and joystick libraries.

```
void setup()
{
 frameRate(30);
 colourMode(RGB);
 size(1024, 768);
 font = createFont("Georgia", 40);
 fontSmall = createFont("Georgia", 20);
 textAlign(CENTER);
 newDestination();
 noCursor();
 saveBlock[0] = "Displacement,";
 saveBlock[1] = "Speed Differential,";
 saveBlock[2] = "Drag Speed,";
 saveBlock[3] = "User Speed,";
 saveBlock[4] = "Milliseconds,";
 saveBlock[5] = "Direction Change,";
 saveBlock[6] = "Word Presentation,";
 saveBlock[7] = "";
 saveBlock[8] = "Direction Change Start Time,";
 saveBlock[9] = "Direction Change Duration,";
 saveBlock[10] = "Direction Change Stabilized,";
 saveBlock[11] = "";
 saveBlock[12] = "Word Presentation Start Time,";
 saveBlock[13] = "Word Presentation Duration,";
 saveBlock[14] = "Word Presentation Stabilized,";
 beeper = new Minim(this);
 beep = beeper.loadFile("beep1.mp3");
 try {
  bt = new Bluetooth(this);
  bt.start("bluetoothServer");
 }
 catch (RuntimeException e) {
  println("bluetooth device is not available.");
  println(e);
 }
 papplet = this;

 //Joystick Setup
 controll = ControllIO.getInstance(this);

 device = controll.getDevice("Saitek ST90 USB Stick");
 device.setTolerance(0.05f);

 ControllSlider sliderX = device.getSlider(1);
 ControllSlider sliderY = device.getSlider(0);
```

```
 stick = new ControllStick(sliderX,sliderY);

 button = device.getButton(0);
 //Joystick Setup End
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function runs once before the experiment proper begins. It sets the times for word displays during the experiment. Each word is given an 8 second window in which it can occur with a uniform distribution. Windows only appear every 12 seconds meaning word presentations will be a minimum of 4 seconds apart. The first window appears after 30 seconds so participants have time to stabilize their responding before words begin to appear.
```
void createDisplayMillisArray()
{
  for (int i=0; i < 20; i = i + 1)
  {
    displayMillisArray[i] = startMillis + 30*1000 + i*12*1000 + floor(random(8*1000));
  }

 displayMillisArray[20] = 99999999;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function runs every frame and control the bulk of the experiment. Due to its size, comments will be fragmented through the function.
```
void draw()
{
```
//This section daws a grey background and displays connection debug information in bottom center of the screen in a light grey colour.
```
 background(128,128,128);

 fill(170,170,170);
 textFont(fontSmall,20);
 if (currentClientName.equals("") == false)
 {
 text(currentClientName + " is connected.", width/2, height - 50);
 }
```
//stepState controls which screen the program is on. Screens 5 and 6 (the waiting and experiment screens) allow the participant to move the joystick and this section of code does that movement. The formula rescales the square co-ordinates received by the joystick into circular ones, meaning the same force can be applied in each direction.
```
 if (stepState == 5 || stepState == 6)
 {
   joyTravelX = joySpeed*stick.getX()*sqrt(1-(sq(stick.getY())/2));
   joyTravelY = joySpeed*stick.getY()*sqrt(1-(sq(stick.getX())/2));
   joyMouseX = joyMouseX + joyTravelX;
   joyMouseY = joyMouseY + joyTravelY;
```

```
 }

//stepState controls which screen the program is on. This switch chooses which screen to display, starting with case 1.
 switch(stepState)
 {
 case 1:   //ready
//Screen 1 is the title screen. It is displays limited information and asks the participant to click through.
 fill(220,220,220);
 textFont(font,40);
 text("Compensatory Tracking", 512, 386);
 textFont(fontSmall,20);
 text("(Press enter or click to continue)", 512, 600);
 break;
 case 2:   //consent
//Screen 2 displays a page of text containing the consent information for the experiment.
 fill(220,220,220);
 textFont(fontSmall,20);
 text("Consent \n" +
     "\n" +
     "You are being asked to participate in a research study of divided attention. \n" +
     "Your participation will involve two or more training sessions with a compensatory tracking task \n" +
     "followed by an experimental condition involving the same tracking task and a word memory task.\n" +
     "\n" +
     "Participation in this research project is voluntary. You have the right to say no \n" +
     "or change your mind and withdraw your participation at any time with no penalty.\n" +
     "\n" +
     "If you have any concerns or questions about your role or rights as a participant, \n" +
     "contact either Matt Ward (Primary Researcher) or Deak Helton (Supervisor).\n" +
     "\n" +
     "By clicking to continue you agree to participate in this research and consent to the data \n" +
     "collected being used for research purposes. No identifying information will be published.\n", 512, 140);
 textFont(fontSmall,20);
 text("(Press enter or click to continue)", 512, 600);
 break;
 case 3:   //save entry
//Screen 3 asks the participant to enter some ID which will be used to separate their results during analysis.
 fill(220,220,220);
 textFont(font,40);
 text("Please enter a unique identifier", 512, 220);
 textFont(fontSmall,20);
 text("(Must be at least 5 characters long)", 512, 280);
 textFont(font,40);
```

```
 switch (typedText.length())
  {
    case 0:
    text("_____", 512, 386);
    break;
    case 1:
    text(typedText + "____", 512, 386);
    break;
    case 2:
    text(typedText + "___", 512, 386);
    break;
    case 3:
    text(typedText + "__", 512, 386);
    break;
    case 4:
    text(typedText + "_", 512, 386);
    break;
    default:
    text(typedText, 512, 386);
  }
 textFont(fontSmall,20);
 text("(Press enter or click to continue)", 512, 600);
 break;
 case 4:   //information
```

//Screen 4 displays written instructions on how the experiment will be run, and what will be expected of the participants.

```
 fill(220,220,220);
 textFont(fontSmall,20);
 text("Compensatory Tracking\n\n"
   + "On the next screen you will see a red target in the center of the screen.\n"
   + "Your task is to keep the cursor (the white circle) as close to the center of this target as
possible.\n\n"
   + "In this condition, words will be presented on the google glass at random intervals.\n"
   + "Read the words out loud as you see them and remember as many as possible.\n"
   + "In this condition a beep will prceed the presentation of each word.\n\n"
   + "Please check that the Google glass has connected before proceeding.\n"
   + "Each task will take 5 minutes and conclude automatically.\n"
   , 512, 200);
 textFont(fontSmall,20);
 text("(Press enter or click to continue)", 512, 600);
 break;
 case 5:   //centering
```

//Screen 5 is a waiting screen on which the participant can become familiar with the sensitivity of the joystick. They have the ability to control the cursor on this screen and they cannot progress until the move it successfully to the center of the target.

```
 fill(220,220,220);
 textFont(fontSmall,20);
```

```
 text("Please center your mouse on the red target before starting the test\nThe test will begin
immediately", 512, 200);
 textFont(fontSmall,20);
 text("(Press enter or click to continue)", 512, 600);
 break;
 case 6:   //play
```
//Screen 6 is the experiment proper. Again, this section will have fragmented comments for added clarity.
```
 //word display trigger
```
//This sections uses the times generated by the previous function to determine if a word should be displayed. If so, the current drag target is cancelled and participants are given an opportunity to recenter themselves while a countdown to a presentation begins. A warning sound, if in the appropriate condition, plays. Then, once one second has elapsed, the word is displayed.
```
   if (millis() > displayMillisArray[displayWordCounter])
   {
    displayCountdown = 31+wordDisplayTime;
    displayWordCounter = displayWordCounter + 1;
    if (displayDirectionChange[displayWordCounter-1] == 1)
     {
       displacementTargetTime = 31;
     }
    beep.play();
    beep.rewind();
   }

   if (displayCountdown > 0)
   {
    displayCountdown = displayCountdown - 1;
    if (displayCountdown == wordDisplayTime)
    {
     displayWord();
    }
    if (displayCountdown == 0)
    {
     currentClient.write(0);
    }
   }
 //word display trigger end
```
//This section controls direction changes and drag force movements. Several values are calculated and held for printing.
```
 displacementTargetTime = displacementTargetTime - 1;
 measurementCounter = measurementCounter + 1;

 if(displacementTargetTime<1)
 {
  newDestination();
 }
```

```
  displacementCurrentTravelX =
displacementTravelX*(min(abs(displacementTargetTime),abs(displacementTargetTime-
displacementTargetTimeInitial),30)/30);
  displacementCurrentTravelY =
displacementTravelY*(min(abs(displacementTargetTime),abs(displacementTargetTime-
displacementTargetTimeInitial),30)/30);
  displacementX = displacementX + displacementCurrentTravelX;
  displacementY = displacementY + displacementCurrentTravelY;

  currentDisplacement = int(floor(sqrt(sq(joyMouseX + displacementX - 512)+sq(joyMouseY +
displacementY - 384))));
  currentDifferential =
int(floor(10*sqrt(sq(joyTravelX+displacementCurrentTravelX)+sq(joyTravelY+displacementCurrentTr
avelY))));
  currentDrag =
int(floor(10*sqrt(sq(displacementCurrentTravelX)+sq(displacementCurrentTravelY))));
```

//This sections saves data to the print arrays to be added to the output file when the experiment is completed.

```
  if (measurementCounter%6 == 0)
    {
     arrayDisplacement = append(arrayDisplacement,currentDisplacement);
     arraySpeedDifferential = append(arraySpeedDifferential,currentDifferential);
     arrayDragSpeed = append(arrayDragSpeed,currentDrag);
     arrayUserSpeed = append(arrayUserSpeed,int(floor(10*sqrt(sq(joyTravelX)+sq(joyTravelY)))));
     arrayMillis = append(arrayMillis, millis() - startMillis);
     if (directionChangeHappened == 1)
     {
       arrayDirectionChange = append(arrayDirectionChange,1);
       directionChangeHappened = 0;
     }
     else
     {
       arrayDirectionChange = append(arrayDirectionChange,0);
     }
     if (wordPresentationHappened == 1)
     {
       arrayWordPresentation = append(arrayWordPresentation,1);
       wordPresentationHappened = 0;
     }
     else
     {
       arrayWordPresentation = append(arrayWordPresentation,0);
     }
    }
  averageDisplacement = ((measurementCounter-1)/measurementCounter)*averageDisplacement +
(1/measurementCounter)*currentDisplacement;
```

```
 averageDifferential = ((measurementCounter-1)/measurementCounter)*averageDifferential +
(1/measurementCounter)*currentDifferential;
 if (directionChangeStablization == 1)
  {
   if (currentDisplacement < max(averageDisplacement,15)
    && currentDifferential < 0.8*currentDrag
    && (joyTravelX + joyTravelY) != 0)
   {
    directionChangeStablizationCounter = directionChangeStablizationCounter + 1;
   }
   else
   {
    directionChangeStablizationCounter = 0;
   }
   if (directionChangeStablizationCounter >= 6)
   {
    arrayDirectionChangeDuration = append(arrayDirectionChangeDuration, millis() -
currentDisplacementMillis - startMillis);
    arrayDirectionChangeStabilized = append(arrayDirectionChangeStabilized, 1);
    directionChangeStablization = 0;
   }
  }
```

// wordPresentationStablization is an internal measure of whether the participants had taken control over the new direction change. It was not used in the final analysis due to large variations in how well participants could hold the cursor steady against the drag force.

```
 if (wordPresentationStablization == 1)
  {
   if (currentDisplacement < max(averageDisplacement,15)
    && currentDifferential < 0.8*currentDrag
    && (joyTravelX + joyTravelY) != 0)
   {
    wordPresentationStablizationCounter = wordPresentationStablizationCounter + 1;
   }
   else
   {
    wordPresentationStablizationCounter = 0;
   }
   if (wordPresentationStablizationCounter >= 6)
   {
    arrayWordPresentationDuration = append(arrayWordPresentationDuration, millis() -
currentDisplacementMillis2 - startMillis);
    arrayWordPresentationStabilized = append(arrayWordPresentationStabilized, 1);
    wordPresentationStablization = 0;
   }
  }
```

//This section ended the session at 5 minutes. At this point the data is printed to a .csv file

```
 if (millis() > startMillis + 5*60*1000)
 {
  if (directionChangeStablization == 1)
  {
   arrayDirectionChangeDuration = append(arrayDirectionChangeDuration, millis() -
currentDisplacementMillis - startMillis);
   arrayDirectionChangeStabilized = append(arrayDirectionChangeStabilized, 0);
  }
  if (wordPresentationStablization == 1)
  {
   arrayWordPresentationDuration = append(arrayWordPresentationDuration, millis() -
currentDisplacementMillis2 - startMillis);
   arrayWordPresentationStabilized = append(arrayWordPresentationStabilized, 0);
  }
  printData();
  stepState = 7;
 }
 break;
 case 7:   //debrief
```
//Screen 7 was an end screen that asked for the participant to wait for the experimenter to set up the next condition
```
 fill(220,220,220);
 textFont(fontSmall,20);
 text("Debrief\n\nThank you for participating in this research project.\nYou have completed a test
component and your data has been saved.\n\nPlease wait for the test supervisor to set up the next
task.\n\nAverage Displacement: " + floor(averageDisplacement*100)/100 + "\n Average Reaction
Time: " + float(floor(averageReactionTime/10))/100  + " seconds", 512, 200);
 textFont(fontSmall,20);
 text("(Press enter or click to close the program)", 512, 600);
 break;
 default :
 }
```
//This section draws the target and cursor during the appropriate screens (the waiting screen and the experiment screen)
```
 if (stepState == 5 || stepState == 6)
 {
  stroke(0,0,0,160);
  fill(255,255,255,40);
  strokeWeight(1);
  ellipse(512,384,200,200);
  fill(255,0,0,60);
  strokeWeight(2);
  ellipse(512,384,160,160);
  fill(255,0,0,70);
  strokeWeight(3);
  ellipse(512,384,120,120);
  fill(255,0,0,80);
```

```
  strokeWeight(4);
  ellipse(512,384,80,80);
  fill(255,0,0,200);
  strokeWeight(5);
  ellipse(512,384,40,40);

  stroke(0,0,0,255);
  strokeWeight(2);
  fill(255,255,255,255);
  ellipse(joyMouseX + displacementX,joyMouseY + displacementY,20,20);
 }
```

//This section detects the trigger on the joystick. On most screens, this just moves to the next screen. It is disabled during the experiment screen.

```
 if(button.pressed())
 {
  if(joyClick == 0)
  {
   joyClick = 1;
  }
 }
 else
 {
  if(joyClick == 2)
  {
   joyClick = 0;
  }
 }

 if (joyClick == 1)
 {
  joyClick = 2;
  switch(stepState)
  {
   case 1:   //ready
   stepState = 2;
   break;
   case 2:   //consent
   stepState = 3;
   break;
   case 3:   //save entry
    if (typedText.length() > 4) {
     stepState = 4;
     typedText += ".csv";
    }
   break;
   case 4:   //information
   stepState = 5;
```

```
    break;
    case 5:   //centering
     if (int(floor(sqrt(sq(joyMouseX + displacementX - 512)+sq(joyMouseY + displacementY - 384)))) <
80 || 1==1) //REMOVE
     {
       stepState = 6;
       startMillis = millis();
       createDisplayMillisArray();
     }
    break;
    case 6:   //play

    break;
    case 7:   //debrief
       exit();
    break;
    default :
  }
 }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
//This function is called when a word needs to be displayed. It sends a message via Bluetooth,
instructing the Glass as to which word to display from which set. Prepares some data for recording.
void displayWord()
{
 currentClient.write(displayWordList*21+displayWordCounter);
 if (displayDirectionChange[displayWordCounter-1] == 1)
 {
  if (directionChangeStablization == 1)
  {
   arrayDirectionChangeDuration = append(arrayDirectionChangeDuration, millis() -
currentDisplacementMillis - startMillis);
   arrayDirectionChangeStabilized = append(arrayDirectionChangeStabilized, 0);
  }

  displacementTargetTime = 30 + 6*30 + float(floor(random(3*30)));
  displacementTargetTimeInitial = displacementTargetTime;
  displacementTravelX = random(displacementSpeed/10);
  displacementTravelY = sqrt(sq(displacementSpeed/10)-sq(displacementTravelX));

  directionChangeHappened = 1;

  currentDisplacementMillis = millis() - startMillis;
  arrayDirectionChangeStartMillis = append(arrayDirectionChangeStartMillis, millis() - startMillis);
  directionChangeStablization = 1;
  directionChangeStablizationCounter = 0;
 }
```

```
 if (wordPresentationStablization == 1)
 {
  arrayWordPresentationDuration = append(arrayWordPresentationDuration, millis() -
currentDisplacementMillis2 - startMillis);
  arrayWordPresentationStabilized = append(arrayWordPresentationStabilized, 0);
 }
 wordPresentationHappened = 1;

 currentDisplacementMillis2 = millis() - startMillis;
 arrayWordPresentationStartMillis = append(arrayWordPresentationStartMillis, millis() - startMillis);
 wordPresentationStablization = 1;
 wordPresentationStablizationCounter = 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function is called when the momentum of the cursor needs to change. It creates a new drag force by choosing a location on the screen (so participants can't lose the cursor during the experiment on particularly long responses) and then calculating the angle and time needed to travel there at the preset drag speed. Records that a direction change occurred.
```
void newDestination()
{
 if (directionChangeStablization == 1)
 {
  arrayDirectionChangeDuration = append(arrayDirectionChangeDuration, millis() -
currentDisplacementMillis - startMillis);
  arrayDirectionChangeStabilized = append(arrayDirectionChangeStabilized, 0);
 }

 displacementTargetX = floor(200 + random(624)-512);
 displacementTargetY = floor(200 + random(368)-384);

 displacementTargetTime = 30 + floor(sqrt(sq(displacementX-
displacementTargetX)+sq(displacementY-displacementTargetY))/(displacementSpeed/10));
 displacementTargetTimeInitial = displacementTargetTime;
 displacementTravelX = ((displacementTargetX - displacementX)/(displacementTargetTime-29));
 displacementTravelY = ((displacementTargetY - displacementY)/(displacementTargetTime-29));

 directionChangeHappened = 1;

 currentDisplacementMillis = millis() - startMillis;
 arrayDirectionChangeStartMillis = append(arrayDirectionChangeStartMillis, millis() - startMillis);
 directionChangeStablization = 1;
 directionChangeStablizationCounter = 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function is called once after the experiment screen. It saves all of the gathered data to a formatted .csv file for analysis.
```
void printData()
{
```

```
for (int i = 0; i < arrayDisplacement.length; i++)
 {
  saveBlock[0] = saveBlock[0] + arrayDisplacement[i] + ",";
 }
 for (int i = 0; i < arraySpeedDifferential.length; i++)
 {
  saveBlock[1] = saveBlock[1] + arraySpeedDifferential[i] + ",";
 }
 for (int i = 0; i < arrayDragSpeed.length; i++)
 {
  saveBlock[2] = saveBlock[2] + arrayDragSpeed[i] + ",";
 }
 for (int i = 0; i < arrayUserSpeed.length; i++)
 {
  saveBlock[3] = saveBlock[3] + arrayUserSpeed[i] + ",";
 }
 for (int i = 0; i < arrayMillis.length; i++)
 {
  saveBlock[4] = saveBlock[4] + arrayMillis[i] + ",";
 }
 for (int i = 0; i < arrayDirectionChange.length; i++)
 {
  saveBlock[5] = saveBlock[5] + arrayDirectionChange[i] + ",";
 }
 for (int i = 0; i < arrayWordPresentation.length; i++)
 {
  saveBlock[6] = saveBlock[6] + arrayWordPresentation[i] + ",";
 }
 for (int i = 0; i < arrayDirectionChangeStartMillis.length; i++)
 {
  saveBlock[8] = saveBlock[8] + arrayDirectionChangeStartMillis[i] + ",";
 }
 for (int i = 0; i < arrayDirectionChangeDuration.length; i++)
 {
  saveBlock[9] = saveBlock[9] + arrayDirectionChangeDuration[i] + ",";
  averageReactionTime = averageReactionTime+ arrayDirectionChangeDuration[i];
 }
 for (int i = 0; i < arrayDirectionChangeStabilized.length; i++)
 {
  saveBlock[10] = saveBlock[10] + arrayDirectionChangeStabilized[i] + ",";
 }
 for (int i = 0; i < arrayWordPresentationStartMillis.length; i++)
 {
  saveBlock[12] = saveBlock[12] + arrayWordPresentationStartMillis[i] + ",";
 }
 for (int i = 0; i < arrayWordPresentationDuration.length; i++)
 {
  saveBlock[13] = saveBlock[13] + arrayWordPresentationDuration[i] + ",";
```

```
  }
  for (int i = 0; i < arrayWordPresentationStabilized.length; i++)
  {
   saveBlock[14] = saveBlock[14] + arrayWordPresentationStabilized[i] + ",";
  }
  saveBlock[7] = averageDisplacement + "," + averageDifferential;
  saveStrings(typedText,saveBlock);
  averageReactionTime = averageReactionTime/arrayDirectionChangeDuration.length;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function detects key presses. Enables the entry of an ID on screen 3. On information screens, any key will progress to the next screen. During the experiment, the 'M' key will terminate and save the session early, if necessary.

```
void keyReleased() {
 switch(stepState)
 {
 case 1:   //ready
 stepState = 2;
 break;
 case 2:   //consent
 stepState = 3;
 break;
 case 3:   //save entry
 if (key != CODED) {
  switch(key) {
  case BACKSPACE:
   typedText = typedText.substring(0,max(0,typedText.length()-1));
   break;
  case TAB:
   typedText += "    ";
   break;
  case ENTER:
  case RETURN:
  if (typedText.length() > 4) {
   stepState = 4;
   typedText += ".csv";
  }
   break;
  case ESC:
  case DELETE:
   break;
  default:
   typedText += key;
  }
 }
 break;
 case 4:   //information
 stepState = 5;
```

```
 break;
 case 5:   //centering
  if (int(floor(sqrt(sq(joyMouseX + displacementX - 512)+sq(joyMouseY + displacementY - 384)))) <
80)
  {
   stepState = 6;
   startMillis = millis();
   createDisplayMillisArray();
  }
 break;
 case 6:   //play
  if (key == 'm') {
   printData();
   stepState = 7;
  }
 break;
 case 7:   //debrief
  exit();
 break;
 default :
 }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
//This function shuts down the Bluetooth when the program is closed.
void destroy()
{
 bt.stop();
}
////////////////////////////////////////////////////////////////////////////////////////////////////
//This function detects a connection from a Bluetooth device and save the device ID to send
messages to later for word displays.
void clientConnectEvent(Client client)
{
  currentClientName = client.device.name;
  currentClient = client;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
```

Appendix 1.2a: Data Output File Example

The first block of values displays measures relating to the position and forces acting on the cursor at each update. Displacement measures how far away from the center of the target the cursor is in the moment, measured in pixels. Speed Differential is the net velocity of the cursor, combining both user input and the drag force. Drag speed is the velocity due to the drag force. User speed is the velocity due to user input through the joystick. Milliseconds is the time elapsed since the experiment phase began, in milliseconds. Direction Change and Word Presentation are binary variables indicating if an event of the appropriate time has occurred since the last measurement.

The second and third blocks display information about the incidence of direction changes and word presentations respectively. Stabilisation time, the time it took participants to return the cursor to within a range less than their average displacement turned out to be an unreliable measure and was not used in the analysis. Instead, the displacement curve for each event was extracted instead and used to calculate when participants reached peak displacement following an event. This point is when participant's user input is counteracting the velocity added by the drag force. This requires the participant to be adding their own force within a 90-degree arc counter to the drag force, indicating that the participant has detected the change in drag force and has made a compensatory reaction time. This measure, labelled Reaction Time, is calculated by the Microsoft Excel macros found in Appendix 1.3.

Displacement	3	2	5	4	12	10	3	…
Speed Differential	2	5	8	18	16	15	15	…
Drag Speed	2	5	8	11	14	14	14	…
User Speed	0	0	0	30	30	6	8	…
Milliseconds	193	392	592	793	993	1193	1393	…
Direction Change	1	0	0	0	0	0	0	…
Word Presentation	0	0	0	0	0	0	0	…
Direction Change Start Time	390	7093	9859	20926	30559	40393	43659	…
Direction Change Duration	1404	2766	1000	834	2167	766	3334	…
Direction Change Stabilized	1	0	1	1	1	1	1	…
Word Presentation Start Time	16326	30559	43659	54259	66392	81159	93492	…
Word Presentation Duration	167	2167	3334	599	2600	2666	167	…
Word Presentation Stabilized	1	1	1	1	1	1	1	…

Appendix 1.3: Experiment 1 Analysis Macros

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//This function sets up some headings, sets the cell ranges for later calculations, and calculates the Root Mean Squared Errors (RMSE).
Sub AA_SetUp()

Dim i As Long
Dim tLong1 As Long

Cells(18, 1).Value = "Error Codes:"
Cells(17, 1).Value = 0
Cells(17, 3).Value = "Parameters"
Cells(18, 3).Value = "Measurement Range"
Cells(19, 3).Value = Application.WorksheetFunction.CountA(Range(Cells(1, 2), Cells(1, 2000))) + 1
Cells(18, 4).Value = "Direction Change Count"
Cells(19, 4).Value = 0
Cells(18, 5).Value = "Word Presentation Count"
Cells(19, 5).Value = 0
```

```
Cells(18, 6).Value = "Paired Presentation Count"
Cells(19, 6).Value = 0
Cells(18, 7).Value = "RMSE"

tLong1 = 0
For i = 32 To Cells(19, 3).Value
   tLong1 = tLong1 + Cells(1, i).Value ^ 2
Next i
Cells(19, 7).Value = (tLong1 / (Cells(19, 3).Value - 1)) ^ 0.5

Cells(18, 8).Value = "Stdev.S"
Cells(19, 8).Value = "=Stdev.S(B1:AZZ1)"

End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```
//This function prints out the displacement values from the preceding 2 seconds before an event through to 4 seconds after an event. Separates the events passed on their type (word presentation alone, direction change alone and paired presentations). This is used to calculate the average displacement curve for each type of event which are used in the next stage.
```
Sub AB_PullCurves()

Dim i As Long
Dim j As Long
Dim printRow As Long
Dim AverageArray(-10 To 19) As Long

printRow = 0
Cells(21 + printRow, 3).Value = "Raw Post-Direction Change Curves"
printRow = printRow + 1
Cells(21 + printRow, 8).Value = "Pre-DC Slowing Begins"
Cells(21 + printRow, 13).Value = "Event occurs"
printRow = printRow + 1

For i = -10 To 19

   AverageArray(i) = 0

Next i

For i = 32 To Cells(19, 3).Value

   If (Cells(6, i).Value = 1) Then
   If (Cells(7, i).Value = 0) Then

      Cells(19, 4).Value = Cells(19, 4).Value + 1

      For j = -10 To 19
```

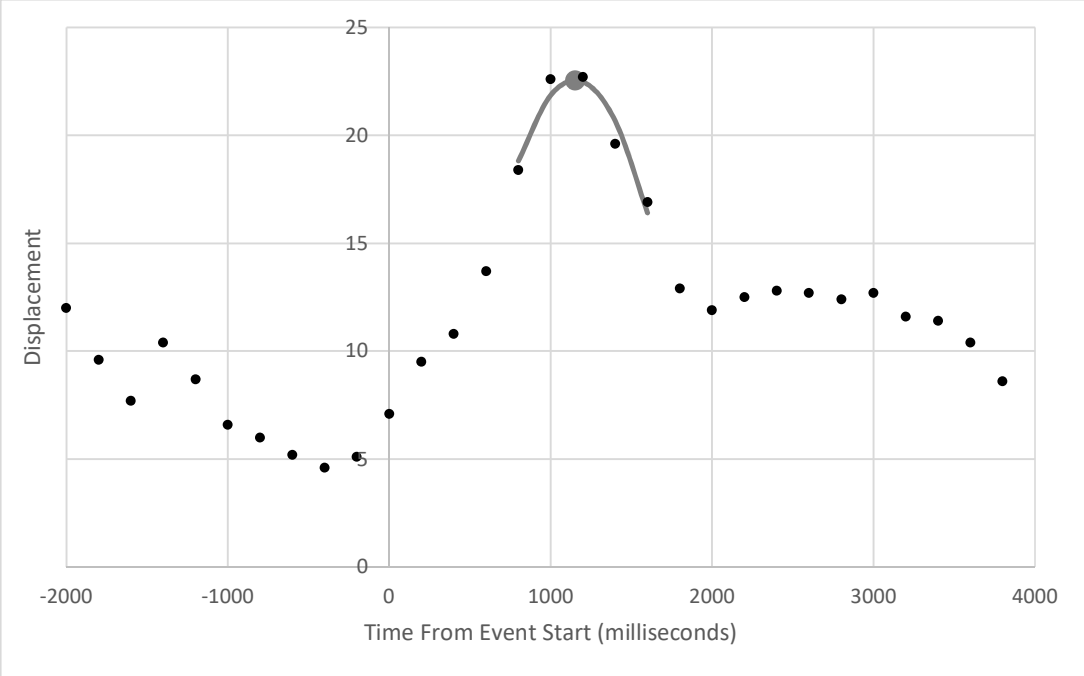
```
            Cells(21 + printRow, 13 + j).Value = Cells(1, i + j).Value
            AverageArray(j) = AverageArray(j) + Cells(1, i + j).Value

        Next j

        printRow = printRow + 1

    End If
    End If

Next i
printRow = printRow + 1
Cells(18, 9).Value = "DC Change Average Row"
Cells(19, 9).Value = 21 + printRow
For i = -10 To 19

    Cells(21 + printRow, 13 + i).Value = AverageArray(i) / Cells(19, 4).Value

Next i
printRow = printRow + 1
printRow = printRow + 1

Cells(21 + printRow, 3).Value = "Raw Post-Word Presentation Curves"
printRow = printRow + 1
Cells(21 + printRow, 13).Value = "Event occurs"
printRow = printRow + 1

For i = -10 To 19

    AverageArray(i) = 0

Next i

For i = 32 To Cells(19, 3).Value

    If (Cells(7, i).Value = 1) Then
    If (Cells(6, i).Value = 0) Then

        Cells(19, 5).Value = Cells(19, 5).Value + 1

        For j = -10 To 19

            Cells(21 + printRow, 13 + j).Value = Cells(1, i + j).Value
            AverageArray(j) = AverageArray(j) + Cells(1, i + j).Value

        Next j

        printRow = printRow + 1
```

```
   End If
   End If

Next i
printRow = printRow + 1
Cells(18, 10).Value = "WP Change Average Row"
Cells(19, 10).Value = 21 + printRow
For i = -10 To 19

   Cells(21 + printRow, 13 + i).Value = AverageArray(i) / Cells(19, 5).Value

Next i
printRow = printRow + 1
printRow = printRow + 1

Cells(21 + printRow, 3).Value = "Raw Post-Paired Presentation Curves"
printRow = printRow + 1
Cells(21 + printRow, 8).Value = "Pre-DC Slowing Begins"
Cells(21 + printRow, 13).Value = "Event occurs"
printRow = printRow + 1

For i = -10 To 19

   AverageArray(i) = 0

Next i

For i = 32 To Cells(19, 3).Value

   If (Cells(6, i).Value = 1) Then
   If (Cells(7, i).Value = 1) Then

      Cells(19, 6).Value = Cells(19, 6).Value + 1

      For j = -10 To 19

         Cells(21 + printRow, 13 + j).Value = Cells(1, i + j).Value
         AverageArray(j) = AverageArray(j) + Cells(1, i + j).Value

      Next j

      printRow = printRow + 1

   End If
   End If

Next i
```

```
printRow = printRow + 1
Cells(18, 11).Value = "PP Change Average Row"
Cells(19, 11).Value = 21 + printRow
For i = -10 To 19
   Cells(21 + printRow, 13 + i).Value = AverageArray(i) / Cells(19, 6).Value
Next i

End Sub
```

//

To get each participant's average response time the final step was to fit a quadratic, polynomial function in a range around the peak of the of the response curve using the excel solver function. The maximum of the curve was used as the average Response Time for each participant for each type of event. Below is an example of the fitting procedure. Data comes from participant 1's tracking task alone condition, paired presentation events. Black dots are the average displacement curve. the grey quadratic curve is the fitted peak. The large grey dot is the maximum of the curve.



Appendix 1.4: Experiment 1 Materials

Two printed pages were used during the experiment. The first was held by the experimenter and was used to mark when a participant read aloud a word. The second page was presented to the participant immediately after the experiment and was used to report which words they are able to remember. These pages were scored by hand.

ID _____

ABc

"slipper",
"glacier",
"pepper",
"settler",
"odour",
"dreamer",
"cigar",
"captive",
"banner",
"poster",
"arrow",
"piston",
"pencil",
"hardwood",
"beggar",
"invoice",
"robber",
"elbow",
"butcher",
"salute"

AB

"candy",
"reptile",
"comrade",
"sunset",
"shotgun",
"profile",
"scarlet",
"harness",
"bandit",
"portrait",
"barrel",
"sunburn",
"warbler",
"lobster",
"rubble",
"bouquet",
"hostage",
"cellar",
"tweezers",
"abyss"

Bc

"fabric",
"mucus",
"hurdle",
"leopard",
"pianist",
"footwear",
"cuisine",
"hammer",
"trellis",
"garments",
"nectar",
"sultan",
"trumpet",
"doorman",
"hillside",
"pudding",
"kettle",
"daybreak",
"speaker",
"portal"

B

"costume",
"saloon",
"kerchief",
"python",
"fiord",
"bullet",
"mammal",
"salad",
"lemon",
"twilight",
"abode",
"jelly",
"singer",
"goblet",
"painter",
"wigwam",
"typhoon",
"basement",
"sulphur",
"juggler"

ID _____

ABc

AB

Bc

B

Appendix 2.0: Programming and Data Analysis for Experiment 2

Experiment 2 shared an experimental design with experiment 1 and therefore used the same programs with only minor adjustments. The University of Canterbury Human Ethic Approval for this experiment is presented in Appendix 2.1. Four paper materials were used during Experiment. These are available in Appendix 2.2. Added to the experiment were a paper copy of the experiment information sheet and consent form. A new word set was generated for the experiment and an updated word list page is available.

Appendix 2.1: Human Ethics Approval for Experiment 2

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE

Secretary, Lynda Griffioen
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2015/02/LR-PS

2 February 2015

Matthew Ward
Department of Psychology
UNIVERSITY OF CANTERBURY

Dear Matthew

Thank you for forwarding to the Human Ethics Committee a copy of the low risk application you have recently made for your research proposal "The effect of delivery mode on the dual-task interference of receiving information from a Google Glass".

I am pleased to advise that this application has been reviewed and I confirm support of the Department's approval for this project.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 29 January 2015.

With best wishes for your project.

Yours sincerely

Lindsey MacDonald
Chair, Human Ethics Committee

Appendix 2.2: Paper Materials for Experiment 2

Information sheet for Experiment 2

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

The Effect of Delivery Mode on the Dual-Task Interference of Receiving Information from a Google Glass.

Information Sheet for participants.

You are being asked to participate in human factors research as part of Matt Ward's doctoral research at the University of Canterbury. This study, 'The Effect of Delivery Mode on the Dual-Task Interference of Receiving Information from a Google Glass', is investigating how well people can read information from a Google Glass. Your involvement in this project will be performing a simple tracking task on a desktop computer while receiving, reading aloud, and memorising messages delivered by a Google Glass unit that you will be wearing. Data will be recorded automatically and by an observing researcher. Participation will take approximately 40 minutes. At the conclusion of the test you will be compensated for your time with a $10 petrol voucher.

There are no expected risks to participants' physical or mental health, although if the Google Glass becomes uncomfortable for any reason, you may request for the researcher to stop the test without penalty.

You may receive a copy of the project results by contacting Matt Ward at the conclusion of the project.

Participation is voluntary and you have the right to withdraw at any stage without penalty. If you withdraw during the test, all information relating to you will be destroyed. To ensure anonymity and confidentiality, all records of your name will be replaced with a randomly assigned participant number and no other identifying data will be stored. It will therefore be impossible to remove your data from the pool after it has been collected. Paper copies of any testing materials will also use your randomly assigned participant number. A consent form bearing your name will be stored for a period of ten years in a locked filing cabinet and will only be made available to the primary researcher Matt Ward, and his senior supervisor, Dr Deak Helton, as well as the University of Canterbury Human Ethics Committee if such disclosure becomes necessary.

The results of the project will be used in a doctoral thesis and may also be published. A thesis is a public document and will be available through the UC Library.

This project is being carried out by Matt Ward (matt.ward@pg.canterbury.ac.nz) under the supervision of Dr Deak Helton, who can be contacted at deak.helton@canterbury.ac.nz if you have any concerns about participation in this project.

This project has been reviewed approved by the University of Canterbury Psychology department and the UCHEC Low Risk Approval process. Participants should address any complaints to The Chair, Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz). If you agree to participate in the study, you are asked to complete the consent form before continuing with the test.

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

The Effect of Delivery Mode on the Dual-Task Interference of Receiving Information from a Google Glass.

Consent Form for Participants

I have been given a full explanation of this project, I understand what is required of me if I agree to take part in the research and I have had the opportunity to ask questions.

I understand that participation is voluntary and I may withdraw at any time without penalty. I understand that withdrawal of participation will also include the withdrawal of any information I have provided should this remain practically achievable.

I understand that I can request to stop the test at any time if I find wearing the device to be uncomfortable.

I understand that any information or opinions I provide will be kept confidential to the researcher and their senior supervisor and that any published or reported results will not identify me or the other participants. I understand that a thesis is a public document and will be available through the UC Library.

I understand that all identifying data collected for the study will be kept in locked and secure facilities and will be destroyed after ten years.

I understand that I can contact the researcher Matt Ward (matt.ward@pg.canterbury.ac.nz) or supervisor Dr Deak Helton (deak.helton@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Human Ethics Committee, Private Bag 4800, Christchurch (humanethics@canterbury.ac.nz)

By signing below, I agree to participate in this research project.

Name:

Date:

Signature:

Word list sheet for Experiment 2

ID _____

01

wigwam
settler
weapon
odour
candy
feline
scarlet
prairie
meadow
rubble
prison
metal
blossom
painter
disease
warbler
singer
hammer
maiden
trumpet

11

infant
boulder
coffee
goblet
lemon
circuit
sunburn
bandit
angle
salute
hairpin
cottage
opium
slipper
shotgun
bagpipe
pianist
portal
dweller
leopard

20

vapour
blister
cuisine
cellar
thicket
hostage
venom
mantle
piano
locker
glacier
sulphur
skillet
invoice
author
harness
arrow
artist
steamer
product

21

baron
kettle
piston
salad
pepper
tower
hotel
butcher
exhaust
insect
bottle
doorman
lobster
victim
mucus
nectar
monarch
gingham
builder
banker

Appendix 3.0: Ethics, Programming, Data Analysis and Paper Materials for Experiment 3

The University of Canterbury Human Ethic Approval for this experiment is presented in Appendix 3.1.

Experiment 3's core program was written in the Unity engine (Unity3d.com, 2015) using the C# language. The D-track software (Ar-tracking.com, 2015) provided the tracking data and was transferred to the main program's desktop PC via the VRPN software (GitHub, 2015). See Chapter 3 for the block diagram detailing the links between electronic components (Figure 3.1).

Appendix 3.2 contains the core experiment program as well as specialist functions which controlled the head tracking, gun tracking, and data recording. Also included is a simple VRPN receiver library.

Appendix 3.3 contains the program run by the Recon Jet which was responsible for managing the device's display.

Appendix 3.4 shows example data output files from the experiment and the process used to extract useful data. Macros used in the process are written in the Visual Basic language.

Appendix 3.5 contains the paper materials used in this experiment. These include an information sheet and a consent form.

Appendix 3.1: Human Ethics Approval for Experiment 2

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE

Secretary, Lynda Griffioen
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2015/52/LR-PS

28 September 2015

Matt Ward
Department of Psychology
UNIVERSITY OF CANTERBURY

Dear Matt

Thank you for forwarding your low risk application to the Human Ethics Committee for the research proposal titled "Redirecting attention in real-world space with wearable computers".

I am pleased to advise that this application has been reviewed and approved.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 25 September 2015.

With best wishes for your project.

Yours sincerely

Lindsey MacDonald
Chair, Human Ethics Committee

Appendix 3.2: Main Controller Program for Experiment 3.

//
//The first class is a library for receiving information through the VRPN connection.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;
using UnityEngine;

public class VRPN {

    private const string LIBRARY_NAME = "SimpleVRPN";

    [DllImport(LIBRARY_NAME, CharSet = CharSet.Ansi)]
    public static extern void vrpnOpen(string deviceName);

    [DllImport(LIBRARY_NAME)]
    public static extern void vrpnClose();

    [DllImport(LIBRARY_NAME)]
    public static extern bool vrpnGetTrackedBody(int id, float[] pos, float[] quat);

    [DllImport(LIBRARY_NAME)]
    public static extern bool vrpnGetButton(int id);

    [DllImport(LIBRARY_NAME)]
    public static extern bool vrpnUpdate();

}
```

//
//The second class is the controller for the timing and sequencing of the experiment.

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

//This is the network message which is used to synchronise with the Recon Jet.
public class stateTransfer : MessageBase
{
    public int value;
}

public class TargetPlacer : MonoBehaviour {
    public float xShift;
    public float zShift;
    public int currentPosition;
    public int[] timings = new int[113];
    public int[] soundEvents = new int[45];
    public int[] positionEvents = new int[45];
    public int[] visualEvents = new int[45];
    public int currentTiming = 0;
    public int currentDist = 0;
    public int currentEvent = 0;
    public float tideLevel = 0;
    public int eventStage = 0;
    public int metaStage = 0;
```

```
public HeadTracking headtrack;
public GunTracking guntrack;
public Recording record;
public int displayNumber;
public SpriteRenderer render;
public int forceBlank;
public GUIStyle style;

public TBE_3DCore.TBE_Source AC;
public Transform AudioCueTransform;
public audiocue AudioCueScript;

//string ipAddress = "192.168.1.6";
int port = 25000;
//int maxConnections = 10;

// This function sends an update to the Recon Jet when called.
public void sendState(int state)
{
stateTransfer outgoingMsg = new stateTransfer();
outgoingMsg.value = state;
   NetworkServer.SendToAll (7772, outgoingMsg);
}

void Update () {
    //This is the main screen switch. Each value of 'metaStage' is a different
    screen. It runs every frame of the program.
    switch (metaStage)
    {
        case 0: //A wait screen while the connection is established.
            if (Input.GetMouseButtonDown(1))
            {
                metaStage = 1;
                sendState(3);
            }
            break;
        case 1: //A screen which shows an example of the number sequence on the
            central screen and an example update on the Recon Jet. Checks
            participants can read both strings.
            if (Input.GetMouseButtonDown(1))
            {
                metaStage = 2;
                render.colour = Colour.yellow;
                xShift = (float)-2390;
                zShift = (float)1.731967 * xShift + (float)4226;
                this.transform.position = new Vector3(xShift / 100, 0, zShift /
                100);
                this.transform.rotation = Quaternion.Euler(0, -60, 0);
                sendState(2);
            }
            break;
        case 2: //On these screen an example target is displayed in the far left
        case 3: and far right positions. Participants must shoot these targets to
                progress. Checks the gun calibration and participant's range of
                motion.
            //Do Nothing
            break;
        case 114: //This is a wait screen while the experimenter fixes any issues
                  detected in the precious steps. Steps up 6 test trails with a
                  range of different event types.
            if (Input.GetMouseButtonDown(1))
            {
```

```
            metaStage = 4;
            System.IO.File.AppendAllText(record.fileName1, "Trials" +
            System.Environment.NewLine);
            tideLevel = Time.time;
            currentDist = 1;
            currentTiming = 0;
            eventStage = 0;
            sendState(2);
            soundEvents[0] = 2; visualEvents[0] = 1; positionEvents[0] = 4;
            soundEvents[1] = 0; visualEvents[1] = 2; positionEvents[1] = 1;
            soundEvents[2] = 1; visualEvents[2] = 0; positionEvents[2] = 2;
            soundEvents[3] = 0; visualEvents[3] = 0; positionEvents[3] = 5;
            soundEvents[4] = 2; visualEvents[4] = 2; positionEvents[4] = 2;
            soundEvents[5] = 1; visualEvents[5] = 1; positionEvents[5] = 4;
        }
        break;
    case 4: //This is the screen for the test trials. Will alternate between
              screen and Jet messages until an event occurs.
        if (eventStage == 0)
        {
            if (currentDist == 0)
            {
                if (Time.time > tideLevel + 1.0)
                {
                    targetEvent();
                }

            }
            else
            {
                if (Time.time > tideLevel + 5)
                {
                    tideLevel = Time.time;
                    currentDist = 0;

                    sendState(3);
                    //Send new message to HMD
                }
            }
        }
        else
        {
            if (eventStage == 1)
            {

            }
            else
            {
                render.colour = new Vector4(0F, 0.6F, 1F, 1);


                if (headtrack.projectedX > -400f)
                {
                    if (headtrack.projectedX < 400f)
                    {
                        if (headtrack.projectedY < -200f)
                        {
                            tideLevel = Time.time;
                            currentDist = 1;
                            currentTiming = currentTiming + 1;
                            currentEvent = currentEvent + 1;
                            displayNumber = Random.Range(10000000, 99999999);
```

```
                        eventStage = 0;
                        this.transform.position = new Vector3(0, 0, 25);
                        System.IO.File.AppendAllText(record.fileName1,
                        Time.time * 1000 + ",4," + currentPosition + "," +
                        headtrack.projectedX + "," + headtrack.projectedY
                        + "," + headtrack.projectedZ + "," +
                        guntrack.projectedX + "," + guntrack.projectedY +
                        "," + guntrack.projectedZ +
                        System.Environment.NewLine);

                        sendState(2);
                        //End Display Message to HMD

                        if (currentTiming == 6)
                        {
                            metaStage = 5;
                            sendState(2);
                        }
                    }
                }
            }
        }
    }
    break;
case 5: //A break screen between the practice trials and the main
        experiment. Establishes and randomises the arrays that hold the
        sequence of events types.
    if (Input.GetMouseButtonDown(1))
    {
        metaStage = metaStage + 1;
        System.IO.File.AppendAllText(record.fileName1, "Block " +
        ((metaStage / 2) - 2) + System.Environment.NewLine);
        tideLevel = Time.time;
        currentDist = 1;
        currentTiming = 0;
        currentEvent = 0;
        eventStage = 0;
        forceBlank = 0;
        sendState(2);
        establishArrays();
        randomizeArrays();
    }
    break;
case 7: //The two break screens between the three main trial blocks.
case 9: //re-randomises the event order.
    if (Input.GetMouseButtonDown(1))
    {
        metaStage = metaStage + 1;
        System.IO.File.AppendAllText(record.fileName1, "Block " +
        ((metaStage/2)-2) + System.Environment.NewLine);
        tideLevel = Time.time;
        currentDist = 1;
        currentTiming = 0;
        currentEvent = 0;
        eventStage = 0;
        forceBlank = 0;
        sendState(2);
        randomizeArrays();
    }
    break;
case 6: //The main experiment screens for the three blocks of test trials.
case 8: //Will alternate between messages until an event occurs.
```

```
case 10: //continues presenting messages and recording event responses
        until the block of trials is complete.
    if (eventStage == 0)
    {
        if (currentDist == 0)
        {
            if (forceBlank == 0)
            {
                if (timings[currentTiming] == 1)
                {
                    if (Time.time > tideLevel + 1.0)
                    {
                        targetEvent();
                    }
                }
                else
                {
                    if (Time.time > tideLevel + 5.5)
                    {
                        tideLevel = Time.time;
                        currentDist = 1;
                        currentTiming = currentTiming + 1;
                        displayNumber = Random.Range(10000000, 99999999);

                        sendState(2);
                        //End Display Message to HMD

                    }
                }
            }
            else
            {
                if (Time.time > tideLevel + 5.5)
                {
                    tideLevel = Time.time;
                    currentDist = 1;
                    displayNumber = Random.Range(10000000, 99999999);
                    forceBlank = 0;
                    sendState(2);
                    //End Display Message to HMD

                }
            }
        }
        else
        {
            if (Time.time > tideLevel + 4.5)
            {
                tideLevel = Time.time;
                currentDist = 0;

                sendState(3);
                //Send new message to HMD
            }
        }
    }
    else
    {
        if (eventStage == 1)
        {

        }
```

```
                else
                {
                    render.colour = new Vector4(0F, 0.6F, 1F, 1);

                    if (headtrack.projectedX > -400f)
                    {
                        if (headtrack.projectedX < 400f)
                        {
                            if (headtrack.projectedY < -200f)
                            {
                                tideLevel = Time.time;
                                currentDist = 1;
                                currentTiming = currentTiming + 1;
                                currentEvent = currentEvent + 1;
                                displayNumber = Random.Range(10000000, 99999999);
                                eventStage = 0;
                                forceBlank = 1;
                                this.transform.position = new Vector3(0, 0, 25);
                                System.IO.File.AppendAllText(record.fileName1,
                                Time.time * 1000 + ",4," + currentPosition + "," +
                                headtrack.projectedX + "," + headtrack.projectedY
                                + "," + headtrack.projectedZ + "," +
                                guntrack.projectedX + "," + guntrack.projectedY +
                                "," + guntrack.projectedZ +
                                System.Environment.NewLine);

                                sendState(2);
                                //End Display Message to HMD
                            }
                        }
                    }
                }
            }
            if (currentTiming == 72)
            {
                metaStage = metaStage + 1;
                sendState(2);
            }
            break;
        case 11: //An end screen to let the participant know they have finished.
            if (Input.GetMouseButtonDown(1))
            {
                Application.Quit();
            }
            break;
        }
}

//This function runs alongside the Update function and displays the Graphical
User Interface to the screens. This is used to display the text for each
screen.
void OnGUI()
{

       switch (metaStage)
       {
       case 0: //Initial wait screen
              GUI.Label (new Rect (10, 10, 1004, 748), "Server is active: " +
              NetworkServer.active);
              GUI.Label (new Rect (10, 30, 1004, 748), "Active Connections: " +
              NetworkServer.connections.Count);
              GUI.Label(new Rect(1034, 1, 1004, 748), "Please Wait...", style);
```

```
                break;
            case 1: //Message Test
                GUI.Label(new Rect(1034, 300, 1004, 748), " " + displayNumber + "
                ", style);
                break;
            case 2: //Gun Test 1
                GUI.Label(new Rect(1034, 1, 1004, 748), "Shoot the target to the
                left.", style);
                break;
            case 3: //Gun Test 2
                GUI.Label(new Rect(1034, 1, 1004, 748), "Shoot the target to the
                right.", style);
                break;
            case 4: //Experiment Block Screens
            case 6:
            case 8:
            case 10:
                if (eventStage == 0)
                {
                        if (currentDist == 0)
                        {
                                //GUI.Label(new Rect(1034, 1, 1004, 748), "HMD
                                filler", style);
                        }
                        if (currentDist == 1)
                        {
                                GUI.Label(new Rect(1034, 300, 1004, 748), " " +
                                displayNumber + " ", style);
                        }
                }
                else
                {
                    if (eventStage == 2)
                    {
                       GUI.Label(new Rect(1034, 300, 1004, 748), " Return view to
                       here. ", style);
                    }
                }
                break;
            case 5: //Break Screens
            case 7:
            case 9:

                GUI.Label(new Rect(1034, 1, 1004, 748), "Please take a short
                break...", style);
                break;
            case 114: //Pre-Practice Wait Screen
                GUI.Label(new Rect(1034, 1, 1004, 748), "Please Wait...", style);
                break;
            case 11: //End Escrren
                GUI.Label(new Rect(1034, 1, 1004, 748), "Experiment Complete",
                style);
                break;
            default:
                GUI.Label(new Rect(1034, 1, 1004, 748), "PANIC!", style);
                break;
                }

    GUI.Label(new Rect(10, 738, 1004, 20), "Elapsed Time: " + Time.time);
    }
}
```

```
//This function runs once at the start of the program. It sets up the program to
listen for incoming network connections, chooses the first display number and then
creates a pseudorandom array of event timings. A zero on this array means no event
will occur during the same-numbered Recon Jet message during the block. A one means a
target will appear during the same-numbered message.
       void Start ()
       {
        NetworkServer.Listen(port);

        displayNumber = Random.Range(10000000, 99999999);

              timings[0] = 0;
              timings[1] = 0;
              timings[2] = 0;
              timings[3] = 0;
              timings[4] = 0;
              timings[5] = 0;
              timings[6] = 0;
              timings[7] = 0;
              timings[8] = 0;
              timings[9] = 0;
              timings[10] = 0;
              timings[11] = 0;
              timings[12] = 0;
              timings[13] = 0;
              timings[14] = 0;
              timings[15] = 0;
              timings[16] = 0;
              timings[17] = 0;
              timings[18] = 0;
              timings[19] = 0;
              timings[20] = 0;
              timings[21] = 0;
              timings[22] = 0;
              timings[23] = 0;
              timings[24] = 0;
              timings[25] = 0;
              timings[26] = 0;
              timings[27] = 0;
              timings[28] = 0;
              timings[29] = 0;
              timings[30] = 0;
              timings[31] = 0;
              timings[32] = 0;
              timings[33] = 0;
              timings[34] = 0;
              timings[35] = 0;
              timings[36] = 1;
              timings[37] = 1;
              timings[38] = 1;
              timings[39] = 1;
              timings[40] = 1;
              timings[41] = 1;
              timings[42] = 1;
              timings[43] = 1;
              timings[44] = 1;
              timings[45] = 1;
              timings[46] = 1;
              timings[47] = 1;
              timings[48] = 1;
              timings[49] = 1;
```

```
            timings[50] = 1;
            timings[51] = 1;
            timings[52] = 1;
            timings[53] = 1;
            timings[54] = 1;
            timings[55] = 1;
            timings[56] = 1;
            timings[57] = 1;
            timings[58] = 1;
            timings[59] = 1;
            timings[60] = 1;
            timings[61] = 1;
            timings[62] = 1;
            timings[63] = 1;
            timings[64] = 1;
            timings[65] = 1;
            timings[66] = 1;
            timings[67] = 1;
            timings[68] = 1;
            timings[69] = 1;
            timings[70] = 1;
            timings[71] = 1;

        randomizeArrays();

    }

//This function runs once after the practice trials. It creates a set of paired arrays
which contain all the possible combinations of event types.
    void establishArrays()
    {

        soundEvents[0] = 0; visualEvents[0] = 0; positionEvents[0] = 1;
        soundEvents[1] = 0; visualEvents[1] = 0; positionEvents[1] = 2;
        soundEvents[2] = 0; visualEvents[2] = 0; positionEvents[2] = 4;
        soundEvents[3] = 0; visualEvents[3] = 0; positionEvents[3] = 5;
        soundEvents[4] = 0; visualEvents[4] = 1; positionEvents[4] = 1;
        soundEvents[5] = 0; visualEvents[5] = 1; positionEvents[5] = 2;
        soundEvents[6] = 0; visualEvents[6] = 1; positionEvents[6] = 4;
        soundEvents[7] = 0; visualEvents[7] = 1; positionEvents[7] = 5;
        soundEvents[8] = 0; visualEvents[8] = 2; positionEvents[8] = 1;
        soundEvents[9] = 0; visualEvents[9] = 2; positionEvents[9] = 2;
        soundEvents[10] = 0; visualEvents[10] = 2; positionEvents[10] = 4;
        soundEvents[11] = 0; visualEvents[11] = 2; positionEvents[11] = 5;
        soundEvents[12] = 1; visualEvents[12] = 0; positionEvents[12] = 1;
        soundEvents[13] = 1; visualEvents[13] = 0; positionEvents[13] = 2;
        soundEvents[14] = 1; visualEvents[14] = 0; positionEvents[14] = 4;
        soundEvents[15] = 1; visualEvents[15] = 0; positionEvents[15] = 5;
        soundEvents[16] = 1; visualEvents[16] = 1; positionEvents[16] = 1;
        soundEvents[17] = 1; visualEvents[17] = 1; positionEvents[17] = 2;
        soundEvents[18] = 1; visualEvents[18] = 1; positionEvents[18] = 4;
        soundEvents[19] = 1; visualEvents[19] = 1; positionEvents[19] = 5;
        soundEvents[20] = 1; visualEvents[20] = 2; positionEvents[20] = 1;
        soundEvents[21] = 1; visualEvents[21] = 2; positionEvents[21] = 2;
        soundEvents[22] = 1; visualEvents[22] = 2; positionEvents[22] = 4;
        soundEvents[23] = 1; visualEvents[23] = 2; positionEvents[23] = 5;
        soundEvents[24] = 2; visualEvents[24] = 0; positionEvents[24] = 1;
        soundEvents[25] = 2; visualEvents[25] = 0; positionEvents[25] = 2;
        soundEvents[26] = 2; visualEvents[26] = 0; positionEvents[26] = 4;
        soundEvents[27] = 2; visualEvents[27] = 0; positionEvents[27] = 5;
        soundEvents[28] = 2; visualEvents[28] = 1; positionEvents[28] = 1;
        soundEvents[29] = 2; visualEvents[29] = 1; positionEvents[29] = 2;
```

```
        soundEvents[30] = 2; visualEvents[30] = 1; positionEvents[30] = 4;
        soundEvents[31] = 2; visualEvents[31] = 1; positionEvents[31] = 5;
        soundEvents[32] = 2; visualEvents[32] = 2; positionEvents[32] = 1;
        soundEvents[33] = 2; visualEvents[33] = 2; positionEvents[33] = 2;
        soundEvents[34] = 2; visualEvents[34] = 2; positionEvents[34] = 4;
        soundEvents[35] = 2; visualEvents[35] = 2; positionEvents[35] = 5;

    }

//This function before each block of trials. It randomises the order of the event
arrays while keeping each row paired with each other.
    void randomizeArrays()
    {
        for (int i = 35; i > 0; i--)
        {
            int j = Mathf.FloorToInt(Random.Range(0, i));
            int swapValue1 = soundEvents[j];
            int swapValue2 = visualEvents[j];
            int swapValue3 = positionEvents[j];
            soundEvents[j] = soundEvents[i];
            visualEvents[j] = visualEvents[i];
            positionEvents[j] = positionEvents[i];
            soundEvents[i] = swapValue1;
            visualEvents[i] = swapValue2;
            positionEvents[i] = swapValue3;
        }

        for (int i = 71; i > 0; i--)
        {
            int j = Mathf.FloorToInt(Random.Range(0, i));
            int swapValue1 = timings[j];
            timings[j] = timings[i];
            timings[i] = swapValue1;
        }
    }

//This function if called every time a target appears on the screens. It moves the
yellow target circle to the correct position on the screen based on the current
event's position value and rotates the dot to match the screen it is to be displayed
on. It also sends the correct cue through to the Recon Jet for display.
    void targetEvent()
    {
        eventStage = 1;
        render.colour = Colour.yellow;

        if (soundEvents[currentEvent] == 1)
        {
            AudioCueTransform.position = new Vector3(0, 0, 0);
            AC.Play();
        }
        if (soundEvents[currentEvent] == 2)
        {
            AudioCueTransform.position = new Vector3(0, 0, 0);
            AudioCueScript.Fire();
            AC.Play();
        }

        if (positionEvents[currentEvent] == 1)
        {
            xShift = (float)-2390;
            zShift = (float)1.731967 * xShift + (float)4226;
            this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
```

```
        this.transform.rotation = Quaternion.Euler(0, -60, 0);
        if (visualEvents[currentEvent] == 0)
        { sendState(26); }
        if (visualEvents[currentEvent] == 1)
        { sendState(11); }
        if (visualEvents[currentEvent] == 2)
        { sendState(21); }
    }
    if (positionEvents[currentEvent] == 2)
    {
        currentPosition = 2;
        xShift = (float)-1446;
        zShift = (float)1.731967 * xShift + (float)4226;
        this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
        this.transform.rotation = Quaternion.Euler(0, -60, 0);
        if (visualEvents[currentEvent] == 0)
        { sendState(26); }
        if (visualEvents[currentEvent] == 1)
        { sendState(12); }
        if (visualEvents[currentEvent] == 2)
        { sendState(22); }
    }
    if (positionEvents[currentEvent] == 3)
    {
        currentPosition = 3;
        xShift = (float)0;
        zShift = (float)2113;
        this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
        this.transform.rotation = Quaternion.Euler(0, 0, 0);
        if (visualEvents[currentEvent] == 0)
        { sendState(26); }
        if (visualEvents[currentEvent] == 1)
        { sendState(13); }
        if (visualEvents[currentEvent] == 2)
        { sendState(23); }
    }
    if (positionEvents[currentEvent] == 4)
    {
        currentPosition = 4;
        xShift = (float)1446;
        zShift = (float)-1.731967 * xShift + (float)4226;
        this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
        this.transform.rotation = Quaternion.Euler(0, 60, 0);
        if (visualEvents[currentEvent] == 0)
        { sendState(26); }
        if (visualEvents[currentEvent] == 1)
        { sendState(14); }
        if (visualEvents[currentEvent] == 2)
        { sendState(24); }
    }
    if (positionEvents[currentEvent] == 5)
    {
        currentPosition = 5;
        xShift = (float)2390;
        zShift = (float)-1.731967 * xShift + (float)4226;
        this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
        this.transform.rotation = Quaternion.Euler(0, 60, 0);
        if (visualEvents[currentEvent] == 0)
        { sendState(26); }
        if (visualEvents[currentEvent] == 1)
        { sendState(15); }
        if (visualEvents[currentEvent] == 2)
```

```
            { sendState(25); }
        }
        if (positionEvents[currentEvent] == 0) //If this happens something has gone
                                                wrong.
        {
            currentPosition = 4;
            xShift = (float)1446;
            zShift = (float)-1.731967 * xShift + (float)4226;
            this.transform.position = new Vector3(xShift / 100, 0, zShift / 100);
            this.transform.rotation = Quaternion.Euler(0, 60, 0);
            if (visualEvents[currentEvent] == 0)
            { sendState(26); }
            if (visualEvents[currentEvent] == 1)
            { sendState(15); }
            if (visualEvents[currentEvent] == 2)
            { sendState(25); }
            System.IO.File.AppendAllText(record.fileName1, "Position Zero error
            occured" + System.Environment.NewLine);
        }

        System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 + ",1," +
        currentPosition + "," + headtrack.projectedX + "," + headtrack.projectedY + ","
        + headtrack.projectedZ + "," + guntrack.projectedX + "," + guntrack.projectedY
        + "," + guntrack.projectedZ + "," + soundEvents[currentEvent] + "," +
        visualEvents[currentEvent] + "," + positionEvents[currentEvent] +
        System.Environment.NewLine);
    }
}
```

//

//The third class is the head tracking object. It is responsible for connecting to the tracking software via VRPN and keeping the position and rotation of the head tracker updated. It also projects a ray from the user's head position out to the center of their view on the screen which is marked as a small green dot.

```
using UnityEngine;
using System.Collections;

public class HeadTracking : MonoBehaviour
{
        public float horizontalRotation;
        public float verticalRotation;
        public float xShift;
        public float yShift;
        public float zShift;
        public float projectedX;
        public float projectedY;
        public float projectedZ;
        public GameObject redDot;
        public string deviceName = "DTrack@tcp://hitlvstracker.canterbury.ac.nz";

        //Establishes VRPN connection when the program starts
        void Start ()
        {
                VRPN.vrpnOpen(deviceName);
                Debug.Log("Open");
        }

        //Terminates VRPN connection when the program ends
        void OnDestroy()
```

```
{
	VRPN.vrpnClose();
	Debug.Log("Close");
}

//Each frame the new tracking data is pulled from the VRPN. This is then
converted into the same units and scale that is used by unity; degrees ->
radians, ranging from -PI to PI horizontally and -0.4PI to 0.4PI vertical.
void Update ()
{
	VRPN.vrpnUpdate();

	float[] pos = new float[3];
	float[] quat = new float[4];
	VRPN.vrpnGetTrackedBody(109, pos, quat);
	Quaternion q = new Quaternion(-quat[0], -quat[1], quat[2], quat[3]);
	horizontalRotation = (q.eulerAngles.y-0)*Mathf.PI/180F;
	if (horizontalRotation > Mathf.PI)
	{
		horizontalRotation = horizontalRotation - Mathf.PI - Mathf.PI;
	}
	verticalRotation = (q.eulerAngles.x-10)*Mathf.PI/180F;
	if (verticalRotation > Mathf.PI)
	{
		verticalRotation = verticalRotation - Mathf.PI - Mathf.PI;
	}
	if (verticalRotation > 0.4F*Mathf.PI)
	{
		verticalRotation = 0.4F*Mathf.PI;
	}
	if (verticalRotation < -0.4F*Mathf.PI)
	{
		verticalRotation = -0.4F*Mathf.PI;
	}
	verticalRotation = -verticalRotation;
	xShift = pos[0]*1000F;
	yShift = pos[1]*1000F;
	zShift = -pos[2]*1000F;

	ProjectOntoPlane ();
}

//Called each update. Uses trigonometry to project the user's view out from
their position and onto the virtual representation of the screens. First
projects the user's head bearing onto the far screen and then, if the position
falls off the edge of this screen to the left or the right, projects the user's
bearing onto the left or right screen appropriately.
void ProjectOntoPlane()
{
	float projXTemp = xShift + ((float)2113 - zShift) *
	Mathf.Tan(horizontalRotation);
	if (projXTemp < -1220)
	{
		if (horizontalRotation == 0)
		{
			projectedX = xShift;
		}
		else
		{
			projectedX = ((float)4226 + (xShift /
			Mathf.Tan(horizontalRotation)) - zShift) / ((float)-
			1.731967 + (1 / Mathf.Tan(horizontalRotation)));
```

```
                }
                projectedZ = (float)+1.731967 * projectedX + (float)4226;
                projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
                (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
                zShift)) * Mathf.Tan(verticalRotation));
                redDot.transform.position = new Vector3(projectedX / 100,
                projectedY / 100, projectedZ / 100);
                redDot.transform.rotation = Quaternion.Euler(0, -60, 0);
            }
            else if (projXTemp < 1220)
            {
                projectedX = projXTemp;
                projectedZ = (float)2113;
                projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
                (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
                zShift)) * Mathf.Tan(verticalRotation));
                redDot.transform.position = new Vector3(projectedX / 100,
                projectedY / 100, projectedZ / 100);
                redDot.transform.rotation = Quaternion.Euler(0, 0, 0);
            }
            else
            {
                if (horizontalRotation == 0)
                {
                    projectedX = xShift;
                }
                else
                {
                    projectedX = ((float)4226 + (xShift /
                    Mathf.Tan(horizontalRotation)) - zShift) /
                    ((float)1.731967 + (1 / Mathf.Tan(horizontalRotation)));
                }
                projectedZ = (float)-1.731967 * projectedX + (float)4226;
                projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
                (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
                zShift)) * Mathf.Tan(verticalRotation));
                redDot.transform.position = new Vector3(projectedX / 100,
                projectedY / 100, projectedZ / 100);
                redDot.transform.rotation = Quaternion.Euler(0, 60, 0);
            }
        }

}
```

//

//The fourth class is the gun tracking object. It runs a similar set of functions to the head tracking unit but it also handles the trigger pulls and creation of the virtual paint splats. The projection of the gun's bearing onto the virtual screen is marked with a blue dot.

```
using UnityEngine;
using System.Collections;

public class GunTracking : MonoBehaviour
{
    public float horizontalRotation;
    public float verticalRotation;
    public float xShift;
    public float yShift;
    public float zShift;
    public float projectedX;
```

```
public float projectedY;
public float projectedZ;
public GameObject redDot;
public Transform paintsplotch;
public string deviceName = "DTrack@tcp://hitlvstracker.canterbury.ac.nz";
public Recording record;
public HeadTracking headtrack;
public TargetPlacer target;
public float tDist;
public float tDist1;
public float tDist2;
public float tDist3;

//Establishes VRPN connection when the program starts
void Start ()
{
	VRPN.vrpnOpen(deviceName);
	Debug.Log("Open");
}

//Terminates VRPN connection when the program ends
void OnDestroy()
{
	VRPN.vrpnClose();
	Debug.Log("Close");
}

//Each frame the new tracking data is pulled from the VRPN. This is then
converted into the same units and scale that is used by unity; degrees ->
radians, ranging from -PI to PI horizontally and -0.4PI to 0.4PI vertical.
void Update ()
{
	//VRPN.vrpnUpdate(); This is updated by the head tracker and doesn't
	need to be called twice.

	float[] pos = new float[3];
	float[] quat = new float[4];
	VRPN.vrpnGetTrackedBody(10, pos, quat);
	Quaternion q = new Quaternion(-quat[0], -quat[1], quat[2], quat[3]);
	horizontalRotation = (q.eulerAngles.y-0)*Mathf.PI/180F;
	if (horizontalRotation > Mathf.PI)
	{
		horizontalRotation = horizontalRotation - Mathf.PI - Mathf.PI;
	}
	verticalRotation = (q.eulerAngles.x-3)*Mathf.PI/180F;
	if (verticalRotation > Mathf.PI)
	{
		verticalRotation = verticalRotation - Mathf.PI - Mathf.PI;
	}
	if (verticalRotation > 0.4F*Mathf.PI)
	{
		verticalRotation = 0.4F*Mathf.PI;
	}
	if (verticalRotation < -0.4F*Mathf.PI)
	{
		verticalRotation = -0.4F*Mathf.PI;
	}
	verticalRotation = -verticalRotation;
	xShift = pos[0]*1000F;
	yShift = pos[1]*1000F;
	zShift = -pos[2]*1000F;
```

```
        ProjectOntoPlane ();
}

//Called each update. Uses trigonometry to project the user's view out from
their position and onto the virtual representation of the screens. First
projects the user's head bearing onto the far screen and then, if the position
falls off the edge of this screen to the left or the right, projects the user's
bearing onto the left or right screen appropriately. Also, detects trigger
pulls, records the shot and instantiates the paint splotch. Progresses the
event timing structure of the main function loop if the shot is close to an
active target.
void ProjectOntoPlane()
{
        float projXTemp = xShift + ((float)2113 - zShift) *
        Mathf.Tan(horizontalRotation);
        if (projXTemp < -1220)
        {
                if (horizontalRotation == 0)
                {
                        projectedX = xShift;
                }
                else
                {
                        projectedX = ((float)4226 + (xShift /
                        Mathf.Tan(horizontalRotation)) - zShift) / ((float)-
                        1.731967 + (1 / Mathf.Tan(horizontalRotation)));
                }
                projectedZ = (float)+1.731967 * projectedX + (float)4226;
                projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
                (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
                zShift)) * Mathf.Tan(verticalRotation));
                redDot.transform.position = new Vector3(projectedX / 100,
                projectedY / 100, projectedZ / 100);
                redDot.transform.rotation = Quaternion.Euler(0, -60, 0);
                if (Input.GetMouseButtonDown(0))
                {
                        Instantiate(paintsplotch,new
                        Vector3(projectedX/100,projectedY/100,projectedZ/100),Quat
                        ernion.Euler(0, -60, Random.Range(0F, 360F)));
                        switch (target.metaStage)
                        {
                        case 2:
                                if (Vector3.Distance(new Vector3(projectedX,
                                projectedY, projectedZ), new Vector3(target.xShift,
                                0, target.zShift)) < 200F)
                                {
                                        target.metaStage = 3;
                                        target.xShift = (float)2390;
                                        target.zShift = (float)-1.731967 *
                                        target.xShift + (float)4226;
                                        target.transform.position = new
                                        Vector3(target.xShift / 100, 0, target.zShift
                                        / 100);
                                        target.transform.rotation =
                                        Quaternion.Euler(0, 60, 0);
                                }
                                break;
                        case 4:
                        case 6:
                        case 8:
                        case 10:
```

```
                        if ((target.eventStage == 1) &&
                        (Vector3.Distance(new Vector3(projectedX,
                        projectedY, projectedZ), new Vector3(target.xShift,
                        0, target.zShift)) < 200F))
                        {
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",3," +
                                target.currentPosition + "," +
                                headtrack.projectedX + "," +
                                headtrack.projectedY + "," +
                                headtrack.projectedZ + "," + projectedX + ","
                                + projectedY + "," + projectedZ +
                                System.Environment.NewLine);
                                target.eventStage = 2;
                        }
                        else
                        {
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",2," +
                                target.currentPosition + "," +
                                headtrack.projectedX + "," +
                                headtrack.projectedY + "," +
                                headtrack.projectedZ + "," + projectedX + ","
                                + projectedY + "," + projectedZ +
                                System.Environment.NewLine);
                        }
                        break;
                }
        }
}
else if (projXTemp < 1220)
{
        projectedX = projXTemp;
        projectedZ = (float)2113;
        projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
        (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
        zShift)) * Mathf.Tan(verticalRotation));
        redDot.transform.position = new Vector3(projectedX / 100,
        projectedY / 100, projectedZ / 100);
        redDot.transform.rotation = Quaternion.Euler(0, 0, 0);
        if (Input.GetMouseButtonDown(0))
        {
                Instantiate(paintsplotch,new
                Vector3(projectedX/100,projectedY/100,projectedZ/100),Quat
                ernion.Euler(0, 0, Random.Range(0F, 360F)));
                switch (target.metaStage)
                {
                case 4:
                case 6:
                case 8:
                case 10:
                        if ((target.eventStage == 1) &&
                        (Vector3.Distance(new Vector3(projectedX,
                        projectedY, projectedZ), new Vector3(target.xShift,
                        0, target.zShift)) < 200F))
                        {
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",3," +
                                target.currentPosition + "," +
                                headtrack.projectedX + "," +
                                headtrack.projectedY + "," +
                                headtrack.projectedZ + "," + projectedX + ","
```

```
                                + projectedY + "," + projectedZ +
                                System.Environment.NewLine);
                                target.eventStage = 2;
                        }
                        else
                        {
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",2," +
                                target.currentPosition + "," +
                                headtrack.projectedX + "," +
                                headtrack.projectedY + "," +
                                headtrack.projectedZ + "," + projectedX + ","
                                + projectedY + "," + projectedZ +
                                System.Environment.NewLine);
                        }
                        break;
                }
        }
}
else
{
        if (horizontalRotation == 0)
        {
                projectedX = xShift;
        }
        else
        {
                projectedX = ((float)4226 + (xShift /
                Mathf.Tan(horizontalRotation)) - zShift) /
                ((float)1.731967 + (1 / Mathf.Tan(horizontalRotation)));
        }
        projectedZ = (float)-1.731967 * projectedX + (float)4226;
        projectedY = yShift + (Mathf.Sqrt((projectedX - xShift) *
        (projectedX - xShift) + (projectedZ - zShift) * (projectedZ -
        zShift)) * Mathf.Tan(verticalRotation));
        redDot.transform.position = new Vector3(projectedX / 100,
        projectedY / 100, projectedZ / 100);
        redDot.transform.rotation = Quaternion.Euler(0, 60, 0);
        if (Input.GetMouseButtonDown(0))
        {
                Instantiate(paintsplotch, new Vector3(projectedX / 100,
                projectedY / 100, projectedZ / 100), Quaternion.Euler(0,
                60, Random.Range(0F, 360F)));
                switch (target.metaStage)
                {
                case 3:
                        if (Vector3.Distance(new Vector3(projectedX,
                        projectedY, projectedZ), new Vector3(target.xShift,
                        0, target.zShift)) < 200F)
                        {
                                target.metaStage = 114;
                                target.transform.position = new Vector3(0, 0,
                                25);
                        }
                        break;
                case 4:
                case 6:
                case 8:
                case 10:
                        if ((target.eventStage == 1) &&
                        (Vector3.Distance(new Vector3(projectedX,
```

```
                                    projectedY, projectedZ), new Vector3(target.xShift,
                                    0, target.zShift)) < 200F))
                                    {
                                        System.IO.File.AppendAllText(record.fileName1
                                        , Time.time * 1000 + ",3," +
                                        target.currentPosition + "," +
                                        headtrack.projectedX + "," +
                                        headtrack.projectedY + "," +
                                        headtrack.projectedZ + "," + projectedX + ","
                                        + projectedY + "," + projectedZ +
                                        System.Environment.NewLine);
                                        target.eventStage = 2;
                                    }
                                    else
                                    {
                                        System.IO.File.AppendAllText(record.fileName1
                                        , Time.time * 1000 + ",2," +
                                        target.currentPosition + "," +
                                        headtrack.projectedX + "," +
                                        headtrack.projectedY + "," +
                                        headtrack.projectedZ + "," + projectedX + ","
                                        + projectedY + "," + projectedZ +
                                        System.Environment.NewLine);
                                    }
                                    break;
                            }
                        }
                    }
            }
    }
```

//
//The fifth class of the main program is controls the data output files. It runs its start function at the beginning of the program and creates the save files, checking that it won't overwrite an existing file. It adds the titles and headers to the data file and then waits to be called by the main function and the gun tracker, both of which record data at various times.

```
using UnityEngine;
using System.Collections;

public class Recording : MonoBehaviour
{
    private bool looping =true;
    private int loopcount = 1;
    public string fileName1 = "Exp3Output-";
    public string fileName2 = "Exp3Output-";
    public string fileName3 = "Exp3Output-";
    public GunTracking guntrack;
    public HeadTracking headtrack;

    void Start ()
    {
        fileName1 = fileName1 + System.DateTime.Now.Year + " " +
        System.DateTime.Now.Month + " " + System.DateTime.Now.Day + " " +
        System.DateTime.Now.Hour;
        fileName2 = fileName2 + System.DateTime.Now.Year + " " +
        System.DateTime.Now.Month + " " + System.DateTime.Now.Day + " " +
        System.DateTime.Now.Hour;
```

```
        fileName3 = fileName3 + System.DateTime.Now.Year + " " +
        System.DateTime.Now.Month + " " + System.DateTime.Now.Day + " " +
        System.DateTime.Now.Hour;
        while (looping)
        {
                if (System.IO.File.Exists(fileName1+"-"+loopcount+"-
                EventLog.txt"))
                {
                        loopcount = loopcount + 1;
                }
                else
                {
                        if (System.IO.File.Exists(fileName2+"-"+loopcount+"-
                        GunTracking.txt"))
                        {
                                loopcount = loopcount + 1;
                        }
                        else
                        {
                                if (System.IO.File.Exists(fileName3+"-
                                "+loopcount+"-HeadTracking.txt"))
                                {
                                        loopcount = loopcount + 1;
                                }
                                else
                                {
                                        fileName1 = fileName1+"-"+loopcount+"-
                                        EventLog.txt";
                                        fileName2 = fileName2+"-"+loopcount+"-
                                        GunTracking.txt";
                                        fileName3 = fileName3+"-"+loopcount+"-
                                        HeadTracking.txt";
                                        looping = false;
                                }
                        }
                }
        }
        System.IO.File.AppendAllText (fileName1,
        "Time_Millis,EventType_1Targets_2Shot_3EventEndingShot_4ReturnToCenter,C
        urrentTargetPosition,HeadProjectedX,HeadProjectedY,HeadProjectedZ,GunPro
        jectedX,GunProjectedY,GunProjectedZ,SoundEventType,VisualEventType,Posit
        ionEventType" + System.Environment.NewLine);
        System.IO.File.AppendAllText (fileName2,
        "Time_Millis,GunProjectedX,GunProjectedY,GunProjectedZ,HorizontalRotatio
        n,VerticalRotation" + System.Environment.NewLine);
        System.IO.File.AppendAllText (fileName3,
        "Time_Millis,HeadProjectedX,HeadProjectedY,HeadProjectedZ,HorizontalRota
        tion,VerticalRotation" + System.Environment.NewLine);
}

//This update creates the full tracking logs for the gun and the head trackers.
It runs every frame.
void Update ()
{
        System.IO.File.AppendAllText (fileName2, Time.time*1000 + "," +
        guntrack.projectedX + "," + guntrack.projectedY + "," +
        guntrack.projectedZ + "," + guntrack.horizontalRotation + "," +
        guntrack.verticalRotation + System.Environment.NewLine);
        System.IO.File.AppendAllText (fileName3, Time.time*1000 + "," +
        headtrack.projectedX + "," + headtrack.projectedY + "," +
        headtrack.projectedZ + "," + headtrack.horizontalRotation + "," +
        headtrack.verticalRotation + System.Environment.NewLine);
```

```
        }
}
```

//

Appendix 3.3: Recon Jet Receiver Program for Experiment 3

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.Networking;

public class stateTransfer : MessageBase
{
    public int value;
}

public class FadeControl : MonoBehaviour
{
    public Text currentText;
    public Text previousText1;
    public Text previousText2;
    public Text previousText3;
    public Text previousText4;
    public float scalar;
    public int stage = 0;
    public int incomingValue = 0;
    public float step = 0;
    NetworkClient myClient;
    float xscalebase = 0.8f;
    float yscalebase = 0.8f;
    public RawImage image;
    public RectTransform imagetrans;
    public Texture arrow1;
    public Texture arrow2;
    public Texture arrow3;
    public Texture arrow4;
    public Texture arrow5;
    public RectTransform canvastrans;

    int alphastage = 0;
    int alphafloor = 1;
    string alphaLocationLetter = "C";
    int alphaLocationNumber = 13;
    int bravostage = 0;
    int bravofloor = 1;
    string bravoLocationLetter = "D";
    int bravoLocationNumber = 7;
    int deltastage = 0;
    int deltafloor = 1;
    string deltaLocationLetter = "E";
    int deltaLocationNumber= 15;
    string display1 = " ";
    string display2 = " ";
    string display3 = " ";
    int ticker= 0;

       //Start by setting the screen to stay active and trying to connect to the
       desktop PC program
       public void Start()
       {
```

```
        Screen.sleepTimeout = SleepTimeout.NeverSleep;
        alphaLocationNumber = Random.Range(3, 20);
        bravoLocationNumber = Random.Range(3, 20);
        deltaLocationNumber = Random.Range(3, 20);
        connectionFunction ();
}

//Attempts the connection with the primary program.
void connectionFunction()
{
        myClient = new NetworkClient();

        myClient.RegisterHandler(MsgType.Connect, OnConnected);
        myClient.RegisterHandler(MsgType.Disconnect, OnDisconnected);
        myClient.RegisterHandler(MsgType.Error, OnError);
        myClient.RegisterHandler(7772, StateUpdate);

        myClient.Connect("192.168.1.66", 25000);
        //myClient.Connect("10.1.1.2", 25000);
}

//Set the program to ready if the connection is achieved
public void OnConnected(NetworkMessage msg)
{
        stage = 1;
}

//Attempt to reconnect if the connection drops
public void OnDisconnected(NetworkMessage msg)
{
        currentText.text = "Disconnected";
        connectionFunction ();
}

//Display an error if necessary
public void OnError(NetworkMessage msg)
{
        currentText.text = "Error";
}

//This function creates the trail of previous messages above the first by
shifting old messages up a step.
void shiftText()
{
previousText2.text = previousText1.text;
previousText1.text = currentText.text;
currentText.text = "";
}

//This function is called every time a sync message is received from the
primary program. This is a list of incoming message types:
    // 0: Connecting
    // 1: Make Sure You Can Read This
    // 2: Blank
    // 3: Generated Message
    // 11: Static Event -100
    // 12: Static Event -50
    // 13: Static Event 0
    // 14: Static Event 50
    // 15: Static Event 100
    // 21: Pursuit Event -100
    // 22: Pursuit Event -50
```

```
    // 23: Pursuit Event 0
    // 24: Pursuit Event 50
    // 25: Pursuit Event 100
    // 26: Pursuit Blank
public void StateUpdate(NetworkMessage msg)
{
       incomingValue = msg.ReadMessage<stateTransfer>().value;
       if (incomingValue < 20)
       {
              canvastrans.position = new Vector3(0f, 0f, 0f);
              canvastrans.localScale = new Vector3(xscalebase, yscalebase, 1f);
       }
       switch (incomingValue)
       {
       case 1:
              stage = 1;
              break;
       case 2:
              stage = 2;
              break;
       case 3:
              display3 = display2;
              display2 = display1;
              int randomInt = Random.Range(1, 4);
              switch (randomInt)
              {
              case 1:
                     if (alphastage == 0)
                     {
                            display1 = "Alpha Team entered site " +
                            alphaLocationLetter + alphaLocationNumber + " at "
                            + (System.DateTime.Now.Hour * 100 +
                            System.DateTime.Now.Minute) + ".";
                            alphastage = 1;
                     }
                     else if (alphastage == 1)
                     {
                            display1 = "Alpha Team cleared floor " + alphafloor
                            + " at " + (System.DateTime.Now.Hour * 100 +
                            System.DateTime.Now.Minute) + ".";
                            int temp = Random.Range(0, 7);
                            if (alphafloor < temp)
                            {
                                   alphafloor = alphafloor + 1;
                            }
                            else
                            {
                                   alphafloor = 1;
                                   alphastage = 2;
                            }
                     }
                     else
                     {
                            display1 = "Alpha Team exited site " +
                            alphaLocationLetter + alphaLocationNumber + " at "
                            + (System.DateTime.Now.Hour * 100 +
                            System.DateTime.Now.Minute) + ".";
                            int temp = Random.Range(1, 5);
                            if (temp == 4)
                            {
                                   alphaLocationNumber = alphaLocationNumber +
                                   2;
```

```
                }
                else
                {
                        alphaLocationNumber = alphaLocationNumber +
                        1;
                }
                alphastage = 0;
        }
        break;
case 2:
        if (bravostage == 0)
        {
                display1 = "Bravo Team entered site " +
                bravoLocationLetter + bravoLocationNumber + " at "
                + (System.DateTime.Now.Hour * 100 +
                System.DateTime.Now.Minute) + ".";
                bravostage = 1;
        }
        else if (bravostage == 1)
        {
                display1 = "Bravo Team cleared floor " + bravofloor
                + " at " + (System.DateTime.Now.Hour * 100 +
                System.DateTime.Now.Minute) + ".";
                int temp = Random.Range(0, 7);
                if (bravofloor < temp)
                {
                        bravofloor = bravofloor + 1;
                }
                else
                {
                        bravofloor = 1;
                        bravostage = 2;
                }
        }
        else
        {
                display1 = "Bravo Team exited site " +
                bravoLocationLetter + bravoLocationNumber + " at "
                + (System.DateTime.Now.Hour * 100 +
                System.DateTime.Now.Minute) + ".";
                int temp = Random.Range(1, 5);
                if (temp == 4)
                {
                        bravoLocationNumber = bravoLocationNumber +
                        2;
                }
                else
                {
                        bravoLocationNumber = bravoLocationNumber +
                        1;
                }
                bravostage = 0;
        }
        break;
case 3:
        if (deltastage == 0)
        {
                display1 = "Delta Team entered site " +
                deltaLocationLetter + deltaLocationNumber + " at "
                + (System.DateTime.Now.Hour * 100 +
                System.DateTime.Now.Minute) + ".";
                deltastage = 1;
```

```
                }
                else if (deltastage == 1)
                {
                        display1 = "Delta Team cleared floor " + deltafloor
                        + " at " + (System.DateTime.Now.Hour * 100 +
                        System.DateTime.Now.Minute) + ".";
                        int temp = Random.Range(0, 7);
                        if (deltafloor < temp)
                        {
                                deltafloor = deltafloor + 1;
                        }
                        else
                        {
                                deltafloor = 1;
                                deltastage = 2;
                        }
                }
                else
                {
                        display1 = "Delta Team exited site " +
                        deltaLocationLetter + deltaLocationNumber + " at "
                        + (System.DateTime.Now.Hour * 100 +
                        System.DateTime.Now.Minute) + ".";
                        int temp = Random.Range(1, 5);
                        if (temp == 4)
                        {
                                deltaLocationNumber = deltaLocationNumber +
                                2;
                        }
                        else
                        {
                                deltaLocationNumber = deltaLocationNumber +
                                1;
                        }
                        deltastage = 0;
                }
                break;
        }
        stage = 3;
        break;
case 11:
        stage = 11;
        break;
case 12:
        stage = 12;
        break;
case 13:
        stage = 13;
        break;
case 14:
        stage = 14;
        break;
case 15:
        stage = 15;
        break;
case 21:
        stage = 21;
        step = 0;
        break;
case 22:
        stage = 22;
        step = 0;
```

```
            break;
        case 23:
            stage = 23;
            step = 0;
            break;
        case 24:
            stage = 24;
            step = 0;
            break;
        case 25:
            stage = 25;
            step = 0;
            break;
        case 26:
            stage = 26;
            step = 0;
            break;
        }
}

//This function runs ever frame and draws the text and images to the screen
based on the current screen being displayed.
void Update()
{
        imagetrans.position = new Vector3(500f, 0f, 0f);
        switch (stage)
        {
        case 0:
            ticker = ticker + 1;
            currentText.text = "Attempting to connect " + ticker;
            break;
        case 1:
            previousText2.text = " ";
            previousText1.text = "Before starting the experiment";
            currentText.text = "Make sure you can clearly read this.";
            break;
        case 2:
            previousText2.text = " ";
            previousText1.text = " ";
            currentText.text = " ";
            break;
        case 3:
            previousText2.text = display3;
            previousText1.text = display2;
            currentText.text = display1;
            break;
        case 11:
            imagetrans.position = new Vector3(0f, 0f, 0f);
            image.texture = arrow1;
            break;
        case 12:
            imagetrans.position = new Vector3(0f, 0f, 0f);
            image.texture = arrow2;
            break;
        case 13:
            imagetrans.position = new Vector3(0f, 0f, 0f);
            image.texture = arrow3;
            break;
        case 14:
            imagetrans.position = new Vector3(0f, 0f, 0f);
            image.texture = arrow4;
            break;
```

```
case 15:
       imagetrans.position = new Vector3(0f, 0f, 0f);
       image.texture = arrow5;
       break;
case 21:
       if (step < 20)
       {
       canvastrans.position = new Vector3(-68f * step, 16f * step, 0f);
       step = step + 1;
       previousText2.text = display3;
       previousText1.text = display2;
       currentText.text = display1;
       }
       break;
case 22:
       if (step < 20)
       {
       canvastrans.position = new Vector3(-63f * step, 30f * step, 0f);
       step = step + 1;
       previousText2.text = display3;
       previousText1.text = display2;
       currentText.text = display1;
       }
       break;
case 23:
       if (step < 20)
       {
       canvastrans.position = new Vector3(0f * step, 23f * step, 0f);
       step = step + 1;
       previousText2.text = display3;
       previousText1.text = display2;
       currentText.text = display1;
       }
       break;
case 24:
       if (step < 20)
       {
       canvastrans.position = new Vector3(63f * step, 30f * step, 0f);
       step = step + 1;
       previousText2.text = display3;
       previousText1.text = display2;
       currentText.text = display1;
       }
       break;
case 25:
       if (step < 20)
       {
       canvastrans.position = new Vector3(68f * step, 16f * step, 0f);
       step = step + 1;
       previousText2.text = display3;
       previousText1.text = display2;
       currentText.text = display1;
       }
       break;
case 26:
       if (step < 20)
       {
       canvastrans.position = new Vector3(500f, 0f, 0f);
       canvastrans.localScale = new Vector3(xscalebase, yscalebase, 1f);
       }
       break;
default:
```

```
                    break;
                }
            }
}
```

//

Appendix 3.4: Example Output and Data Processing for Experiment 3

Example data output for an Experiment 3 tracking log. Measurements are taken even frame of the program. The top row is its printed position in the excel file. The data must be kept in these columns for the macros to run. Displayed data is drawn from the first participant's file.

A	B	C	D	E
Time_Millis	GunProjectedX	GunProjectedY	GunProjectedZ	HorizontalRotation
20	2040.818	-1113.91	691.3708	1.24416
34.68578	2039.233	-1110.89	694.116	1.242716
46.96676	2039.233	-1110.89	694.116	1.242716
62.95216	2037.398	-1109.75	697.2941	1.241045
80.07058	2020.776	-1087.03	726.0836	1.225852
96.62266	2018.302	-1082.21	730.3676	1.223584
114.0251	2017.534	-1082.45	731.697	1.22288
130.2513	2016.87	-1079.27	732.8476	1.22227
147.1893	2016.27	-1076.2	733.8863	1.22172
163.239	2016.816	-1076.52	732.9418	1.222221
180.2064	2017.065	-1073.68	732.5101	1.222449
…	…	…	…	…

Example data output for an Experiment 3 event log. Measurements are taken every time a target appears (Event Type 1), a shot is missed (Event Type 2), a shot hits a target (Event Type 3), and when the participant returns their vision to the center screen (Event Type 4). When a target appears, its position, visual event type (0 = no cue, 1 = arrow cue, 2 = pursuit motion cue), and sound event (0 = no event, 1 = static sound cue, 2 = dynamic sound cue) is recorded. The letters in the top row are the excel columns the data needs to be placed in to run the analysis macros.

G	H	I	J	K	L	M	N	O	P	Q	R
Time_Millis	Event Type	CurrentTargetPosition	HeadProjectedX	HeadProjectedY	HeadProjectedZ	GunProjectedX	GunProjectedY	GunProjectedZ	SoundEvent	VisualEvent	TargetPosition
231654.8	1	4	-147.906	-217.68	2113	-595.118	-554.785	2113	2	1	4
233503.8	3	4	1410.034	-189.675	1783.868	1484.38	-9.44463	1655.103			
234686.8	4	4	362.7701	-202.374	2113	-133.728	-675.64	2113			
240717.5	1	1	318.5417	-69.3741	2113	-156.235	-603.342	2113	0	2	1
244265.5	3	1	-1811.41	79.22141	1088.694	-2377.46	10.24741	108.3179			
245448.6	4	1	-377.033	-217.618	2113	-662.525	-530.689	2113			
251479.3	1	2	92.9645	-165.292	2113	-431.481	-604.551	2113	1	0	2
253128.4	3	2	-1029.56	-105.876	2113	-1415.37	-18.2413	1774.623			
253794.2	4	2	-397.985	-264.656	2113	-1068.55	-353.762	2113			
259825.2	1	5	264.0064	-212.924	2113	-395.689	-668.064	2113	0	0	5
265722	3	5	2170.021	-180.71	467.5953	2419.827	37.12094	34.94015			
266939.2	4	5	387.1001	-439.18	2113	-159.622	-1032.88	2113			
272968.7	1	2	215.6319	-349.668	2113	-694.471	-692.891	2113	2	2	2
274534.7	3	2	-615.923	-194.926	2113	-1452.02	-14.2969	1711.154			
275084.7	4	2	-392.049	-288.421	2113	-1257.61	-236.372	2047.862			
…	…	…	…	…	…	…	…	…	…	…	…

Once the data has been combined into a single file with the data in the correct column the following macro is used to extract the horizontal rotation data from the head tracking around each event, starting one second before an event and continuing until the target is marked with paint.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
Sub headpull1()
'declare variables
Dim DataRow As Long
Dim EndBinary As Integer
Dim PrintColumn As Integer
Dim PrintRow As Integer
Dim StartTime As Long
Dim EndTime As Long
Dim EventRow As Integer
Dim EventBinary As Integer
Dim MarkBinary As Integer
Dim TargetBinary As Integer
Dim TargetBinary2 As Integer
Dim OverBinary As Integer

'set variables
DataRow = 2
EndBinary = 0
PrintColumn = 22
PrintRow = 6
StartTime = 0
EndTime = 0
EventRow = 3
MarkBinary = 0
TargetBinary = 0
TargetBinary2 = 0
OverBinary = 0
StartRow = -1
Cells(6, PrintColumn + 3).Value = "Velocity"
Cells(6, PrintColumn + 4).Value = ""
Cells(7, PrintColumn + 4).Value = "Smoothed Velocity (100ms)"

'find the beginning of the first event and create header (occurs when column 8 has a value of 1)
EventBinary = 0
Do Until EventBinary = 1
        If Application.WorksheetFunction.CountA(Cells(EventRow, 7)) = 1 Then
                If Cells(EventRow, 8).Value = 1 Then
                        StartTime = Cells(EventRow, 7).Value
                        Cells(3, PrintColumn).Value = "Sound Event"
                        Cells(4, PrintColumn).Value = "Visual Event"
                        Cells(5, PrintColumn).Value = "Position"
                        Cells(3, PrintColumn + 1).Value = Cells(EventRow, 16).Value
                        Cells(4, PrintColumn + 1).Value = Cells(EventRow, 17).Value
```

```
                    Cells(5, PrintColumn + 1).Value = Cells(EventRow, 18).Value
                    EventBinary = 1
                Else
                    EventRow = EventRow + 1
                End If
            Else
                EventBinary = 1
                EndBinary = 1
            End If
Loop

'find the end of the first event and store times (occurs when column 8 has a value of 3)
EventBinary = 0
Do Until EventBinary = 1
        If Application.WorksheetFunction.CountA(Cells(EventRow, 7)) = 1 Then
            If Cells(EventRow, 8).Value = 3 Then
                EndTime = Cells(EventRow, 7).Value
                EventBinary = 1
                Cells(3, PrintColumn + 2).Value = "Time to Hit"
                Cells(4, PrintColumn + 2).Value = EndTime - StartTime
            Else
                EventRow = EventRow + 1
            End If
        Else
            EndBinary = 1
            EventBinary = 1
        End If
Loop

'loop through the data rows, pulling out rows which occur between 1000ms before an event and the
end time
Do Until EndBinary = 1
        If Application.WorksheetFunction.CountA(Cells(DataRow, 1)) = 1 Then
            DataRow = DataRow + 1
            Cells(3, 20).Value = DataRow
            If Cells(DataRow, 1).Value > (StartTime - 1000) Then
                Cells(PrintRow, PrintColumn + 1).Value = Cells(DataRow, 1).Value - StartTime
                Cells(PrintRow, PrintColumn + 2).Value = Cells(DataRow, 5).Value
                If PrintRow > 6 Then
                    Cells(PrintRow, PrintColumn + 3).Value = 1000 * (Cells(PrintRow,
                    PrintColumn + 2).Value - Cells(PrintRow - 1, PrintColumn + 2).Value)
                    / (Cells(PrintRow, PrintColumn + 1).Value - Cells(PrintRow - 1,
                    PrintColumn + 1).Value)
                End If
                If PrintRow > 12 Then
                    Cells(PrintRow - 3, PrintColumn + 4).Value = (Cells(PrintRow - 6,
                    PrintColumn + 3).Value + Cells(PrintRow - 5, PrintColumn + 3).Value
                    + Cells(PrintRow - 4, PrintColumn + 3).Value + Cells(PrintRow - 3,
```

```
        PrintColumn + 3).Value + Cells(PrintRow - 2, PrintColumn + 3).Value
        + Cells(PrintRow - 1, PrintColumn + 3).Value + Cells(PrintRow - 0,
        PrintColumn + 3).Value) / 7
End If
If Cells(DataRow, 1).Value > (StartTime) Then
        If MarkBinary = 0 Then
                Cells(PrintRow, PrintColumn).Value = "Target Appears"
                StartRow = PrintRow
                Cells(2, PrintColumn + 3).Value = StartRow
                MarkBinary = 1
        End If
End If
If TargetBinary = 0 Then
        If Cells(5, PrintColumn + 4).Value = 1 Then
                If Cells(DataRow, 2).Value < -1145 Then
                        TargetBinary = 1
                        Cells(3, PrintColumn + 3).Value = "Time to head
                        halfway point"
                        Cells(4, PrintColumn + 3).Value = Cells(DataRow,
                        1).Value - StartTime
                End If
        End If
        If Cells(5, PrintColumn + 4).Value = 2 Then
                If Cells(DataRow, 2).Value < -673 Then
                        TargetBinary = 1
                        Cells(3, PrintColumn + 3).Value = "Time to head
                        halfway point"
                        Cells(4, PrintColumn + 3).Value = Cells(DataRow,
                        1).Value - StartTime
                End If
        End If
        If Cells(5, PrintColumn + 4).Value = 4 Then
                If Cells(DataRow, 2).Value > 673 Then
                        TargetBinary = 1
                        Cells(3, PrintColumn + 3).Value = "Time to head
                        halfway point"
                        Cells(4, PrintColumn + 3).Value = Cells(DataRow,
                        1).Value - StartTime
                End If
        End If
        If Cells(5, PrintColumn + 4).Value = 5 Then
                If Cells(DataRow, 2).Value > 1145 Then
                        TargetBinary = 1
                        Cells(3, PrintColumn + 3).Value = "Time to head
                        halfway point"
                        Cells(4, PrintColumn + 3).Value = Cells(DataRow,
                        1).Value - StartTime
                End If
```

```
                End If
            End If
            PrintRow = PrintRow + 1
End If
‘check if the event has ended. If so, fit the bilinear model and find the start and
finish of the next event.
If Cells(DataRow + 1, 1).Value > (EndTime) Then
            'initial acceleration finder
            MaxTime = 0
            MaxVel = 0
            AccelLoop = 12
            EndRow = -1
            'find initial local maxima
            Do Until AccelLoop = 0
                AccelLoop = AccelLoop + 1
                If Application.WorksheetFunction.Count(Cells(AccelLoop,
            PrintColumn + 4)) = 1 Then
                        If Cells(5, PrintColumn + 1).Value < 3 Then
                            If (AccelLoop > StartRow) Then
                                If Cells(AccelLoop, PrintColumn + 4) < -0.3
                                Then
                                Cells(2, PrintColumn + 2).Value = "Down"
                                    If Cells(AccelLoop, PrintColumn + 4)
                                < Cells(AccelLoop + 1, PrintColumn + 4) Then
                                        If Cells(AccelLoop,
                                    PrintColumn + 4) < Cells(AccelLoop
                                    + 2, PrintColumn + 4) Then
                                            If Cells(AccelLoop,
                                        PrintColumn + 4) <
                                        Cells(AccelLoop + 3,
                                        PrintColumn + 4) Then
                                                MaxTime =
                                                Cells(AccelL
                                                oop,
                                                PrintColum
                                                n + 4).Value
                                                MaxVel =
                                                Cells(AccelL
                                                oop,
                                                PrintColum
                                                n + 1).Value
                                                EndRow =
                                                AccelLoop
                                                Cells(2,
                                                PrintColum
                                                n + 4).Value
                                                = EndRow
```

```
                                        AccelLoop =
                                        0
                                    End If
                                End If
                            End If
                        End If
                    End If
                Else
                    If (AccelLoop > StartRow) Then
                        If Cells(AccelLoop, PrintColumn + 4) > 0.3
                        Then
                            Cells(2, PrintColumn + 2).Value =
                            "Up"
                            If Cells(AccelLoop, PrintColumn + 4)
                            > Cells(AccelLoop + 1, PrintColumn +
                            4) Then
                                If Cells(AccelLoop,
                                PrintColumn + 4) >
                                Cells(AccelLoop + 2,
                                PrintColumn + 4) Then
                                    If Cells(AccelLoop,
                                    PrintColumn + 4) >
                                    Cells(AccelLoop + 3,
                                    PrintColumn + 4)
                                    Then
                                        MaxTime =
                                        Cells(AccelL
                                        oop,
                                        PrintColum
                                        n + 4).Value
                                        MaxVel =
                                        Cells(AccelL
                                        oop,
                                        PrintColum
                                        n + 1).Value
                                        EndRow =
                                        AccelLoop
                                        Cells(2,
                                        PrintColum
                                        n + 4).Value
                                        = EndRow
                                        AccelLoop =
                                        0
                                    End If
                                End If
                            End If
                        End If
                    End If
```

```
                End If
        Else
                AccelLoop = 0
                Cells(3, PrintColumn + 3).Value = "Error"
                Cells(4, PrintColumn + 3).Value = "Failed to find first peak"
        End If
Loop
'find elbow point
If StartRow > 0 Then
        If EndRow > 0 Then
                SquaredError = 0.00001
                MinSquaredError = 99999.99999
                For i = StartRow + 1 To EndRow - 1
                        SquaredError = 0
                        fitAverage =
                        Application.Average(Range(Cells(StartRow,
                        PrintColumn + 4), Cells(i, PrintColumn + 4)))
                        Slope = (Cells(EndRow, PrintColumn + 4).Value -
                        fitAverage) / (Cells(EndRow, PrintColumn + 1).Value
                        - Cells(i, PrintColumn + 1).Value)
                        For j = StartRow To i
                                SquaredError = SquaredError + ((Cells(j,
                                PrintColumn + 4).Value - fitAverage) *
                                (Cells(j, PrintColumn + 4).Value -
                                fitAverage))
                        Next j
                        For j = i + 1 To EndRow
                                PredictedValue = fitAverage + (Cells(j,
                                PrintColumn + 1).Value - Cells(i,
                                PrintColumn + 1).Value) * Slope
                                SquaredError = SquaredError + ((Cells(j,
                                PrintColumn + 4).Value - PredictedValue) *
                                (Cells(j, PrintColumn + 4).Value -
                                PredictedValue))
                        Next j
                        Cells(5, PrintColumn + 3).Value = fitAverage
                        Cells(5, PrintColumn + 4).Value = Slope
                        If (SquaredError < MinSquaredError) Then
                                MinSquaredError = SquaredError
                                Cells(5, PrintColumn + 5).Value =
                                SquaredError
                                Cells(3, PrintColumn + 3).Value = "Onset
                                Time"
                                Cells(4, PrintColumn + 3).Value = Cells(i,
                                PrintColumn + 1).Value
                                Cells(3, PrintColumn + 4).Value = "End Time"
                                Cells(4, PrintColumn + 4).Value =
                                Cells(EndRow, PrintColumn + 1).Value
```

```
                                    Cells(3, PrintColumn + 5).Value = "Initial
                                    Spike Velocity Change"
                                    Cells(4, PrintColumn + 5).Value =
                                    Cells(EndRow, PrintColumn + 4).Value -
                                    Cells(i, PrintColumn + 4).Value
                                    For j = StartRow To i
                                        Cells(j, PrintColumn + 5).Value =
                                    fitAverage
                                    Next j
                                    For j = i + 1 To EndRow
                                        PredictedValue = fitAverage +
                                        (Cells(j, PrintColumn + 1).Value -
                                        Cells(i, PrintColumn + 1).Value) *
                                        Slope
                                        Cells(j, PrintColumn + 5).Value =
                                        PredictedValue
                                    Next j
                            End If
                    Next i
            End If
End If
'initial acceleration finder end
PrintColumn = PrintColumn + 6
PrintRow = 6
MarkBinary = 0
StartRow = -1
Cells(6, PrintColumn + 3).Value = "Velocity"
Cells(6, PrintColumn + 4).Value = ""
Cells(7, PrintColumn + 4).Value = "Smoothed Velocity (100ms)"
'Beginning of new event code
EventBinary = 0
Do Until EventBinary = 1
        If Application.WorksheetFunction.CountA(Cells(EventRow, 7)) = 1
Then
                If Cells(EventRow, 8).Value = 1 Then
                        StartTime = Cells(EventRow, 7).Value
                        Cells(3, PrintColumn).Value = "Sound Event"
                        Cells(4, PrintColumn).Value = "Visual Event"
                        Cells(5, PrintColumn).Value = "Position"
                        Cells(3, PrintColumn + 1).Value = Cells(EventRow,
                        16).Value
                        Cells(4, PrintColumn + 1).Value = Cells(EventRow,
                        17).Value
                        Cells(5, PrintColumn + 1).Value = Cells(EventRow,
                        18).Value
                        EventBinary = 1
                Else
                        EventRow = EventRow + 1
```

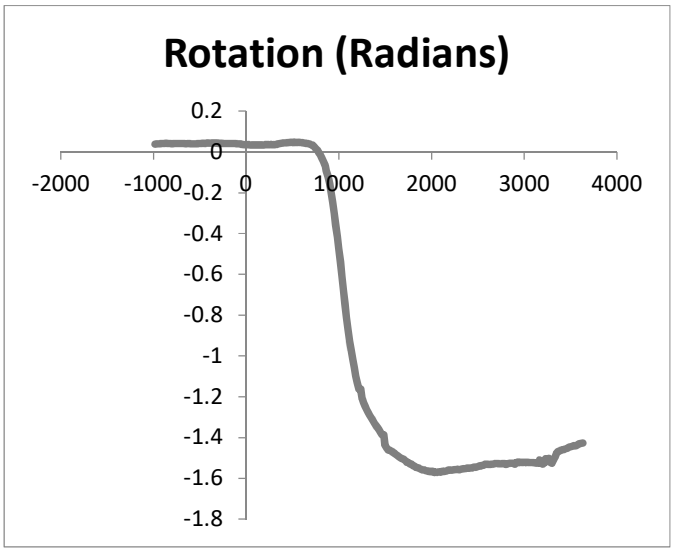
```
                        End If
                    Else
                        EventBinary = 1
                        EndBinary = 1
                    End If
                Loop

                EventBinary = 0
                Do Until EventBinary = 1
                    If Application.WorksheetFunction.CountA(Cells(EventRow, 7)) = 1
                Then
                        If Cells(EventRow, 8).Value = 3 Then
                            EndTime = Cells(EventRow, 7).Value
                            EventBinary = 1
                            Cells(3, PrintColumn + 2).Value = "Time to Hit"
                            Cells(4, PrintColumn + 2).Value = EndTime -
                        StartTime
                        Else
                            EventRow = EventRow + 1
                        End If
                    Else
                        EndBinary = 1
                        EventBinary = 1
                    End If
                Loop
                'end of new event code
            End If
        Else
            EndBinary = 1
        End If
Loop

End Sub
```

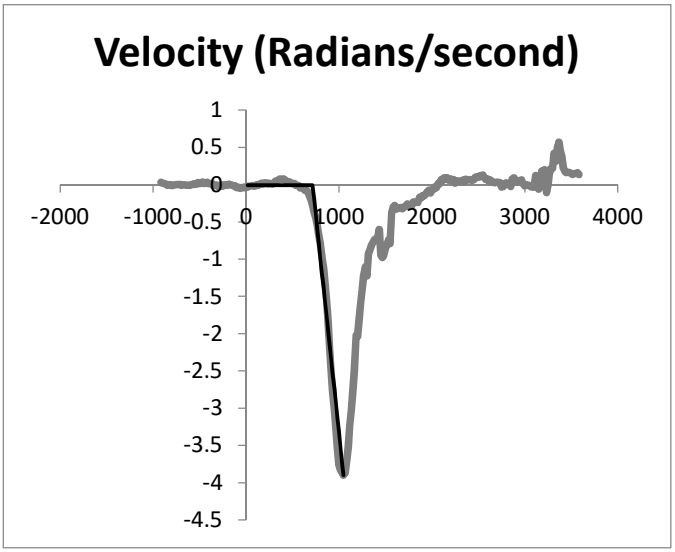
//

This macro marks the event combination in the top left hand corner. It pulls from the event log the duration of the event and records that second. Start time and end time are when the participant begins and completes their first major head rotation following each event. Inspection of the response curves found that participants in general made a linear acceleration during their initial movement and so a bilinear fit was applied to the smoothed velocity. Velocity smoothing was necessary because the tracking updates and main program could occur out of phase and therefore would cause velocity to spike up and down from the zero mark when no update came through. An example of the bilinear model fit can be seen below as the black line on the velocity graph. The model is fit using least squares estimation and the start point is the time at which the acceleration begins, and the end points is when velocity reaches its peak. Time until the target was marked and the start time for the participant's movement were used in the final analysis.

0	Shot Time	Start Time	End Time	Initial Spike Velocity Change
2	3632	716	1049	-3.62473
1				

Time	HorRotate		
-983	0.038174	**Velocity**	
-966	0.039428	0.073792	**Smoothed Velocity**
-950	0.039225	-0.0127	(100ms)
-933	0.040143	0.053989	
-916	0.040966	0.048392	0.031661
-900	0.040922	-0.0027	0.017578
-883	0.041406	0.028468	0.01501
-866	0.041957	0.032389	0.006778
-850	0.04156	-0.02479	-0.00793
-833	0.041039	-0.03068	-0.00576
-816	0.040977	-0.00364	-0.00281
-800	0.040104	-0.05458	-0.01164
-783	0.040316	0.01248	-0.01005
-766	0.041152	0.049171	-0.00152
-750	0.04068	-0.02949	-0.00336
-733	0.040449	-0.0136	0.003431
-716	0.040941	0.028978	0.003582
-700	0.040677	-0.01649	-0.00863
-683	0.040558	-0.00703	-0.00346
-666	0.040788	0.013538	9.49E-05
-650	0.040208	-0.03629	-0.00626
-633	0.040322	0.006708	-0.00931
-616	0.040513	0.011252	-0.004
-600	0.040265	-0.01551	-0.00867
-583	0.039622	-0.03783	-0.00324
-567	0.040103	0.030116	0.010188
-550	0.039778	-0.01911	0.001394
-533	0.039807	0.001682	0.00501
-517	0.041419	0.100718	0.025508
-500	0.040563	-0.0503	0.014421
-483	0.04073	0.009799	0.021481
-467	0.042421	0.105656	0.035248
-450	0.041613	-0.04749	0.0153
-433	0.042128	0.030305	0.024135
-417	0.043697	0.098052	0.030217
-400	0.043036	-0.03892	0.011704
-383	0.043232	0.011544	0.010643
-367	0.04407	0.052378	0.014573
-350	0.043663	-0.02394	-0.01822





-333	0.042729	-0.05493	-0.00779	
-317	0.043654	0.057818	-0.00573	
-300	0.041419	-0.13147	-0.02933	
-283	0.041999	0.034084	-0.0199	
-267	0.042413	0.025909	-0.00996	
-250	0.040496	-0.11278	-0.02165	
-233	0.041211	0.042032	-0.00285	
-217	0.041445	0.014684	-0.00176	
-200	0.041037	-0.02404	-0.00903	
-183	0.04104	0.000182	0.006248	
-167	0.041707	0.041672	-0.0014	
-150	0.041282	-0.02497	-0.01068	
-133	0.041183	-0.00583	-0.01692	
-117	0.040999	-0.01149	-0.01676	
-100	0.040145	-0.05028	-0.04149	
-83	0.038993	-0.06773	-0.03875	
-67	0.039013	0.001271	-0.04786	
-50	0.036779	-0.13142	-0.043	
-33	0.036681	-0.00578	-0.04196	
-17	0.035568	-0.06957	-0.03505	
0	0.03595	0.022469	-0.04115	**Bilinear Fit**
17	0.035219	-0.043	-0.0221	-0.00968
33	0.03491	-0.01934	-0.02681	-0.00968
50	0.034205	-0.04143	-0.01859	-0.00968
67	0.034238	0.001923	-0.01376	-0.00968
83	0.033618	-0.03876	-0.01959	-0.00968
100	0.033414	-0.01199	-0.01554	-0.00968
117	0.034371	0.056287	0.000581	-0.00968
133	0.03303	-0.08381	-0.00796	-0.00968
150	0.033182	0.008965	0.009921	-0.00968
167	0.034397	0.071447	0.019092	-0.00968
183	0.033471	-0.05789	0.007751	-0.00968
200	0.03494	0.086439	0.022572	-0.00968
217	0.035828	0.052214	0.024746	-0.00968
233	0.035458	-0.0231	0.009787	-0.00968
250	0.035797	0.019935	0.019443	-0.00968
267	0.036208	0.024184	0.010157	-0.00968
283	0.035676	-0.03327	0.007081	-0.00968
300	0.035841	0.009696	0.028205	-0.00968
316	0.036184	0.021436	0.04561	-0.00968
333	0.036705	0.030679	0.050056	-0.00968
350	0.038826	0.124768	0.065034	-0.00968
366	0.041095	0.141773	0.074069	-0.00968
383	0.042035	0.055309	0.069971	-0.00968
400	0.043252	0.071578	0.073613	-0.00968
416	0.044419	0.072938	0.063378	-0.00968

433	0.044296	-0.00725	0.045547	-0.00968
450	0.04525	0.056171	0.035561	-0.00968
466	0.0461	0.053125	0.03834	-0.00968
483	0.046389	0.016954	0.015697	-0.00968
500	0.046141	-0.01459	0.014587	-0.00968
516	0.047597	0.09103	0.013947	-0.00968
533	0.046143	-0.08557	-0.01034	-0.00968
550	0.045887	-0.01502	-0.0105	-0.00968
566	0.046714	0.051691	-0.01188	-0.00968
583	0.044727	-0.11691	-0.04395	-0.00968
600	0.044996	0.015833	-0.04246	-0.00968
616	0.044608	-0.02423	-0.03854	-0.00968
633	0.04234	-0.13345	-0.07762	-0.00968
650	0.041063	-0.0751	-0.07559	-0.00968
666	0.041262	0.012415	-0.10243	-0.00968
683	0.037489	-0.22189	-0.16032	-0.00968
700	0.035744	-0.10267	-0.20857	-0.00968
716	0.032991	-0.17205	-0.27439	-0.00968
733	0.02569	-0.42949	-0.37295	-0.20824
749	0.018151	-0.47119	-0.44511	-0.39512
766	0.009041	-0.53587	-0.55997	-0.59368
783	-0.00248	-0.67748	-0.67005	-0.79224
800	-0.01484	-0.72704	-0.78632	-0.9908
816	-0.02934	-0.90665	-0.95789	-1.17768
833	-0.04537	-0.94262	-1.12968	-1.37624
850	-0.0665	-1.2434	-1.31561	-1.5748
866	-0.09326	-1.67221	-1.56413	-1.76168
883	-0.12281	-1.73835	-1.84258	-1.96024
900	-0.15646	-1.97901	-2.18435	-2.1588
916	-0.19592	-2.46665	-2.52022	-2.34568
933	-0.24447	-2.85581	-2.77458	-2.54424
950	-0.30117	-3.335	-3.02531	-2.7428
966	-0.35868	-3.59452	-3.30326	-2.92968
983	-0.41737	-3.45275	-3.52838	-3.12824
999	-0.47327	-3.49346	-3.76448	-3.31512
1016	-0.53999	-3.92465	-3.82786	-3.51368
1033	-0.60871	-4.04249	-3.86526	-3.71224
1049	-0.68085	-4.50849	-3.89912	-3.89912
1066	-0.74508	-3.77868	-3.86743	
1083	-0.81064	-3.85626	-3.6967	
1099	-0.86968	-3.68979	-3.5216	
1116	-0.9253	-3.27166	-3.23513	
1133	-0.9717	-2.72953	-3.02797	
1149	-1.01677	-2.81676	-2.78114	
1166	-1.05932	-2.50324	-2.49032	
1183	-1.09891	-2.32853	-2.02294	

1199	-1.13296	-2.12844	-2.03377
1216	-1.16108	-1.65406	-1.80292
1233	-1.16108	0	-1.59667
1250	-1.20877	-2.80535	-1.40756
1266	-1.22798	-1.20081	-1.22286
1283	-1.246	-1.05947	-1.10227
1299	-1.26207	-1.00481	-1.22531
1316	-1.27628	-0.83553	-0.92739
1333	-1.29005	-0.80994	-0.87256
1349	-1.30383	-0.86125	-0.81187
1366	-1.31606	-0.71994	-0.7727
1383	-1.32995	-0.81694	-0.74083
1399	-1.34011	-0.63469	-0.72703
1416	-1.35253	-0.73059	-0.70424
1433	-1.36294	-0.61247	-0.60139
1449	-1.37435	-0.71331	-0.94339
1466	-1.38628	-0.70171	-0.97837
1483	-1.38628	0	-0.95045
1499	-1.43766	-3.211	-0.86352
1516	-1.45261	-0.87953	-0.79903
1533	-1.46171	-0.53512	-0.74038
1549	-1.46177	-0.004	-0.79066
1566	-1.46622	-0.26182	-0.381
1583	-1.47117	-0.29118	-0.30533
1599	-1.4768	-0.352	-0.27936
1616	-1.48264	-0.34335	-0.32744
1633	-1.48859	-0.34982	-0.32036
1649	-1.49424	-0.35337	-0.31121
1666	-1.50003	-0.34053	-0.30864
1683	-1.50364	-0.21224	-0.32525
1699	-1.50727	-0.22719	-0.29844
1716	-1.51295	-0.334	-0.29244
1733	-1.52076	-0.45959	-0.25904
1749	-1.52336	-0.16219	-0.2977
1766	-1.52865	-0.31135	-0.28621
1783	-1.53047	-0.10671	-0.28151
1799	-1.53819	-0.48287	-0.2353
1816	-1.54069	-0.14676	-0.2407
1833	-1.54581	-0.30112	-0.22959
1849	-1.54798	-0.13606	-0.23787
1866	-1.55138	-0.2	-0.17154
1882	-1.55512	-0.23363	-0.18061
1899	-1.55792	-0.16465	-0.14976
1916	-1.55824	-0.01853	-0.14901
1932	-1.5616	-0.21031	-0.12754
1949	-1.56305	-0.08518	-0.10292

1966	-1.56527	-0.13076	-0.07229
1982	-1.56607	-0.04975	-0.11382
1999	-1.56711	-0.06124	-0.07243
2016	-1.56626	0.049706	-0.06597
2032	-1.57121	-0.30919	-0.02803
2049	-1.56986	0.079412	-0.04065
2066	-1.57054	-0.04	0.016753
2082	-1.56838	0.134812	0.009652
2099	-1.57073	-0.13806	0.065847
2116	-1.56494	0.340588	0.056335
2132	-1.56494	0	0.090183
2149	-1.56351	0.084176	0.072596
2166	-1.56329	0.012824	0.090907
2182	-1.56014	0.196937	0.060787
2199	-1.55994	0.011706	0.074493
2216	-1.56011	-0.00988	0.057275
2232	-1.55803	0.12975	0.065862
2249	-1.5564	0.095941	0.019888
2266	-1.55702	-0.03635	0.034376
2282	-1.55585	0.072938	0.050457
2299	-1.55798	-0.12488	0.048215
2316	-1.55605	0.113118	0.038064
2332	-1.55441	0.102687	0.060731
2349	-1.55247	0.114059	0.037975
2366	-1.55205	0.024882	0.070294
2382	-1.55009	0.122312	0.062134
2399	-1.55156	-0.08635	0.068213
2416	-1.54984	0.101353	0.059686
2432	-1.54894	0.056	0.056132
2449	-1.54647	0.145235	0.067104
2465	-1.5456	0.054375	0.100364
2482	-1.5456	0	0.090555
2499	-1.54222	0.199118	0.110732
2516	-1.53973	0.146471	0.103716
2532	-1.5392	0.032687	0.124662
2549	-1.53585	0.197235	0.129309
2565	-1.53431	0.096125	0.092587
2582	-1.5309	0.201	0.066806
2599	-1.53034	0.032529	0.073834
2615	-1.53127	-0.05794	0.045505
2632	-1.53185	-0.034	0.060681
2649	-1.53046	0.081882	0.030311
2665	-1.53047	-0.00106	0.027646
2682	-1.52703	0.202353	0.026612
2699	-1.52723	-0.01159	0.031477
2715	-1.52701	0.013875	0.028084

2732	-1.52812	-0.06518	0.024941
2749	-1.52811	5.88E-05	-0.03287
2765	-1.52718	0.058125	-0.03018
2782	-1.52758	-0.02306	-0.00284
2799	-1.53102	-0.20229	0.024196
2815	-1.5309	0.007188	0.016116
2832	-1.52741	0.205294	0.012384
2849	-1.5253	0.124059	-0.02448
2865	-1.52621	-0.0565	0.074551
2882	-1.52566	0.032	0.090037
2899	-1.53044	-0.28112	0.044289
2915	-1.52259	0.490937	0.034085
2932	-1.52062	0.115588	0.047206
2949	-1.52257	-0.11494	0.044164
2965	-1.52173	0.052625	0.064708
2982	-1.52113	0.035353	0.008902
2999	-1.52095	0.010706	-0.01882
3015	-1.52315	-0.13731	-0.00159
3032	-1.52144	0.100294	-0.00699
3049	-1.52278	-0.07847	-0.01627
3065	-1.52268	0.005688	-0.02979
3082	-1.52243	0.014824	-0.02827
3099	-1.52294	-0.02965	-0.03549
3115	-1.52428	-0.08387	0.123293
3132	-1.52643	-0.12671	-0.02682
3149	-1.52559	0.049765	-0.06081
3165	-1.50906	1.033	-0.00968
3182	-1.52683	-1.04512	0.181705
3198	-1.53039	-0.22306	0.073976
3215	-1.52481	0.328235	0.198581
3232	-1.50347	1.255824	-0.10828
3248	-1.51756	-0.88081	-0.00172
3265	-1.50188	0.922	0.14621
3282	-1.52084	-1.11506	0.21737
3298	-1.52563	-0.29919	0.216984
3315	-1.51182	0.812471	0.419689
3332	-1.49777	0.826353	0.31452
3348	-1.47772	1.253125	0.49619
3365	-1.46857	0.538118	0.568939
3382	-1.46541	0.185824	0.460158
3398	-1.4629	0.156625	0.371982
3415	-1.45933	0.210059	0.218989
3432	-1.45847	0.051	0.173721
3448	-1.45512	0.209125	0.161201
3465	-1.45202	0.182176	0.16233
3482	-1.44826	0.221235	0.158658

3498	-1.44669	0.098187	0.153506
3515	-1.44389	0.164529	0.134497
3532	-1.44076	0.184353	0.142951
3548	-1.44052	0.014938	0.151346
3565	-1.43923	0.076059	0.158689
3582	-1.43513	0.241353	0.137478
3598	-1.43065	0.28	
3615	-1.4281	0.149588	
3632	-1.42783	0.016059	

Appendix 3.5: Paper Materials for Experiment 3

Consent form for Experiment 3.

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Redirecting Attention in Real-World Space with Wearable Computers.

Consent Form for Participants

I have been given a full explanation of this project, I understand what is required of me if I agree to take part in the research and I have had the opportunity to ask questions.

I understand that participation is voluntary and I may withdraw at any time without penalty. I understand that withdrawal of participation will also include the withdrawal of any information I have provided should this remain practically achievable.

I understand that I can request to stop the test at any time if I feel any discomfort for any reason.

I understand that any information or opinions I provide will be kept confidential to the researcher and their senior supervisor and that any published or reported results will not identify me or the other participants. I understand that the data may be used in a thesis which is a public document and will be available through the UC Library.

I understand that all identifying data collected for the study will be kept in locked and secure facilities and will be destroyed after ten years.

I understand that I can contact the researcher Matt Ward (matt.ward@pg.canterbury.ac.nz) or supervisor Dr Deak Helton (deak.helton@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Human Ethics Committee, Private Bag 4800, Christchurch (humanethics@canterbury.ac.nz)

By signing below, I agree to participate in this research project.

Name:

Date:

Signature:

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Information Sheet for participants.

You are being asked to participate in the study 'Redirecting Attention in Real-World Space with Wearable Computers'. This study will require you to read alternating sections of text from near and far screen while occasional detecting and marking targets with virtual paintballs. Participation will take approximately 90 minutes with two 5 minute breaks.

There are no expected risks to participants' physical or mental health, although if the task is uncomfortable for any reason, you may request for the researcher to stop the test without penalty.

A copy of the results will be provided to the class after the experiment has concluded.

Participation is voluntary and you have the right to withdraw at any stage without penalty. If you withdraw during the test, all information relating to you will be destroyed. To ensure anonymity and confidentiality, all digital instances of your name will be replaced with a randomly assigned participant number and no other identifying data will be stored. It will therefore be impossible to remove your data from the pool after it has been collected. Paper copies of any testing materials will also use your randomly assigned participant number. A consent form bearing your name will be stored for a period of ten years in a locked filing cabinet and will only be made available to the primary researchers Matt Ward and Amit Barde, and their senior supervisor, Dr Deak Helton, as well as the University of Canterbury Human Ethics Committee if such disclosure becomes necessary.

The results of the project might be used in a doctoral thesis and may also be published. A thesis is a public document and will be available through the UC Library.

This project is being carried out by Matt Ward (matt.ward@pg.canterbury.ac.nz) and Amit Barde (amit.barde@pg.canterbury.ac.nz) under the supervision of Dr Deak Helton, who can be contacted at deak.helton@canterbury.ac.nz if you have any concerns about participation in this project.

This project has been reviewed approved by the University of Canterbury Psychology department and the UCHEC Low Risk Approval process. Participants should address any complaints to The Chair, Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz). If you agree to participate in the study, you are asked to complete the consent form before continuing with the test.

Appendix 4.0: Ethics, Programming, Data Analysis and Paper Materials for Experiment 4

The University of Canterbury Human Ethic Approval for this experiment is presented in Appendix 4.1.

Experiment 4's core program was written in the Unity engine (Unity3d.com, 2015) using the C# language. The D-track software (Ar-tracking.com, 2015) provided the tracking data and was transferred to the main program's desktop PC via the VRPN software (GitHub, 2015). See Chapter 4 for the block diagram detailing the links between electronic components (Figure 4.1).

Appendix 4.2 contains the core experiment program. Specialist functions controlling the head tracking, gun tracking, and data recording are shared with experiment 3, along with the VRPN receiver library, with minor alterations. The key difference is the ability to re-center the projection for the participant in order to be able to calibrate the projections for situations where the head tracker would not sit squarely on a participant.

Appendix 4.3 contains HMD display program. This experiment sees a return to the use of the Google Glass device, this is because a work around was found for the Glass' unstable wireless connection. A bridging program through an Android Nexus 5 smartphone connected to the desktop through a wireless connection and to the Glass through a Bluetooth connection. This bridging method was used for the remainder of the experiments. This experiment's bridging program is available below the main display program in appendix 4.3.

Appendix 4.4 shows example data output files from the experiment and the process used to extract useful data. Macros used in the process are written in the Visual Basic language.

Appendix 4.5 contains the paper materials used in this experiment. These include an information sheet, a consent form and a two page post-experiment questionnaire.

Appendix 4.1: Human Ethics Approval for Experiment 4

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE

Secretary, Rebecca Robinson
Telephone: +64 03 364 2987, Extn 45588
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2016/10/LR-PS

10 March 2016

Matt Ward
Psychology
UNIVERSITY OF CANTERBURY

Dear Matt

Thank you for submitting your low risk application to the Human Ethics Committee for the research proposal titled "Assisting Visual Search with Head Mounted Displays".

I am pleased to advise that this application has been reviewed and approved.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 9th March 2016.

With best wishes for your project.

Yours sincerely

Lindsey MacDonald
Chair, Human Ethics Committee

Appendix 4.2: Main Controller Program for Experiment 4.

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

public class screenTransfer : MessageBase
{
	public int transferScreenInt;
}

public class valueTransfer : MessageBase
{
	public float transferValue;
}

public class ExperimentController : MonoBehaviour
{

	public GUIStyle style;
	public GUIStyle styleSmall;
	public HeadTracking headtrack;
	public GunTracking guntrack;
	public Recording record;
	public Transform targetDot;
	public int experimentStage = 0;
	public float timeFloat = 0;
	public float timeFloat2 = 0;
	public SpriteRenderer[,] stimuliArray = new SpriteRenderer[30, 10];
	public SpriteRenderer stimuliObject;
	int port = 25000;
	public int[] conditionOrder = { 0, 0, 0, 0, 0, 0, 0 };
	public int currentCondition = 0;
	public int trialCounter = 0;
	int fillingConditions = 0;
	public float targetDistance = 0;
	public float xDistance = 0;
	public float yDistance = 0;
	public float zDistance = 0;
	public float hDistance = 0;
	public float gunDistance = 0;
	public bool clicked = false;
	public bool keyPressed = false;

	int myReliableChannelId;
	int socketId;
	int socketPort = 8888;
	int connectionId;
	int connectionCounter = 0;
	int connectionCounter2 = 0;
	int reconnectionAttempts = 0;
	bool connected = false;
	string clientIP = "";

	int outgoingScreen;
	int outgoingValue;
	int outgoingValue2;

	public Sprite sprite3;
```

```
public Sprite spriteE;
public Sprite sprite8;
public Sprite spriteBlank;
public Sprite spriteDot;

public int[] screenArray = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 };
public int[] LRArray = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
public int[] xPosArray = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 };
public int[] yPosArray = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 };


//These two functions, Awake() and Start(), run once at the beginning of the
program. The Start() function sets up the network, creates the three 10x10
grids of targets on the three virtual screens and then makes truns all the
targets blank.
void Awake()
{
	Application.targetFrameRate = 60;
}


void Start ()
{
	initializeNetwork();
	//NetworkServer.Listen("10.32.21.161", 6682);
	for (int i = 10; i < 20; i++)
	{
		for (int j = 0; j < 10; j++)
		{
			stimuliArray[i, j] = Instantiate(stimuliObject, new
			Vector3((i - 10) * 2.2f - 11f + 1.1f, -j * 1.5f + 7.5f -
			0.75f, 21), new Quaternion(0, 0, 0, 1)) as SpriteRenderer;
		}
	}
	for (int i = 0; i < 10; i++)
	{
		for (int j = 0; j < 10; j++)
		{
			stimuliArray[i, j] = Instantiate(stimuliObject, new
			Vector3(Mathf.Cos(Mathf.PI / 3) * ((i) * 2.2f - 11f +
			1.1f) - 18.3f, -j * 1.5f + 7.5f - 0.75f,
			Mathf.Sin(Mathf.PI / 3) * ((i) * 2.2f - 11f + 1.1f) +
			10.5f), new Quaternion(0, -0.6f, 0, 1)) as SpriteRenderer;
		}
	}

	for (int i = 20; i < 30; i++)
	{
		for (int j = 0; j < 10; j++)
		{
			stimuliArray[49 - i, j] = Instantiate(stimuliObject, new
			Vector3(Mathf.Cos(Mathf.PI / 3) * ((29 - i) * 2.2f - 11f +
			1.1f) + 18.3f, -j * 1.5f + 7.5f - 0.75f,
```

```
                        Mathf.Sin(Mathf.PI / 3) * ((i - 20) * 2.2f - 11f + 1.1f) +
                        10.5f), new Quaternion(0, 0.6f, 0, 1)) as SpriteRenderer;
                }
        }
        BlankStimuli();
}

//initializeNetwork() is called in the Start() function. It sets up the new
socket connection style, although the actual connection occurs later.
void initializeNetwork()
{
        NetworkTransport.Init();
        ConnectionConfig config = new ConnectionConfig();
        myReliableChannelId = config.AddChannel(QosType.StateUpdate);
        int maxConnections = 10;
        HostTopology topology = new HostTopology(config, maxConnections);
        socketId = NetworkTransport.AddHost(topology, socketPort);
        Debug.Log("Socket Open. SocketId is: " + socketId);
}

//The Update() function is called every frame of the program and runs the
majority of the timings and actions.
void Update ()
{
        //Checks of incoming messages (connections/disconnections) and if a
        reconnect timer is active, counts it down until it is time to try a
        reconnect.
        updateNetwork();
        if (connectionCounter > 1)
        {
                connectionCounter = connectionCounter - 1;
        }
        else if (connectionCounter == 1)
        {
                reconnectionAttempts++;
                connectionCounter = 0;
                Disconnect();
                Connect();
        }

        //Two keys are active during the experiment. ‘R’ orders an immediate
        reconnect attempt if the connection breaks. ‘C’ marks an error code in
        the event log. Used to make trials in which a disconnection or other
        interruption occurs, such as someone knocking on the door, etc.
        if (Input.GetKeyDown(KeyCode.C))
        {
                System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
        ",9," + System.Environment.NewLine);
        }
        if (Input.GetKeyDown(KeyCode.R))
        {
                Disconnect();
                Connect();
                reconnectionAttempts++;
        }

        outgoingScreen = 0;
        outgoingValue = 0;
        outgoingValue2 = 0;

        //This switch determines which screen of the experiment is to be
        displayed.
```

```
switch (experimentStage)
{
case 0: //Calibration Screen
//This first screen is the calibration screen. The participant is asked
to look directly at a marked point in the middle of the center screen.
At the same time the experiment enters the IP address for the bridging
program device. When the experimenter presses the ‘Enter’ key the head
tracker is calibrated and the first connection attempt is made.
       InputIP();
       if (Input.GetKeyDown(KeyCode.Return) == true)
       {
              headtrack.Calibrate();
              experimentStage = 1;
              Connect();
              timeFloat = Time.time;
              targetDot.position = new Vector3(0, 0, 25);
       }
       break;
case 1: //Wait Screen
//This second screen is a waiting screen displayed for 2 seconds.
       if (Time.time > timeFloat + 2)
       {
              SendBlankMessage();
              if (fillingConditions == 0)
              {
                     experimentStage = 2;
              }
              else
              {
                     experimentStage = 3;
              }
       }
       break;
case 2: //Condition Screen
//This screen waits for the experimenter to enter the condition order
which was randomised before the experiment begins (using an extended
Latin square design). Once all 7 blocks have been assigned a condition,
the experimenter can press the ‘Enter’ key to progress.
       if (fillingConditions < conditionOrder.Length)
       {
              if (Input.GetKeyDown(KeyCode.Alpha0) == true)
              {
                     conditionOrder[fillingConditions] = 0;
                     fillingConditions++;
              }
              if (Input.GetKeyDown(KeyCode.Alpha1) == true)
              {
                     conditionOrder[fillingConditions] = 1;
                     fillingConditions++;
              }
              if (Input.GetKeyDown(KeyCode.Alpha2) == true)
              {
                     conditionOrder[fillingConditions] = 2;
                     fillingConditions++;
              }
              if (Input.GetKeyDown(KeyCode.Alpha3) == true)
              {
                     conditionOrder[fillingConditions] = 3;
                     fillingConditions++;
              }
              if (Input.GetKeyDown(KeyCode.Alpha4) == true)
              {
```

```
                        conditionOrder[fillingConditions] = 4;
                        fillingConditions++;
                }
                if (Input.GetKeyDown(KeyCode.Alpha5) == true)
                {
                        conditionOrder[fillingConditions] = 5;
                        fillingConditions++;
                }
                if (Input.GetKeyDown(KeyCode.Alpha6) == true)
                {
                        conditionOrder[fillingConditions] = 6;
                        fillingConditions++;
                }
                if (Input.GetKeyDown(KeyCode.Alpha7) == true)
                {
                        conditionOrder[fillingConditions] = 7;
                        fillingConditions++;
                }
                if (Input.GetKeyDown(KeyCode.Alpha8) == true)
                {
                        conditionOrder[fillingConditions] = 8;
                        fillingConditions++;
                }
                if (Input.GetKeyDown(KeyCode.Alpha9) == true)
                {
                        conditionOrder[fillingConditions] = 9;
                        fillingConditions++;
                }
        }
        else
        {
                if (Input.GetKeyDown(KeyCode.Return) == true)
                {
                        experimentStage = 3;
                        timeFloat = Time.time;
                }
        }
        if (Input.GetKeyDown(KeyCode.Backspace) == true)
        {
                conditionOrder[0] = 0;
                conditionOrder[1] = 0;
                conditionOrder[2] = 0;
                conditionOrder[3] = 0;
                conditionOrder[4] = 0;
                conditionOrder[5] = 0;
                conditionOrder[6] = 0;
                fillingConditions = 0;
        }
        break;
case 3: //Wait Screen, Displays block number for the participant
//A second 2-second wait screen
        if (Time.time > timeFloat + 2)
        {
                experimentStage = conditionOrder[currentCondition] + 10;
        }
        break;
case 10: //Blank Condition Screen
//This screen only appears if no condition was not assigned for a block
        if (Input.GetKeyDown(KeyCode.Return) == true)
        {
                currentCondition++;
                experimentStage = 3;
```

```
        }
        break;
case 11:
case 12:
case 13:
case 14:
case 15:
case 16:
case 17:
case 18:
case 19: //Info Screens
//These screens are displayed before each block of trials and they have
a written description of the current block's cue type. These
instructions are also given verbally by the experimenter.
        if (Input.GetKeyDown(KeyCode.Return) == true)
        {
                experimentStage = 21;
                timeFloat = Time.time;
                timeFloat2 = Time.time;
                randomizeArrays();
                trialCounter = -1;
        }
        break;
case 20: //Trial screen
//This is the screen during each trial. Each frame it calculates the
various distances between the projected gun marker and the current
trial's correct target. If the cue is dynamic, this information is then
synchronised to the Glass. When the trigger on the gun is pulled, the
shot is recorded and if it is close to the correct target, the targets
are retuned to dots and program moves to the inter-trial pause.
        xDistance = stimuliArray[screenArray[trialCounter % 60] * 10 +
        xPosArray[trialCounter % 60], yPosArray[trialCounter %
        60]].gameObject.transform.position.x - headtrack.projectedX /
        100;
        yDistance = stimuliArray[screenArray[trialCounter % 60] * 10 +
        xPosArray[trialCounter % 60], yPosArray[trialCounter %
        60]].gameObject.transform.position.y - headtrack.projectedY /
        100;
        zDistance = stimuliArray[screenArray[trialCounter % 60] * 10 +
        xPosArray[trialCounter % 60], yPosArray[trialCounter %
        60]].gameObject.transform.position.z - headtrack.projectedZ /
        100;
        if (xDistance > 0)
        {
                hDistance = Mathf.Sqrt(xDistance * xDistance + zDistance *
                zDistance);
        }
        else
        {
                hDistance = -Mathf.Sqrt(xDistance * xDistance + zDistance
                * zDistance);
        }
        targetDistance = Mathf.Sqrt(hDistance * hDistance + yDistance *
        yDistance);
        gunDistance = (stimuliArray[screenArray[trialCounter % 60] * 10 +
        xPosArray[trialCounter % 60], yPosArray[trialCounter %
        60]].gameObject.transform.position - new
        Vector3(guntrack.projectedX / 100, guntrack.projectedY / 100,
        guntrack.projectedZ / 100)).magnitude;
        if (conditionOrder[currentCondition] == 3 ||
        conditionOrder[currentCondition] == 4 ||
```

```
        conditionOrder[currentCondition] == 5 ||
        conditionOrder[currentCondition] == 6)
        {
                SendScreenMessage();
        }
        if (VRPN.vrpnGetButton(0))
        {
                if (clicked == false)
                {
                        clicked = true;
                        if (gunDistance < 3)
                        {
                                experimentStage = 21;
                                DotStimuli();
                                SendBlankMessage();
                                timeFloat = Time.time;
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",2,,,," +
                                currentCondition + "," +
                                conditionOrder[currentCondition] + "," +
                                trialCounter + "," + gunDistance + "," +
                                targetDistance + "," + hDistance + "," +
                                yDistance + System.Environment.NewLine);
                        }
                        else
                        {
                                System.IO.File.AppendAllText(record.fileName1
                                , Time.time * 1000 + ",1,,,," +
                                currentCondition + "," +
                                conditionOrder[currentCondition] + "," +
                                trialCounter + "," + gunDistance + "," +
                                targetDistance + "," + hDistance + "," +
                                yDistance + System.Environment.NewLine);
                        }
                }
        }
        else
        {
                clicked = false;
        }

        break;
case 21: //Inter-trial screen
//During the inter-trial pause, the figure 8's are replaced by dots.
After a 1 second pause on this screen, the block time is checked. If the
block has run for more than 5 minutes it moves to the next block's
information screen. Otherwise it chooses a new target (screen, position,
orientation) and refills the figure 8's and returns to the trial screen.
        if (Time.time > timeFloat + 1)
        {
                if (Time.time > timeFloat2 + 300 ||
                (Input.GetKey(KeyCode.Z)&& Input.GetKey(KeyCode.M)))
                {
                        currentCondition++;
                        for (int i = 0; i < 60; i++)
                        {
                                xPosArray[i] =
                                Mathf.FloorToInt(Random.Range(0, 10));
                                yPosArray[i] =
                                Mathf.FloorToInt(Random.Range(0, 10));
                        }
                        if (currentCondition == conditionOrder.Length)
```

```
        {
                BlankStimuli();
                experimentStage = 99;
        }
        else
        {
                BlankStimuli();
                timeFloat = Time.time;
                experimentStage = 3;
        }
}
else
{
        trialCounter++;
        if (trialCounter % 60 == 60)
        {
                for (int i = 0; i < 60; i++)
                {
                        xPosArray[i] =
                        Mathf.FloorToInt(Random.Range(0, 10));
                        yPosArray[i] =
                        Mathf.FloorToInt(Random.Range(0, 10));
                }
        }
        FillStimuli();
        int tempInt;
        if (LRArray[trialCounter % 60] == 0)
        {
                tempInt = 0;
        }
        else
        {
                tempInt = 1;
        }

        xDistance = stimuliArray[screenArray[trialCounter %
        60] * 10 + xPosArray[trialCounter % 60],
        yPosArray[trialCounter %
        60]].gameObject.transform.position.x -
        headtrack.projectedX / 100;
        yDistance = stimuliArray[screenArray[trialCounter %
        60] * 10 + xPosArray[trialCounter % 60],
        yPosArray[trialCounter %
        60]].gameObject.transform.position.y -
        headtrack.projectedY / 100;
        zDistance = stimuliArray[screenArray[trialCounter %
        60] * 10 + xPosArray[trialCounter % 60],
        yPosArray[trialCounter %
        60]].gameObject.transform.position.z -
        headtrack.projectedZ / 100;
        if (xDistance > 0)
        {
                hDistance = Mathf.Sqrt(xDistance * xDistance
        + zDistance * zDistance);
        }
        else
        {
                hDistance = -Mathf.Sqrt(xDistance * xDistance
        + zDistance * zDistance);
        }
        targetDistance = Mathf.Sqrt(hDistance * hDistance +
        yDistance * yDistance);
```

```
                                gunDistance =
                                (stimuliArray[screenArray[trialCounter % 60] * 10 +
                                xPosArray[trialCounter % 60],
                                yPosArray[trialCounter %
                                60]].gameObject.transform.position - new
                                Vector3(guntrack.projectedX / 100,
                                guntrack.projectedY / 100, guntrack.projectedZ /
                                100)).magnitude;

                                stimuliArray[screenArray[trialCounter % 60] * 10 +
                                xPosArray[trialCounter % 60],
                                yPosArray[trialCounter % 60]].sprite = sprite3;
                                System.IO.File.AppendAllText(record.fileName1,
                                Time.time * 1000 + ",0," + tempInt + "," +
                                (screenArray[trialCounter % 60] * 10 +
                                xPosArray[trialCounter % 60]) + "," +
                                yPosArray[trialCounter % 60] + "," +
                                currentCondition + "," +
                                conditionOrder[currentCondition] + "," +
                                trialCounter + "," + gunDistance + "," +
                                targetDistance + "," + hDistance + "," + yDistance
                                + System.Environment.NewLine);

                                experimentStage = 20;
                                if (conditionOrder[currentCondition] == 1 ||
                                conditionOrder[currentCondition] == 2)
                                {
                                        SendScreenMessage();
                                }
                        }
                }
                break;
        default:

                break;
        }
}

//The OnGUI() function is called every frame like the Update() function. It
displays any text on the screens. These include participant instructions, cue
descriptions for each block and debug information hidden in the top left
corner.
void OnGUI()
{
        switch (experimentStage)
        {
        case 0:
                GUI.Label(new Rect(1034, 1, 1004, 548), "Please look at the
                yellow dot to calibrate the tracker:", style);
                break;
        case 1:
                GUI.Label(new Rect(1034, 1, 1004, 548), "Calibrating...", style);
                break;
        case 2:
                GUI.Label(new Rect(1034, 1, 1004, 548), "[" + conditionOrder[0] +
                "," + conditionOrder[1] + "," + conditionOrder[2] + "," +
                conditionOrder[3] + "," + conditionOrder[4] + "," +
                conditionOrder[5] + "," + conditionOrder[6] + "]", style);
                break;
        case 3:
                GUI.Label(new Rect(1034, 1, 1004, 548), "Condition " +
                (currentCondition + 1), style);
```

```
        break;
    case 10:
        GUI.Label(new Rect(1034, 1, 1004, 548), "Blank Condition",
        style);
        break;
    case 11: //static arrow
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition an stationary arrow will appear" +
        System.Environment.NewLine +
        "on your device pointing in the direction of the" +
        System.Environment.NewLine +
        "target from the center of the three screens."
        , style);
        break;
    case 15: //dynamic arrow
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition an arrow will appear on your device" +
        System.Environment.NewLine +
        "and point in the direction of the target. It will stand" +
        System.Environment.NewLine +
        "fully vertical when you reach the horizontal position" +
        System.Environment.NewLine +
        "of the target."
        , style);
        break;
    case 13: //luminance
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition your device's screen will" +
        System.Environment.NewLine +
        "brighten the closer your head is to pointing" +
        System.Environment.NewLine +
        "towards the target."
        , style);
        break;
    case 14: //hue
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition your device's screen will become" +
        System.Environment.NewLine +
        "more red the closer your head is to pointing towards" +
        System.Environment.NewLine +
        "the target."
        , style);
        break;
    case 16: //edge arrow
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition a small arrow will appear and" +
        System.Environment.NewLine +
        "point in the directon of the target. It will" +
        System.Environment.NewLine +
        "shrink as you approach the target."
        , style);
        break;
    case 12: //static grid
        GUI.Label(new Rect(1034, 1, 1004, 548),
        "In this condition a symbolic representation of the" +
        System.Environment.NewLine +
        "space will be displayed on your device. The target will" +
        System.Environment.NewLine +
        "be in the screen quadrant marked red."
        , style);
        break;
    case 17: //nothing
```

```
                GUI.Label(new Rect(1034, 1, 1004, 548),
                "In this condition you will get no assistance from" +
                System.Environment.NewLine +
                " your device." + System.Environment.NewLine
                , style);
                break;
        case 19: //nothing
                GUI.Label(new Rect(1034, 1, 1004, 548),
                "Blank" + System.Environment.NewLine
                , style);
                break;
        case 99:
                GUI.Label(new Rect(1034, 1, 1004, 548),
                "Experiment complete." + System.Environment.NewLine +
                "Thank you for participanting."
                , style);
                break;
        case 20: //during
                GUI.Label(new Rect(1034, 738, 1004, 20), "Head Distance: " +
                targetDistance);
                GUI.Label(new Rect(1034, 718, 1004, 20), "Gun Distance: " +
                gunDistance);
                break;
        default:

                break;
        }
        GUI.Label(new Rect(10, 738, 1004, 20), "Elapsed Time: " + Time.time);
        GUI.Label(new Rect(10, 10, 1004, 748), "connected: " + connected);
        GUI.Label(new Rect(10, 30, 1004, 748), "client IP: " + clientIP);
        GUI.Label(new Rect(10, 60, 1004, 748), "Reconnect Attempts: " +
        reconnectionAttempts);
        GUI.Label(new Rect(10, 80, 1004, 748), "Reconnection Timer: " +
        connectionCounter);
}

//These three functions, BlackStimuli(), DotStimuli(), and FillStimuli() are
called to hide all the targets, replace the targets with dots, and restore the
targets to figure 8's respectively. They are called between screen transitions.
void BlankStimuli()
{
        for (int i = 0; i < 30; i++)
        {
                for (int j = 0; j < 10; j++)
                {
                        stimuliArray[i, j].sprite = spriteBlank;
                }
        }
}

void DotStimuli()
{
        for (int i = 0; i < 30; i++)
        {
                for (int j = 0; j < 10; j++)
                {
                        stimuliArray[i, j].sprite = spriteDot;
                }
        }
}

void FillStimuli()
```

```
{
	for (int i = 0; i < 30; i++)
	{
		for (int j = 0; j < 10; j++)
		{
			stimuliArray[i, j].sprite = sprite8;
		}
	}
}

//This function is called between blocks to randomise the order of presentation
of events.
void randomizeArrays()
{
	for (int i = 59; i > 0; i--)
	{
		int j = Mathf.FloorToInt(Random.Range(0, i));
		int swapValue1 = screenArray[j];
		int swapValue2 = LRArray[j];
		screenArray[j] = screenArray[i];
		LRArray[j] = LRArray[i];
		screenArray[i] = swapValue1;
		LRArray[i] = swapValue2;
	}
	for (int i = 0; i <60; i++)
	{
		xPosArray[i] = Mathf.FloorToInt(Random.Range(0, 10));
		yPosArray[i] = Mathf.FloorToInt(Random.Range(0, 10));
	}
}

//This function is active only during the first screen and allows the
experiment to type in the IP address to connect to.
void InputIP()
{
	if (Input.anyKeyDown)
	{
		if (Input.GetKeyDown(KeyCode.Alpha0))
		{
			clientIP = clientIP + "0";
		}
		if (Input.GetKeyDown(KeyCode.Alpha1))
		{
			clientIP = clientIP + "1";
		}
		if (Input.GetKeyDown(KeyCode.Alpha2))
		{
			clientIP = clientIP + "2";
		}
		if (Input.GetKeyDown(KeyCode.Alpha3))
		{
			clientIP = clientIP + "3";
		}
		if (Input.GetKeyDown(KeyCode.Alpha4))
		{
			clientIP = clientIP + "4";
		}
		if (Input.GetKeyDown(KeyCode.Alpha5))
		{
			clientIP = clientIP + "5";
		}
		if (Input.GetKeyDown(KeyCode.Alpha6))
```

```
                {
                        clientIP = clientIP + "6";
                }
                if (Input.GetKeyDown(KeyCode.Alpha7))
                {
                        clientIP = clientIP + "7";
                }
                if (Input.GetKeyDown(KeyCode.Alpha8))
                {
                        clientIP = clientIP + "8";
                }
                if (Input.GetKeyDown(KeyCode.Alpha9))
                {
                        clientIP = clientIP + "9";
                }
                if (Input.GetKeyDown(KeyCode.Period))
                {
                        clientIP = clientIP + ".";
                }
                if (Input.GetKeyDown(KeyCode.Backspace))
                {
                        clientIP = "";
                }
        }
}

//updateNetwork() is called every frame by the Update() function and checks for
incoming messages such as connections and disconnections.
void updateNetwork()
{
        int recHostId;
        int recConnectionId;
        int recChannelId;
        byte[] recBuffer = new byte[16];
        int bufferSize = 16;
        int dataSize;
        byte error;
        NetworkEventType recNetworkEvent = NetworkTransport.Receive(out
        recHostId, out recConnectionId, out recChannelId, recBuffer, bufferSize,
        out dataSize, out error);
        switch (recNetworkEvent)
        {
        case NetworkEventType.Nothing:
                break;
        case NetworkEventType.ConnectEvent:
                connected = true;
                Debug.Log("incoming connection event received");
                System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
                ",7,Connected" + System.Environment.NewLine);
                SendBlankMessage();
                break;
        case NetworkEventType.DataEvent:
                Debug.Log("incoming message event received: ");
                break;
        case NetworkEventType.DisconnectEvent:
                connected = false;
                Debug.Log("remote client event disconnected");
                System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
                ",8,Disconnected" + System.Environment.NewLine);
                connectionCounter = 10;
        break;
        }
```

```
}

//Connect() or Disconnect() are called every time a connection or disconnection
attempt is required.
public void Connect()
{
	byte error;
	connectionId = NetworkTransport.Connect(socketId, clientIP, socketPort,
	0, out error);
	Debug.Log("Connected to server. ConnectionId: " + connectionId);
	System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
	",99,ConnectScriptRan" + System.Environment.NewLine);
}

public void Disconnect()
{
	byte error;
	NetworkTransport.Disconnect(socketId, connectionId, out error);
	System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
	",99,DisconnectScriptRan" + System.Environment.NewLine);
}

//SendBlankMessage() is called whenever the Glass's screen should be turned
off/turned fully transparent.
void SendBlankMessage()
{
	outgoingScreen = 0;
	outgoingValue = 0;
	outgoingValue2 = 0;
	SendSocketMessage();
}

//SendScreenMessage() is called whenever the Glass's screen should be turned on
and an image or text message displayed.
void SendScreenMessage()
{
	sendValue();
	SendSocketMessage();
}

//SendValue() chooses the type of message sent by SendScreenMessage() based on
the current condition's cue.
public void sendValue()
{
	switch (conditionOrder[currentCondition])
	{
	case 1: // Static Arrow
		outgoingScreen = 1;
		outgoingValue = Mathf.FloorToInt(1000f * Mathf.Atan2(-
		stimuliArray[screenArray[trialCounter % 60] * 10 +
		xPosArray[trialCounter % 60], yPosArray[trialCounter %
		60]].gameObject.transform.position.x,
		stimuliArray[screenArray[trialCounter % 60] * 10 +
		xPosArray[trialCounter % 60], yPosArray[trialCounter %
		60]].gameObject.transform.position.y));
		break;
	case 2: // Static Grid
		outgoingScreen = 7;
		outgoingValue = 0;
		if (xPosArray[trialCounter % 60] > 4.5)
		{
			outgoingValue += 1000;
```

```
        }
        if (yPosArray[trialCounter % 60] > 4.5)
        {
                outgoingValue += 6000;
        }
        if (screenArray[trialCounter % 60] == 1)
        {
                outgoingValue += 2000;
        }
        if (screenArray[trialCounter % 60] == 2)
        {
                outgoingValue += 4000;
        }
        break;
case 3: //Dynamic Luminance
        outgoingScreen = 31;
        if (targetDistance > 20)
        {
                outgoingValue = 1000;
        }
        else
        {
                outgoingValue = Mathf.FloorToInt(Mathf.Pow(targetDistance
        / 20f, 2) * 1000f);
        }
        break;
case 4: //Dynamic Hue
        outgoingScreen = 4;
        if (targetDistance > 20)
        {
                outgoingValue = 1000;
        }
        else
        {
                outgoingValue = Mathf.FloorToInt(targetDistance / 20f
        *1000f);
        }
        break;
case 5: //Dynamic Horizontal Compass
        outgoingScreen = 2;
        if (yPosArray[trialCounter % 60] < 4.5)
        {
                if (hDistance > 20)
                {
                        outgoingValue = -Mathf.FloorToInt(Mathf.PI / 2 *
                1000);
                }
                else
                {
                        if (hDistance < -20)
                        {
                                outgoingValue = -Mathf.FloorToInt(-Mathf.PI /
                        2 * 1000);
                        }
                        else
                        {
                                outgoingValue = -Mathf.FloorToInt(hDistance /
                        20 * Mathf.PI / 2 * 1000);
                        }
                }
        }
        else
```

```
        {
                if (hDistance > 20)
                {
                        outgoingValue = -Mathf.FloorToInt(Mathf.PI / 2 *
                        1000);
                }
                else
                {
                        if (hDistance < -20)
                        {
                                outgoingValue = -Mathf.FloorToInt(-Mathf.PI /
                        2 * 1000);
                        }
                        else
                        {
                                outgoingValue = -Mathf.FloorToInt((Mathf.PI -
                        hDistance / 20 * Mathf.PI / 2) * 1000);
                        }
                }
        }
        break;
case 6: //Edge Arrow
        outgoingScreen = 61;
        if (targetDistance > 10)
        {
                outgoingValue2 = 100000;
        }
        else
        {
                outgoingValue2 = Mathf.FloorToInt((targetDistance / 10f) *
        100000f);
        }
        outgoingValue = Mathf.FloorToInt(Mathf.Atan2(hDistance,
        yDistance)*1000);
        break;
case 7: //Control
        outgoingScreen = 0;
        break;
case 8: //Dynamic Luminance
        outgoingScreen = 31;
        if (targetDistance > 20)
        {
                outgoingValue = 1000;
        }
        else
        {
                outgoingValue = Mathf.FloorToInt(targetDistance / 20f *
        1000f);
        }
        break;
case 9: //Dynamic Luminance
        outgoingScreen = 31;
        if (targetDistance > 20)
        {
                outgoingValue = 1000;
        }
        else
        {
                outgoingValue = Mathf.FloorToInt(Mathf.Pow(targetDistance
        / 20f, 3) * 1000f);
        }
        break;
```

```
            }
        }

        //SendSocketMessage() pushes through the network message through to the
        bridging program when SendBlankMessage() and SendScreenMessage() have selected
        what information to send.
        public void SendSocketMessage()
        {
            byte error;
            byte[] buffer = new byte[16];

            buffer[0] = System.BitConverter.GetBytes(outgoingScreen)[0];
            buffer[1] = System.BitConverter.GetBytes(outgoingScreen)[1];
            buffer[2] = System.BitConverter.GetBytes(outgoingScreen)[2];
            buffer[3] = System.BitConverter.GetBytes(outgoingScreen)[3];
            buffer[4] = System.BitConverter.GetBytes(outgoingValue)[0];
            buffer[5] = System.BitConverter.GetBytes(outgoingValue)[1];
            buffer[6] = System.BitConverter.GetBytes(outgoingValue)[2];
            buffer[7] = System.BitConverter.GetBytes(outgoingValue)[3];
            buffer[8] = System.BitConverter.GetBytes(outgoingValue2)[0];
            buffer[9] = System.BitConverter.GetBytes(outgoingValue2)[1];
            buffer[10] = System.BitConverter.GetBytes(outgoingValue2)[2];
            buffer[11] = System.BitConverter.GetBytes(outgoingValue2)[3];

            int bufferSize = 16;

            NetworkTransport.Send(socketId, connectionId, myReliableChannelId,
            buffer, bufferSize, out error);
        }
}
```

//

Appendix 4.3: Google Glass Receiver Program and Bridging Program for Experiment 4

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using System.Net;
using UnityEngine.Networking;

using LostPolygon.AndroidBluetoothMultiplayer;

public class ReceiverControl : MonoBehaviour
{

        public SpriteRenderer mainSprite;

        public Sprite whiteArrow;
        public Sprite whiteCircle;
        public Sprite whiteSmallArrow;
        public Sprite whiteScreen;
        public Sprite blankSprite;

        public Sprite[] gridSprite;

        float rotationPercent;

        int screenState = 0;
        float incomingValue = 0;
        float incomingValue2 = 0;
```

```
public GUIStyle style;

int myReliableChannelId;
int socketId;
int socketPort = 8888;
int connectionId;
bool connected = false;
bool updated = false;

string[] ipString = { "", "", "", "", "" };

int disconnectTimer = -1;

float tempValue = 0;
float tempValue2 = 0;
float tempValue3 = 0;

bool isServer = false;
bool isClient = false;

bool showIntroImage = true;

public GameObject ActorPrefab;
public Transform floatHolder;

string uuid = "b909e9da-e83a-4fdb-b96c-1fa7334b756d";
private const string kLocalIp = "127.0.0.1"; // An IP for Network.Connect(),
must always be 127.0.0.1
private const int kPort = 28000; // Local server IP. Must be the same for
client and server

private bool _initResult;
private BluetoothMultiplayerMode _desiredMode = BluetoothMultiplayerMode.None;

//This function runs once as the program begins. It sets the Nexus 5/Glass's
screen to stay always on and determines which device it is. It sets a flag for
being the server on the Nexus 5 and a flag for being the client on any other
device. If the device is the server, that is the bridging device, the Nexus 5,
it sets up the other side of the socket to socket connection used appendix 4.2.
void Start()
{
	Screen.sleepTimeout = SleepTimeout.NeverSleep;

	if (AndroidBluetoothMultiplayer.GetCurrentDevice().Name == "Nexus 5")
	{
		isServer = true;
	}
	else
	{
		isClient = true;
	}
	if (isServer)
	{
		NetworkTransport.Init();
		ConnectionConfig config = new ConnectionConfig();
		myReliableChannelId = config.AddChannel(QosType.StateUpdate);
		int maxConnections = 10;
		HostTopology topology = new HostTopology(config, maxConnections);
		socketId = NetworkTransport.AddHost(topology, socketPort);
		Debug.Log("Socket Open. SocketId is: " + socketId);
	}
```

```
}

void Update()
{
	//This function runs every frame and does a different thing depending on
	whether the device is the bridge or the Glass. If it is the bridge, the
	it checks for incoming Wi-Fi messages from the desktop program. It will
	automatically try to reconnect the Wi-Fi if it disconnects. It also
	checks to make sure the synchronising game object exists. If it does,
	the function creates one. If more than one exists, it destroys them
	until one remains.

	if (isServer)
	{
		int recHostId;
		int recConnectionId;
		int recChannelId;
		byte[] recBuffer = new byte[16];
		int bufferSize = 16;
		int dataSize;
		byte error;
		NetworkEventType recNetworkEvent = NetworkTransport.Receive(out
		recHostId, out recConnectionId, out recChannelId, recBuffer,
		bufferSize, out dataSize, out error);
		switch (recNetworkEvent)
		{
		case NetworkEventType.Nothing:
			break;
		case NetworkEventType.ConnectEvent:
			tempValue2++;
			connected = true;
			break;
		case NetworkEventType.DataEvent:
			updated = true;
			screenState = System.BitConverter.ToInt32(recBuffer, 0);
			incomingValue = (System.BitConverter.ToInt32(recBuffer,
			4)) / 1000f;
			incomingValue2 = (System.BitConverter.ToInt32(recBuffer,
			8)) / 1000f;
			break;
		case NetworkEventType.DisconnectEvent:
			tempValue3++;
			connected = false;
			break;
		}
		if (connected)
		{
			disconnectTimer = 0;
			if (updated)
			{
				updated = false;
				if (floatHolder != null)
				{
					floatHolder.position = new
					Vector3(screenState, incomingValue,
					incomingValue2);
				}
			}
		}
		else
		{
			if (disconnectTimer != -1)
```

```
                {
                        disconnectTimer++;
                }
                if (disconnectTimer == 15)
                {
                        ResetPort();
                }
        }
        if (floatHolder != null)
        {
                NetworkTransform[] objects =
                FindObjectsOfType(typeof(NetworkTransform)) as
                NetworkTransform[];
                if (objects.Length > 1)
                {
                        foreach (NetworkTransform obj in objects)
                        {
                                Destroy(obj.gameObject);
                        }
                }
        }
        else
        {
                if (Network.isServer)
                {
                        GameObject newObject =
                        Network.Instantiate(ActorPrefab, Vector3.zero,
                        Quaternion.identity, 0) as GameObject;
                        floatHolder = newObject.GetComponent<Transform>();
                }
        }
}

//This function runs every frame and does a different thing depending on
whether the device is the bridge or the Glass. If it is the Glass, it
takes the values from the synchronising game object and stores them, or
searches for the object if it can't find one. It then runs the
DrawSprites script to display the cues to the screen.
if (isClient)
{
        if (floatHolder != null)
        {
                screenState =
                Mathf.FloorToInt(floatHolder.position.x+0.5f);
                incomingValue = floatHolder.position.y;
                incomingValue2 = floatHolder.position.z;
        }
        else
        {
                screenState = 0;
                incomingValue = 0;
                incomingValue2 = 0;
                NetworkTransform[] objects =
                FindObjectsOfType(typeof(NetworkTransform)) as
                NetworkTransform[];
                if (objects != null)
                {
                        floatHolder = objects[0].gameObject.transform;
                }
        }
drawSprites();
}
```

```
}

//This function runs every frame when called by the Update() function on the
Glass only. Cases are re-used where applicable; for example, condition 1, the
static arrow, and condition 5, the horizontal+distance dynamic arrow, both use
case 1.
void drawSprites()
{
	switch (screenState)
	{
		case 0: // Blank
		if (showIntroImage)
		{
			mainSprite.sprite = whiteArrow;
			mainSprite.transform.position = new Vector3(0f, 0f, 0f);
			mainSprite.colour = Colour.white;
		}
		else
		{
			mainSprite.sprite = blankSprite;
			mainSprite.transform.position = new Vector3(0f, 0f, 0f);
			mainSprite.colour = Colour.white;
			mainSprite.transform.rotation = new Quaternion(0f, 0f, 0f,
			1f);
			mainSprite.transform.localScale = new Vector3(100f, 100f,
			100f);
		}
		break;
	case 1: //AngledWhiteArrow
		showIntroImage = false;
		mainSprite.sprite = whiteArrow;
		mainSprite.transform.rotation = Quaternion.Euler(0, 0,
		incomingValue * Mathf.Rad2Deg);
		break;
	case 2: //AngledWhiteArrowRedundant (UNUSED)
		showIntroImage = false;
		mainSprite.sprite = whiteArrow;
		mainSprite.transform.rotation = Quaternion.Euler(0, 0,
		incomingValue * Mathf.Rad2Deg);
		break;
	case 3://WhiteCircleLuminance (UNUSED)
		showIntroImage = false;
		mainSprite.sprite = whiteCircle;
		mainSprite.colour = new Vector4(1f, 1f, 1f, (1 - incomingValue));
		break;
	case 31://WhiteCircleLuminanceReducedBrightness
		showIntroImage = false;
		mainSprite.sprite = whiteCircle;
		mainSprite.colour = new Vector4(1f, 1f, 1f, 0.8f * (1 -
		incomingValue));
		break;
	case 4://HueCircledGreenToRed
		showIntroImage = false;
		mainSprite.sprite = whiteCircle;
		mainSprite.colour = Colour.HSVToRGB(0.3f - 0.3f * (1 -
		incomingValue), 1f, 1f);
		break;
	case 6: //MinimapArrowOn/Off (UNUSED)
		showIntroImage = false;
		if (incomingValue == 99)
		{
			mainSprite.sprite = blankSprite;
```

```
            }
            else
            {
                  if (incomingValue > Mathf.PI)
                  {
                        incomingValue = incomingValue - Mathf.PI -
                        Mathf.PI;
                  }
                  mainSprite.sprite = whiteSmallArrow;
                  mainSprite.transform.rotation = Quaternion.Euler(0, 0, -
                  incomingValue * Mathf.Rad2Deg);
                  if (incomingValue < -2.077894831)
                  {
                        mainSprite.transform.position = new Vector3(-
                        Mathf.Tan(incomingValue) * 175f, -175f, 0f);
                  }
                  else if (incomingValue < -1.063697822)
                  {
                        mainSprite.transform.position = new Vector3(-315f,
                        Mathf.Tan((Mathf.PI / 2) + incomingValue) * 315,
                        0f);
                  }
                  else if (incomingValue < 1.063697822)
                  {
                        mainSprite.transform.position = new
                        Vector3(Mathf.Tan(incomingValue) * 175f, 175f, 0f);
                  }
                  else if (incomingValue < 2.077894831)
                  {
                        mainSprite.transform.position = new Vector3(315f,
                        Mathf.Tan((Mathf.PI / 2) - incomingValue) * 315,
                        0f);
                  }
                  else
                  {
                        mainSprite.transform.position = new Vector3(-
                        Mathf.Tan(incomingValue) * 175f, -175f, 0f);
                  }
            }
            break;
      case 61: //MinimapArrowScaling
            showIntroImage = false;
            mainSprite.transform.localScale = new Vector3(incomingValue2,
            incomingValue2, incomingValue2);
            if (incomingValue > Mathf.PI)
            {
                  incomingValue = incomingValue - Mathf.PI - Mathf.PI;
            }
            mainSprite.sprite = whiteSmallArrow;
            mainSprite.transform.rotation = Quaternion.Euler(0, 0, -
            incomingValue * Mathf.Rad2Deg);
            if (incomingValue < -2.077894831)
            {
                  mainSprite.transform.position = new Vector3(-
                  Mathf.Tan(incomingValue) * 175f, -175f, 0f);
            }
            else if (incomingValue < -1.063697822)
            {
                  mainSprite.transform.position = new Vector3(-315f,
                  Mathf.Tan((Mathf.PI / 2) + incomingValue) * 315, 0f);
            }
            else if (incomingValue < 1.063697822)
```

```
                {
                        mainSprite.transform.position = new
                        Vector3(Mathf.Tan(incomingValue) * 175f, 175f, 0f);
                }
                else if (incomingValue < 2.077894831)
                {
                        mainSprite.transform.position = new Vector3(315f,
                        Mathf.Tan((Mathf.PI / 2) - incomingValue) * 315, 0f);
                }
                else
                {
                        mainSprite.transform.position = new Vector3(-
                        Mathf.Tan(incomingValue) * 175f, -175f, 0f);
                }

                break;
        case 7: //GridMap
                showIntroImage = false;
                mainSprite.sprite = gridSprite[(int)incomingValue];
                mainSprite.transform.position = new Vector3(0f, 0f, 0f);
                mainSprite.colour = Colour.white;
                mainSprite.transform.rotation = new Quaternion(0f, 0f, 0f, 1f);
                break;
        }
}

//This function runs each frame and runs separate function based on whether it
is the bridging device or the Google Glass.
private void OnGUI()
{
        //On the bridging device, the values being passed through (current
        screen, value1 and value2) are displayed on the phone screen. A
        disconnection message is displayed if the program disconnects after
        having been connected initially. It also starts the Bluetooth server if
        it is not currently running.
        if (isServer)
        {
                GUI.Label(new Rect(0, 0, 640, 360), "screenState: " +
                screenState, style);
                GUI.Label(new Rect(0, 50, 640, 360), "incomingValue1: " +
                incomingValue, style);
                GUI.Label(new Rect(0, 100, 640, 360), "incomingValue2: " +
                incomingValue2, style);
                if (disconnectTimer == -1)
                {
                        GUI.Label(new Rect(0, -100, 640, 360), "Waiting For
                        Network Connection", style);
                }
                if (disconnectTimer > 30)
                {
                        GUI.Label(new Rect(0, -100, 640, 360), "Connection Failed
                        [" + tempValue3 + "/" + tempValue2 + "]", style);
                }

                if (_initResult)
                {
                        BluetoothMultiplayerMode currentMode =
                        AndroidBluetoothMultiplayer.GetCurrentMode();
                        if (currentMode == BluetoothMultiplayerMode.None)
                        {
```

```
                        if
                        (AndroidBluetoothMultiplayer.GetIsBluetoothEnabled(
                        ))
                        {
                                AndroidBluetoothMultiplayer.StartServer(kPort
                        );
                        }
                        else
                        {
                                AndroidBluetoothMultiplayer.RequestEnableDisc
                                overability(120);
                                _desiredMode =
                                BluetoothMultiplayerMode.Server;
                        }
                }
        }
}

        //On the Google Glass, this function connects to the Bluetooth server if
        it is not currently connected. It brings up an enable Bluetooth popup if
        Bluetooth has not been activated on the Glass.
        if (isClient)
        {
                if (_initResult)
                {
                        BluetoothMultiplayerMode currentMode =
                        AndroidBluetoothMultiplayer.GetCurrentMode();
                        if (currentMode == BluetoothMultiplayerMode.None)
                        {
                                if
                                (AndroidBluetoothMultiplayer.GetIsBluetoothEnabled(
                                ))
                                {
                                        BluetoothDevice[] bluetoothDeviceList =
                                        AndroidBluetoothMultiplayer.GetBondedDevices(
                                        );
                                        AndroidBluetoothMultiplayer.Connect(bluetooth
                                        DeviceList[0].Address, kPort);
                                }
                                else
                                {
                                        AndroidBluetoothMultiplayer.RequestEnableDisc
                                        overability(120);
                                        _desiredMode =
                                        BluetoothMultiplayerMode.Client;
                                }
                        }
                }
        }
}

//The ResetPort() function was tested as a method to quickly reconnect the
experiment when the connection had failed too many times and would block any
further attempts. It did not solve the problem and only triggers when a
complete chain collapse has already occurred. In this situationm the Glass
needs to be reset and the experiment restarted from the begin of the current
block.
void ResetPort()
{
        NetworkTransport.RemoveHost(socketId);
        ConnectionConfig config = new ConnectionConfig();
        myReliableChannelId = config.AddChannel(QosType.StateUpdate);
```

```
        int maxConnections = 10;
        HostTopology topology = new HostTopology(config, maxConnections);
        socketId = NetworkTransport.AddHost(topology, socketPort);
}

//Called to retrieve IP address. These are now hardcoded and this function
doesn't need to be called.
void GetLocalIPAddress()
{
        var host = Dns.GetHostEntry(Dns.GetHostName());
        int tempCount = 0;
        foreach (var ip in host.AddressList)
        {
                tempCount++;
                if (tempCount < 6)
                {
                        ipString[tempCount - 1] = ip.ToString();
                }
        }
}

//The remaining functions are provided by the Android Bluetooth Multiplayer
Library. Together these functions synchronise a game object between the server
and client through a Bluetooth connection. This object passes messages and
values from Nexus 5 to Google Glass.
private void Awake()
{
        // Setting the UUID. Must be unique for every application
        _initResult = AndroidBluetoothMultiplayer.Initialize(uuid);

        // Enabling verbose logging. See log cat!
        AndroidBluetoothMultiplayer.SetVerboseLog(true);

        // Registering the event delegates
        AndroidBluetoothMultiplayer.ListeningStarted +=
        OnBluetoothListeningStarted;
        AndroidBluetoothMultiplayer.ListeningStopped +=
        OnBluetoothListeningStopped;
        AndroidBluetoothMultiplayer.AdapterEnabled += OnBluetoothAdapterEnabled;
        AndroidBluetoothMultiplayer.AdapterEnableFailed +=
        OnBluetoothAdapterEnableFailed;
        AndroidBluetoothMultiplayer.AdapterDisabled +=
        OnBluetoothAdapterDisabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnabled +=
        OnBluetoothDiscoverabilityEnabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnableFailed +=
        OnBluetoothDiscoverabilityEnableFailed;
        AndroidBluetoothMultiplayer.ConnectedToServer +=
        OnBluetoothConnectedToServer;
        AndroidBluetoothMultiplayer.ConnectionToServerFailed +=
        OnBluetoothConnectionToServerFailed;
        AndroidBluetoothMultiplayer.DisconnectedFromServer +=
        OnBluetoothDisconnectedFromServer;
        AndroidBluetoothMultiplayer.ClientConnected +=
        OnBluetoothClientConnected;
        AndroidBluetoothMultiplayer.ClientDisconnected +=
        OnBluetoothClientDisconnected;
        AndroidBluetoothMultiplayer.DevicePicked += OnBluetoothDevicePicked;
}

// Don't forget to unregister the event delegates!
void OnDestroy()
```

```
{
        AndroidBluetoothMultiplayer.ListeningStarted -=
        OnBluetoothListeningStarted;
        AndroidBluetoothMultiplayer.ListeningStopped -=
        OnBluetoothListeningStopped;
        AndroidBluetoothMultiplayer.AdapterEnabled -= OnBluetoothAdapterEnabled;
        AndroidBluetoothMultiplayer.AdapterEnableFailed -=
        OnBluetoothAdapterEnableFailed;
        AndroidBluetoothMultiplayer.AdapterDisabled -=
        OnBluetoothAdapterDisabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnabled -=
        OnBluetoothDiscoverabilityEnabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnableFailed -=
        OnBluetoothDiscoverabilityEnableFailed;
        AndroidBluetoothMultiplayer.ConnectedToServer -=
        OnBluetoothConnectedToServer;
        AndroidBluetoothMultiplayer.ConnectionToServerFailed -=
        OnBluetoothConnectionToServerFailed;
        AndroidBluetoothMultiplayer.DisconnectedFromServer -=
        OnBluetoothDisconnectedFromServer;
        AndroidBluetoothMultiplayer.ClientConnected -=
        OnBluetoothClientConnected;
        AndroidBluetoothMultiplayer.ClientDisconnected -=
        OnBluetoothClientDisconnected;
        AndroidBluetoothMultiplayer.DevicePicked -= OnBluetoothDevicePicked;
}

void OnBackToMenu()
{
        // Gracefully closing all Bluetooth connectivity and loading the menu
        try
        {
        AndroidBluetoothMultiplayer.StopDiscovery();
        AndroidBluetoothMultiplayer.Stop();
        }
        catch
        {

        }
}

private void OnBluetoothListeningStarted()
{
        Network.InitializeServer(4, kPort, false);
}

private void OnBluetoothListeningStopped()
{
        AndroidBluetoothMultiplayer.Stop();
}

private void OnBluetoothDevicePicked(BluetoothDevice device)
{
        AndroidBluetoothMultiplayer.Connect(device.Address, kPort);
}

private void OnBluetoothDisconnectedFromServer(BluetoothDevice device)
{
        Network.Disconnect();
}

private void OnBluetoothConnectedToServer(BluetoothDevice device)
```

```
        {
                Network.Connect(kLocalIp, kPort);
        }

        private void OnBluetoothAdapterEnabled()
        {
                switch (_desiredMode)
                {
                case BluetoothMultiplayerMode.Server:
                        Network.Disconnect();
                        AndroidBluetoothMultiplayer.StartServer(kPort);
                        break;
                case BluetoothMultiplayerMode.Client:
                        Network.Disconnect();
                        BluetoothDevice[] bluetoothDeviceList =
                        AndroidBluetoothMultiplayer.GetBondedDevices();
                        AndroidBluetoothMultiplayer.Connect(bluetoothDeviceList[0].Addres
                        s, kPort);
                        break;
                }
                _desiredMode = BluetoothMultiplayerMode.None;
        }

        private void OnFailedToConnect(NetworkConnectionError error)
        {
                AndroidBluetoothMultiplayer.Stop();
        }

        private void OnDisconnectedFromServer()
        {
                AndroidBluetoothMultiplayer.Stop();
        }
}
```

//

Appendix 4.4: Example Output and Data Processing for Experiment 4

Example data output for an Experiment 4 event log. Measurements are taken every time search begins (Event Type 0), a shot made away from the correct target (Event Type 1), and when a shot is made close to the current target, ending the trial (Event Type 2). X and Y coordinates of the target are measured from the top left hand corner position as (0,0) to the bottom right hand corner as (29,9).

Start Time:	2016	7	23	14	7	12	578			
Time_Millis	Type	Stimuli3OrE	StimuliXPos	StimuliYPos	BlockNumber	ConditionType	Trial	RotationToTarget	HorizontalRotationToTarget	VerticalRotationToTarget
61005.4	99	ConnectScriptRan								
61022.29	7	Connected								
235757	0	1	7	6	0	5	0	35.22518	35.12754	2.664988
397179.9	1				0	5	0	7.658808	3.495472	6.822484
410772.7	2				0	5	0	5.078483	3.311448	3.851174
411788.7	0	0	26	2	0	5	1	92.9855	93.79096	9.447144
432962.5	2				0	5	1	20.82723	13.70237	15.75106
433962.8	0	1	2	5	0	5	2	85.47562	85.34558	5.287613
438642.9	2				0	5	2	7.247871	6.785038	2.551903
439643	0	0	2	1	0	5	3	29.22376	22.87356	18.49442
442041.3	2				0	5	3	15.23943	0.118694	15.239
443058.2	0	0	17	5	0	5	4	53.6104	53.54525	2.78263
445407	2				0	5	4	2.813064	0.22985	2.803724
446423.5	0	1	4	6	0	5	5	52.30927	52.33127	1.238911
449187.9	2				0	5	5	7.458589	6.951047	2.727998
450204	0	0	9	1	0	5	6	17.65787	3.214002	17.36733
453985.7	2				0	5	6	17.4557	9.450639	14.72873
455002	0	1	29	0	0	5	7	106.7624	107.4328	19.63453
462382	2				0	5	7	14.16794	7.881124	11.89575
463397.9	0	1	19	3	0	5	8	29.29914	28.60481	6.758799
468612.7	2				0	5	8	11.04641	3.897973	10.33904
469628.8	0	0	12	0	0	5	9	32.49241	25.10205	20.99286
...	...	...	...	...	...	...	...	...	...	...

To compress the information to a single row per trial the following excel macro was used:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
Sub CompressData()
Dim SearchMisses As Integer
Dim StartHold As Double
Dim StartHold2 As Double
Dim CurrentRow As Long
SearchMisses = 0
StartHold = 0
StartHold2 = 0
StartHold3 = 0
StartHold4 = 0
CurrentRow = 2
PrintRow = 2

Do While (CurrentRow < 2000)
  If (Cells(CurrentRow, 2).Value = 0) Then
    SearchMisses = 0
    StartHold = Cells(CurrentRow, 1).Value
    StartHold2 = Cells(CurrentRow, 9).Value
    StartHold3 = Cells(CurrentRow, 10).Value
    StartHold4 = Cells(CurrentRow, 11).Value
    'Cells(PrintRow, 21).Value = Cells(CurrentRow, 1).Value
    'Cells(PrintRow, 22).Value = Cells(CurrentRow, 4).Value
    'Cells(PrintRow, 23).Value = Cells(CurrentRow, 5).Value
  Else
    If (Cells(CurrentRow, 2).Value = 2 Or Cells(CurrentRow, 2).Value = 9) Then
      SearchMisses = 0
```

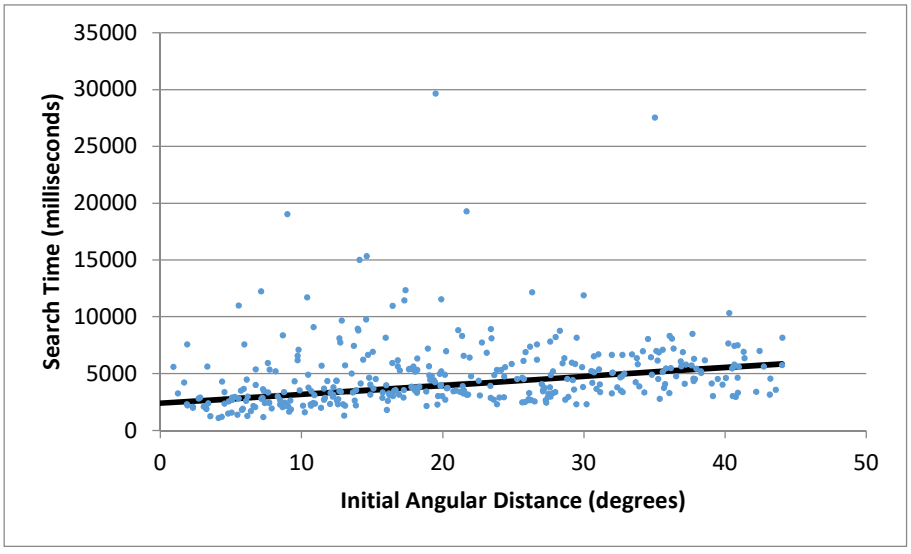
```
        If (Cells(CurrentRow, 7).Value > 0) Then
          Cells(PrintRow, 14).Value = Cells(CurrentRow, 6).Value
          Cells(PrintRow, 15).Value = Cells(CurrentRow, 7).Value
          Cells(PrintRow, 16).Value = Cells(CurrentRow, 8).Value
          Cells(PrintRow, 17).Value = Cells(CurrentRow, 1).Value - StartHold
          Cells(PrintRow, 18).Value = StartHold2
          Cells(PrintRow, 19).Value = StartHold3
          Cells(PrintRow, 20).Value = StartHold4

          Cells(PrintRow, 22).Value = StartHold

          'Cells(PrintRow, 20).Value = StartHold2 - Cells(CurrentRow, 10).Value
        Else
          Cells(PrintRow, 14).Value = "Break"
        End If
        PrintRow = PrintRow + 1
      Else
        SearchMisses = SearchMisses + 1
      End If
    End If
    CurrentRow = CurrentRow + 1
Loop

End Sub
```

//

This macro simplifies the data, taking the start and stop points from the initial log and calculating the duration of each trial as well as maintaining the initial angular distance (in degrees) to needed be travelled by the user in order to reach the target. The first 10 trials of each block were removed to give participant performance a chance to stabilise after being first exposed to a new condition. An example of the compressed data is shown below.

The response time data were heavily right skewed with several outliers. This is because participants were able to miss a target on their first, or subsequent, passes, greatly increasing the amount of time to find random targets throughout the block. To account for this, responses were averaged using medians rather than means. A peripheral vision arc of ±55^{o} was used to split targets inside and outside of peripheral vision and a median response time for all targets, targets initially inside peripheral vision and targets outside peripheral vision was calculated. This is the primary data used in Chapter 4. A quantile regression was also fit to the data (Koenker & Hallock, 2001) to measure the effect of initial distance to the target on response times, excluding the outliers created by participants not seeing a target on their first pass. This model was fit using the Excel solver function to the aggregation of responses from all participants. An example regression fit is shown below. The parameters for the fit are displayed in table 4.1 of chapter 4.

Trial	TimeToCompletion	Direct Rotation To Target	Horizontal Rotation To Target	Vertical Rotation To Target	Head Travel Distance
10	3164	44.7	44.6	3.0	24.3
11	4539	29.1	28.4	6.6	30.9
12	6560	46.5	46.4	3.0	53.0
13	8315	82.9	82.2	24.4	45.5
14	2033	14.7	14.7	1.0	11.3
15	2965	13.0	5.6	11.7	26.3
16	3360	16.7	14.3	8.8	12.0
17	1193	7.9	4.2	6.7	1.9
18	3880	42.5	37.2	22.3	22.8
19	1652	18.4	15.8	9.6	3.5
20	1176	14.2	7.6	12.0	3.1
21	1587	20.2	8.3	18.5	4.4
22	6988	85.5	85.5	2.5	36.5
23	1905	13.1	13.0	1.2	5.0
24	3163	30.3	18.6	24.4	6.9
25	6826	44.9	44.9	1.9	38.4
26	8923	46.2	45.7	7.8	47.8
27	19268	46.5	44.6	14.7	112.1
28	3743	42.0	41.9	2.1	24.9
29	5434	34.2	32.3	11.7	21.2
30	8156	63.7	62.4	17.1	65.8
31	1375	11.3	10.4	4.5	4.5
32	3501	44.1	39.9	20.5	32.2
...	...	...	...	...	...



Appendix 4.5: Paper Materials for Experiment 4

Consent form for Experiment 4.

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Assisting Visual Search with Head Mounted Displays

Consent Form for Participants

I have been given a full explanation of this project, I understand what is required of me if I agree to take part in the research and I have had the opportunity to ask questions.

I understand that participation is voluntary and I may withdraw at any time without penalty. I understand that withdrawal of participation will also include the withdrawal of any information I have provided should this remain practically achievable.

I understand that I can request to stop the test at any time if I feel any discomfort for any reason.

I understand that any information or opinions I provide will be kept confidential to the researcher and their senior supervisor and that any published or reported results will not identify me or the other participants. I understand that the data may be used in published articles and a thesis which is a public document and will be available through the UC Library.

I understand that all identifying data collected for the study will be kept in locked and secure facilities and will be destroyed after ten years.

I understand that I can contact the researchers Matt Ward (matt.ward@pg.canterbury.ac.nz), Hao Chen (hao.chen@pg.canterbury.ac.nz) or supervisor Dr Deak Helton (deak.helton@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Human Ethics Committee, Private Bag 4800, Christchurch (humanethics@canterbury.ac.nz)

By signing below, I agree to participate in this research project.

Name:

Date:

Signature:

Psychology
Telephone: +64 364 2987, ext. 7098
Email: matt.ward@pg.canterbury.ac.nz
29/12/14

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Information sheet for participants.

You are being asked to participate in the study 'Assisting Visual Search with Head Mounted Displays'. This study will require you to find the image that is different from the others on a set of three large screens and then point to that target with a 3D mouse. You will be assisted in finding the odd targets by a wearable computer, a Google Glass, which will use pointers, symbols, colours, and brightness changes to direct you. A palm sensor will measure your heart rate and skin conductance (sweatiness) while you do so as a measure of the effort it takes to follow the cues.

There are no expected risks to participants' physical or mental health, although if the task is uncomfortable for any reason, you may request for the researcher to stop the test without penalty.

Participation is voluntary and you have the right to withdraw at any stage without penalty. If you withdraw during the test, all information relating to you will be destroyed. To ensure anonymity and confidentiality, all digital copies of your name will be replaced with a randomly assigned participant number and no other identifying data will be stored. It will therefore be impossible to remove your data from the pool after it has been collected. Paper copies of any testing materials will also use your randomly assigned participant number. A consent form bearing your name will be stored for a period of ten years in a locked filing cabinet and will only be made available to the primary researcher Matt War, and their senior supervisor, Dr Deak Helton, as well as the University of Canterbury Human Ethics Committee if such disclosure becomes necessary.

The results of the project might be used in a doctoral thesis and may also be published. A thesis is a public document and will be available through the UC Library.

This project is being carried out by Matt Ward (matt.ward@pg.canterbury.ac.nz), Hao Chen (hao.chen@pg.canterbury.ac.nz) under the supervision of Dr Deak Helton, who can be contacted at deak.helton@canterbury.ac.nz if you have any concerns about participation in this project.

This project has been reviewed approved by the University of Canterbury Psychology department and the UCHEC Low Risk Approval process. Participants should address any complaints to The Chair, Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz). If you agree to participate in the study, you are asked to complete the consent form before continuing with the test.

Post-study questionnaire for Experiment 4, page 1:

ID Number Age Gender (M/F/O) _____ ____ _____ How did you use the cue? (Tick 1 of 3)	Didn't look at all/Ignored it.	Looked at the device once at the beginning, or just a couple of times.	Constantly looked between device and th screens/ through device at screen.
The static (non-moving) arrow pointing in the direction of the target from the center.			
The symbolic representation of the three screens (3 white squares) which highlighted the region holding the target in red.			
The white dot that became brighter as you got closer to the target.			
The green dot transitioned to red as you got closer to the target.			
The arrow that became vertical as you reached the horizontal position of the target.			
The arrow which pointed on the edge of the screen towards the target and shrunk as you got close.			
No cue.			

If you were going to have to do this task for prizes, which one cue would you want to use?

_____

EXP04CQ

Post-study questionnaire for Experiment 4, page 2:

How did you feel about the various cue types? *1: Actively distracting/counterproductive. Would prefer no cue.* *2: Not useful. Same or similar to having no assistance.* *3: Better than nothing but sometimes distracting or unclear.* *4: A good, clear cue. Only minor issues, if any.* *5: An excellent cue type. Would choose to use it above any others.*	How did you feel about the various cue types? (1-5)	Please rank the cue types 1-7 with 7 being the best.
The static (non-moving) arrow pointing in the direction of the target from the center.		
The symbolic representation of the three screens (3 white squares) which highlighted the region holding the target in red.		
The white dot that became brighter as you got closer to the target.		
The green dot transitioned to red as you got closer to the target.		
The arrow that became vertical as you reached the horizontal position of the target.		
The arrow which pointed on the edge of the screen towards the target and shrunk as you got close.		
No cue.		

Any notes on the cue types or experiment as a whole:

EXP04CQ

Appendix 5.0: Ethics, Programming, Data Analysis and Paper Materials for Experiment 5

The University of Canterbury Human Ethic Approval for this experiment is presented in Appendix 5.1.

Experiment 5's core and HMD receiver programs were written in the Unity engine (Unity3d.com, 2015) using the C# language. The Motive software (OptiTrack, 2017) provided the tracking data were transferred locally to the main experiment program.

Appendix 5.2 contains the core experiment program.

Appendix 5.3 contains the HMD display program and desktop to Glass bridging program for a smartphone.

Appendix 5.4 shows example data output files from the experiment and the process used to extract useful data. Macros used in the process are written in the Visual Basic language.

Appendix 5.5 contains the paper materials used in this experiment. These include an information sheet, a consent form and pre and post-experiment questionnaires.

Appendix 5.1: Human Ethics Approval for Experiment 5

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE
Secretary, Rebecca Robinson
Telephone: +64 03 364 2987, Extn 45588
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2016/58/LR-PS

3 October 2016

Amit Barde
HIT Lab NZ
UNIVERSITY OF CANTERBURY

Dear Amit

Thank you for submitting your low risk application to the Human Ethics Committee for the research proposal titled "A Real World Search Task Using Spatialised Auditory Cues and Visual Cues Delivered over a Wearable Interface".

I am pleased to advise that this application has been reviewed and approved.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 27th September 2016.

With best wishes for your project.

Yours sincerely

pp. R. Robinson

Kelly Dombroski
Deputy Chair, Human Ethics Committee

Appendix 5.2: Main Controller Program for Experiment 5.

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

public class NetworkControllerScript : MonoBehaviour
{

        int messageType;
        // 0: start trial
        // 1: end trial
        // 2: update position
        // 3: dead trial

        int myReliableChannelId;
        int socketId;
        int socketPort = 8888;
        int connectionId;
        int connectionCounter = 0;
        int connectionCounter2 = 0;
        int reconnectionAttempts = 0;
        bool connected = false;
        bool hasConnected = false;
        string clientIP = "192.168.43.1";
        float[,] posHold = new float[2, 3];
        bool posUpdate = false;
        bool netUpdate = false;
        bool primed = false;
        int inRangeSwitch = 0;
        bool showCue = false;
        int internaltrialNumber = 0;
        bool soundCued = false;
        float soundTimeFloat = 0;

        int[] buttonPressedArray = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        //These arrays hold the sequence of events shared by all participants.
        int[] tableArray =    { 1, 6, 5, 4, 4, 3, 6,  6,  1,  6,  6,  5, 2, 1, 5, 6, 4,
        2, 6, 5, 3, 6, 1, 2, 4, 2, 2, 6, 1, 5, 3, 2, 1, 6, 6, 3, 2, 1, 1, 5, 1, 2, 3,
        6, 6, 6, 2, 2, 3, 2, 6, 6, 6, 5, 4, 2, 2, 1, 2, 2, 1, 2, 3, 1, 1, 4, 2, 1, 6,
        5, 3, 6, 2, 4, 6, 1, 1, 4, 3, 3, 1, 6, 3, 2, 6, 6, 6, 1, 6, 3 };
        int[] positionArray = { 6, 4, 4, 1, 9, 6, 12, 15, 15, 11, 14, 3, 2, 3, 1, 8, 7,
        10, 15, 3, 5, 11, 10, 11, 4, 15, 10, 5, 13, 4, 3, 6, 15, 5, 2, 1, 9, 6, 14, 1,
        9, 9, 2, 6, 15, 15, 11, 6, 8, 15, 1, 12, 2, 3, 9, 14, 11, 7, 10, 3, 15, 6, 6,
        15, 14, 2, 12, 14, 7, 2, 4, 5, 10, 6, 1, 2, 3, 8, 8, 1, 14, 1, 5, 4, 15, 8, 11,
        2, 13, 7 };
        int[] rowArray =      { 2, 1, 2, 1, 3, 2, 3,  3,  3,  3,  3,  2, 1, 1, 1, 2, 3,
        4, 3, 2, 2, 3, 2, 4, 2, 5, 4, 1, 3, 2, 1, 2, 3, 1, 1, 1, 3, 2, 3, 1, 2, 3, 1,
        2, 3, 3, 4, 2, 3, 5, 1, 3, 1, 2, 3, 5, 4, 2, 4, 1, 3, 2, 2, 3, 3, 1, 4, 3, 2,
        1, 2, 1, 4, 2, 1, 1, 1, 3, 3, 1, 3, 1, 2, 2, 3, 2, 3, 1, 3, 3 };
        int[] columnArray =   { 1, 4, 2, 1, 3, 3, 2,  5,  5,  1,  4,  1, 2, 3, 1, 3, 1,
        1, 5, 1, 2, 1, 5, 2, 1, 3, 1, 5, 3, 2, 3, 3, 5, 5, 2, 1, 3, 1, 4, 1, 4, 3, 2,
        1, 5, 5, 2, 3, 2, 3, 1, 2, 2, 1, 3, 2, 2, 2, 1, 3, 5, 3, 3, 5, 4, 2, 3, 4, 2,
        2, 1, 5, 1, 3, 1, 2, 3, 2, 2, 1, 4, 1, 2, 1, 5, 3, 1, 2, 3, 1 };
        int[] mapXArray = { 214, 331, 209, 364, 404, 299, 283, 355, 310, 259, 331, 184,
        186, 262, 184, 307, 364, 164, 355, 184, 279, 259, 310, 186, 364, 208, 164, 355,
        262, 209, 299, 208, 310, 355, 283, 259, 208, 214, 286, 184, 286, 208, 279, 259,
        355, 355, 186, 208, 279, 208, 259, 283, 283, 184, 404, 186, 186, 238, 164, 208,
        310, 208, 299, 310, 286, 384, 208, 286, 283, 209, 259, 355, 164, 404, 259, 238,
        262, 384, 279, 259, 286, 259, 279, 164, 355, 307, 259, 238, 307, 259 };
        int[] mapYArray = { 259, 52, 27, 206, 141, 145, 8, 8, 237, 8, 8, 27, 237, 281,
        52, 30, 141, 165, 8, 27, 145, 8, 259, 165, 174, 141, 165, 52, 237, 27, 177,
```

```
213, 237, 52, 52, 177, 189, 259, 237, 52, 259, 189, 177, 30, 8, 8, 165, 213,
112, 141, 52, 8, 52, 27, 141, 141, 165, 259, 165, 237, 237, 213, 145, 237, 237,
206, 165, 237, 30, 52, 145, 52, 165, 174, 52, 281, 281, 141, 112, 177, 237, 52,
145, 213, 8, 30, 8, 281, 8, 112 };

public Texture texCentralDot;
public Texture texLeftArrow;
public Texture texRightArrow;
public Texture texWhiteSpace;
public Texture texMap;
public Texture texHotSpot;
public GUIStyle glassStyle;
public GUIStyle blackWindowStyle;
public Rect windowRect = new Rect(20, 20, 640, 360);

int[] cueArray = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
string[] cueArrayVisual = { "None", "Map", "Arrow", "None", "Map", "Arrow",
"None", "Map", "Arrow" };
string[] cueArraySound = { "None", "None", "None", "Static", "Static",
"Static", "Dynamic", "Dynamic", "Dynamic" };
int trialNumber = 0;
float timeFloat = 0f;

float rotationToTarget = 0f;
float distanceToTarget = 0f;
float rotationOffset = 0f;

public Transform headTrack;
public Transform staticSoundSource;

public RecorderScript record;

int outMessageType;
int outMessageValue1; //angle difference
int outMessageValue2; //target distance
int outMessageValue3; //trial number
int outMessageValue4; //cue type
int outMessageValue5; //show cue

bool playingStaticSound = false;

public TBE_3DCore.TBE_Source soundSourceDynamic;
public TBE_3DCore.TBE_Source soundSourceStatic;
public Transform AudioCueTransform;
public AudioCue AudioCueScript;

float[,] targetColumnArrayX = {{ -0.73f, -0.37f, -0.05f, 0.29f, 0.67f }, { -
1.48f, -1.19f, -0.91f, 99f, 99f }, { 0f, 0.22f, 0.43f, 99f, 99f }, { 1.32f,
1.53f, 1.74f, 99f, 99f }, { -1.41f, -1.15f, 99f, 99f, 99f }, { -0.26f, 0.04f,
0.34f, 0.64f, 0.94f }};
float[,] targetRowArrayZ = { { -1.84f, -1.52f, -1.2f, 99f, 99f }, { -1.04f, -
0.74f, -0.44f, -0.14f, 0.16f }, { 0.04f, 0.40f, 0.76f, 99f, 99f }, { -0.41f, -
0.03f, 0.35f, 99f, 99f }, { 1.49f, 1.78f, 99f, 99f, 99f }, { 1.81f, 2.10f,
2.39f, 99f, 99f } };

//This function runs once on start up. It calls the two following functions.
void Start()
{
       initializeNetwork();
       ShuffleCueArray();
}
```

```
//This function is called by the Start() function. It sets up the socket to
socket connection server to allow the bridging device to connect to this
program.
void initializeNetwork()
{
	NetworkTransport.Init();
	ConnectionConfig config = new ConnectionConfig();
	myReliableChannelId = config.AddChannel(QosType.StateUpdate);
	int maxConnections = 10;
	HostTopology topology = new HostTopology(config, maxConnections);
	socketId = NetworkTransport.AddHost(topology, socketPort);
	Debug.Log("Socket Open. SocketId is: " + socketId);
}

//This function is called each time a new block of 9 trials needs to be
generated. It shuffles the order of cue types.
void ShuffleCueArray()
{
	for (int i = 0; i < 9; i++)
	{
		cueArray[i] = i + 1;
	}
	for (int i = 0; i < 3; i++)
	{
		cueArraySound[i] = "None";
		cueArraySound[i + 3] = "Static";
		cueArraySound[i + 6] = "Dynamic";
		cueArrayVisual[3 * i] = "None";
		cueArrayVisual[3 * i + 1] = "Map";
		cueArrayVisual[3 * i + 2] = "Arrow";
	}
	int tempRandom;
	int tempHold;
	string tempHold2;
	string tempHold3;
	for (int i = 8; i > 0; i--)
	{
		tempRandom = Mathf.FloorToInt(Random.Range(0f, i + 0.99999f));
		tempHold = cueArray[i];
		tempHold2 = cueArrayVisual[i];
		tempHold3 = cueArraySound[i];
		cueArray[i] = cueArray[tempRandom];
		cueArrayVisual[i] = cueArrayVisual[tempRandom];
		cueArraySound[i] = cueArraySound[tempRandom];
		cueArray[tempRandom] = tempHold;
		cueArrayVisual[tempRandom] = tempHold2;
		cueArraySound[tempRandom] = tempHold3;
	}
}

//The Update() function runs every frame. It does two things; first it checks
for the various hotkeys. Left and right arrow keys move between trials. ‘S’
primes a trial to start when the participant enters the room. ‘R’ ends the
trial when the card has been flipped. ‘Enter’ or ‘C’ tries to connect to the
bridging device at the currently input IP address. The IP address can only be
changed when waiting on the first trial.

//The second part of Update() checks if the trial is primed and then starts the
trial the next time the tracking software detects 5 consecutive frames of
movement from the head tracker. Because no updates occur when the participant
is outside the room, this detects when the participant has entered the room and
is in range of the camera array.
```

```
void Update()
{
	staticSoundSource.position = new
	Vector3(targetColumnArrayX[tableArray[trialNumber] - 1,
	columnArray[trialNumber] - 1], 0f,
	targetRowArrayZ[tableArray[trialNumber] - 1, rowArray[trialNumber] -
	1]);



	if (Input.GetKeyDown(KeyCode.Keypad9))
	{
		NetworkTransport.Shutdown();
		connected = false;
		initializeNetwork();
	}

	if (Input.GetKeyDown(KeyCode.LeftArrow))
	{
		if (showCue == false)
		{
			if (trialNumber > 0)
			{
				trialNumber--;
				PrintEventRow("10", "Previous trial");
				if (trialNumber % 9 == 8)
				{
					ShuffleCueArray();
				}
			}
			buttonPressedArray[5] = 10;
		}
	}
	if (Input.GetKeyDown(KeyCode.RightArrow))
	{
		if (showCue == false)
		{
			if (trialNumber < 90)
			{
				trialNumber++;
				PrintEventRow("11", "Next Trial");
				if (trialNumber % 9 == 0)
				{
					ShuffleCueArray();
				}
			}
		buttonPressedArray[6] = 10;
		}
	}
	if (Input.GetKeyDown(KeyCode.R))
	{
		EndTrial();
		buttonPressedArray[2] = 10;
	}
	if (Input.GetKeyDown(KeyCode.S))
	{
		if (primed == true)
		{
			primed = false;
			PrintEventRow("21", "Removed Priming");
		}
		else
```

```
        {
        if (showCue == false)
                {
                primed = true;
                PrintEventRow("20", "Primed");
                }
        }
        buttonPressedArray[0] = 10;
}
if (Input.GetKeyDown(KeyCode.Return) || Input.GetKeyDown(KeyCode.C))
{
        Connect();
}
if (trialNumber == 0)
{
        if (connected == false)
        {
                InputIP();
        }
}
else
{
        InputCueType();
}

posUpdate = false;
if (posHold[1, 0] != headTrack.position.x)
{
        posUpdate = true;
        posHold[1, 0] = headTrack.position.x;
}
if (posHold[1, 1] != headTrack.position.y)
{
        posUpdate = true;
        posHold[1, 1] = headTrack.position.y;
}
if (posHold[1, 2] != headTrack.position.z)
{
        posUpdate = true;
        posHold[1, 2] = headTrack.position.z;
}
if (posUpdate)
{
        if (inRangeSwitch < 5)
        {
                inRangeSwitch++;
        }
        else
        {
                if (primed == true)
                {
                        StartTrial();
                }
        }
        System.IO.File.AppendAllText(record.fileName2, Time.time + "," +
        headTrack.position.x + "," + headTrack.position.y + "," +
        headTrack.position.z + "," + headTrack.rotation.eulerAngles.x +
        "," + headTrack.rotation.eulerAngles.y + "," +
        headTrack.rotation.eulerAngles.z + "," + trialNumber + "," +
        internaltrialNumber + "," + showCue + "," +
        targetColumnArrayX[tableArray[trialNumber]-1,
        columnArray[trialNumber]-1] + "," +
```

```
                targetRowArrayZ[tableArray[trialNumber]-1, rowArray[trialNumber]-
                1] + System.Environment.NewLine);
        }
        else
        {
                if (inRangeSwitch > 0)
                {
                        inRangeSwitch--;
                }
        }

        updateNetwork();

        CalculateOutGoingValues();
}

//This function is called each frame by Update(). It checks for incoming
messages from the bridging device. These include connections, disconnections,
forced trial starts, forced trial ends, an output error code, or an output
incorrect flip.
void updateNetwork()
{
        int recHostId;
        int recConnectionId;
        int recChannelId;
        byte[] recBuffer = new byte[24];
        int bufferSize = 24;
        int dataSize;
        byte error;
        NetworkEventType recNetworkEvent = NetworkTransport.Receive(out
        recHostId, out recConnectionId, out recChannelId, recBuffer, bufferSize,
        out dataSize, out error);
        switch (recNetworkEvent)
        {
        case NetworkEventType.Nothing:
                break;
        case NetworkEventType.ConnectEvent:
                connected = true;
                hasConnected = true;
                Debug.Log("incoming connection event received");
                System.IO.File.AppendAllText(record.fileName1, Time.time +
                ",8,Connected" + System.Environment.NewLine);
                break;
        case NetworkEventType.DataEvent:
                Debug.Log("incoming message event received: ");
                messageType = System.BitConverter.ToInt32(recBuffer, 0);
                if (messageType == 0)
                {
                        StartTrial();
                }
                if (messageType == 1)
                {
                        EndTrial();
                }
                if (messageType == 2)
                {
                        PrintEventRow("93", "Mobile Error");
                }
                if (messageType == 3)
                {
                        PrintEventRow("3", "Incorrect Flip");
                }
```

```
                if (messageType < 5)
                {
                        netUpdate = true;
                }
                break;
        case NetworkEventType.DisconnectEvent:
                connected = false;
                Debug.Log("remote client event disconnected");
                System.IO.File.AppendAllText(record.fileName1, Time.time +
                ",9,Disconnected" + System.Environment.NewLine);
                connectionCounter = 10;
                break;
        }
}

/*
This function is called each frame by Update(). It calculates the values which
are sent through to the bridging program every frame. These values are
1: the angle between the current facing of the participant and the facing
required to be looking straight towards the hidden target. Rounded to 3 d.p..
2: The distance in meters from the participant to the hidden target. Rounded to
3 d.p..
3: The current trial number.
4: The current trial's type of cue.
5: Whether the cue should be shown currently
*/
void CalculateOutGoingValues()
{
        rotationToTarget = Mathf.Rad2Deg *
        Mathf.Atan2((targetColumnArrayX[tableArray[trialNumber]-
        1,columnArray[trialNumber]-1] - headTrack.position.x),
        (targetRowArrayZ[tableArray[trialNumber]-1, rowArray[trialNumber]-1] -
        headTrack.position.z));
        distanceToTarget = Mathf.Sqrt((headTrack.position.x -
        targetColumnArrayX[tableArray[trialNumber]-1, columnArray[trialNumber]-
        1]) * (headTrack.position.x -
        targetColumnArrayX[tableArray[trialNumber]-1, columnArray[trialNumber]-
        1]) + (headTrack.position.z - targetRowArrayZ[tableArray[trialNumber]-1,
        rowArray[trialNumber]-1]) * (headTrack.position.z -
        targetRowArrayZ[tableArray[trialNumber]-1, rowArray[trialNumber]-1]));
        rotationOffset = ((rotationToTarget - headTrack.rotation.eulerAngles.y)
        + 720) % 360;
        if (rotationOffset > ((headTrack.rotation.eulerAngles.y -
        rotationToTarget) + 720) % 360)
        {
                rotationOffset = -((headTrack.rotation.eulerAngles.y -
                rotationToTarget) + 720) % 360;
        }
        outMessageType = 6;
        outMessageValue1 = Mathf.FloorToInt(rotationOffset * 1000f);
        outMessageValue2 = Mathf.FloorToInt(distanceToTarget * 1000f);
        outMessageValue3 = trialNumber;
        outMessageValue4 = cueArray[trialNumber % 9];
        if (showCue == true)
        {
                outMessageValue5 = 1;
        }
        else
        {
                outMessageValue5 = 0;
        }
```

```
        if (connected)
        {
                SendSocketMessage();
        }
}

//This function is called each frame by CalculateOutGoingValues(). It sends the
5 synchronising values through to the bridging program.
public void SendSocketMessage()
{
        byte error;
        byte[] buffer = new byte[24];

        buffer[0] = System.BitConverter.GetBytes(outMessageType)[0];
        buffer[1] = System.BitConverter.GetBytes(outMessageType)[1];
        buffer[2] = System.BitConverter.GetBytes(outMessageType)[2];
        buffer[3] = System.BitConverter.GetBytes(outMessageType)[3];
        buffer[4] = System.BitConverter.GetBytes(outMessageValue1)[0];
        buffer[5] = System.BitConverter.GetBytes(outMessageValue1)[1];
        buffer[6] = System.BitConverter.GetBytes(outMessageValue1)[2];
        buffer[7] = System.BitConverter.GetBytes(outMessageValue1)[3];
        buffer[8] = System.BitConverter.GetBytes(outMessageValue2)[0];
        buffer[9] = System.BitConverter.GetBytes(outMessageValue2)[1];
        buffer[10] = System.BitConverter.GetBytes(outMessageValue2)[2];
        buffer[11] = System.BitConverter.GetBytes(outMessageValue2)[3];
        buffer[12] = System.BitConverter.GetBytes(outMessageValue3)[0];
        buffer[13] = System.BitConverter.GetBytes(outMessageValue3)[1];
        buffer[14] = System.BitConverter.GetBytes(outMessageValue3)[2];
        buffer[15] = System.BitConverter.GetBytes(outMessageValue3)[3];
        buffer[16] = System.BitConverter.GetBytes(outMessageValue4)[0];
        buffer[17] = System.BitConverter.GetBytes(outMessageValue4)[1];
        buffer[18] = System.BitConverter.GetBytes(outMessageValue4)[2];
        buffer[19] = System.BitConverter.GetBytes(outMessageValue4)[3];
        buffer[20] = System.BitConverter.GetBytes(outMessageValue5)[0];
        buffer[21] = System.BitConverter.GetBytes(outMessageValue5)[1];
        buffer[22] = System.BitConverter.GetBytes(outMessageValue5)[2];
        buffer[23] = System.BitConverter.GetBytes(outMessageValue5)[3];

        int bufferSize = 24;

        NetworkTransport.Send(socketId, connectionId, myReliableChannelId,
        buffer, bufferSize, out error);
}

/*
This function is called each frame and displays a large number of debug
information to the screen including the connection statistics, the current
positon of the target and the participant, and the trial/cue information.

It also displays buttons for controlling the experiment. These include buttons
for priming the experiment. forcing the trail to start in absence of the
participant. ending the trial. marking an error and moving back and forward
between trials.
*/
void OnGUI()
{
        GUI.Label(new Rect(10, 10, 500, 200), "clientIP= " + clientIP);
        GUI.Label(new Rect(10, 30, 500, 200), "Connected= " + connected);
```

```
GUI.Label(new Rect(10, 60, 500, 200), "Head Tracking Angle= " +
headTrack.rotation.eulerAngles.y);
GUI.Label(new Rect(10, 80, 500, 200), "Angle From Head To Target= " +
rotationToTarget);
GUI.Label(new Rect(10, 100, 500, 200), "Angle Offset= " +
rotationOffset);

GUI.Label(new Rect(10, 130, 500, 200), "headTrack.position.x= " +
headTrack.position.x);
GUI.Label(new Rect(10, 150, 500, 200), "headTrack.position.y= " +
headTrack.position.y);
GUI.Label(new Rect(10, 170, 500, 200), "headTrack.position.z= " +
headTrack.position.z);

GUI.Label(new Rect(300, 10, 500, 200), "Absolute Trial Number= " +
internaltrialNumber);
GUI.Label(new Rect(300, 30, 500, 200), "Trial Number= " + trialNumber);
GUI.Label(new Rect(300, 50, 500, 200), "Table= " +
tableArray[trialNumber]);
GUI.Label(new Rect(300, 70, 500, 200), "Position= " +
positionArray[trialNumber]);
GUI.Label(new Rect(300, 90, 500, 200), "Row= " + rowArray[trialNumber]);
GUI.Label(new Rect(300, 110, 500, 200), "Column= " +
columnArray[trialNumber]);
GUI.Label(new Rect(300, 130, 500, 200), "Cue ID= " +
cueArray[trialNumber % 9]);
GUI.Label(new Rect(300, 150, 500, 200), "Visual Cue= " +
cueArrayVisual[trialNumber % 9]);
GUI.Label(new Rect(300, 170, 500, 200), "Sound Cue= " +
cueArraySound[trialNumber % 9]);

GUI.Label(new Rect(500, 10, 500, 200), "Trial Primed= " + primed);
GUI.Label(new Rect(500, 30, 500, 200), "Show Cue= " + showCue);

for (int i = 0; i < 9; i++)
{
       if ((i - (trialNumber % 9)) == 0)
       {
              GUI.Label(new Rect(700 + 30 * i, 10, 500, 200), "" +
              (trialNumber - (trialNumber % 9) + i) + "*");
       }
       else
       {
              GUI.Label(new Rect(700 + 30 * i, 10, 500, 200), "" +
              (trialNumber - (trialNumber % 9) + i));
       }
       GUI.Label(new Rect(700 + 30 * i, 30, 500, 200), "" +
       cueArray[i]);
}
if (showCue == true)
{
       GUI.Label(new Rect(500, 50, 500, 200), "Trial Time= " +
       Mathf.FloorToInt(Time.time - timeFloat));
}
else
{
       GUI.Label(new Rect(500, 50, 500, 200), "Trial Time= Pending");
}
if (inRangeSwitch == 0)
{
       GUI.Label(new Rect(500, 70, 500, 200), "Detection= Outside
       Range");
```

```
}
else if (inRangeSwitch == 5)
{
	GUI.Label(new Rect(500, 70, 500, 200), "Detection= Tracking");
}
else
{
	GUI.Label(new Rect(500, 70, 500, 200), "Detection=
Transitioning");
}

if (hasConnected == true)
{
	if (connected == false)
	{
		GUI.colour = Colour.red;
		GUI.Box(new Rect(30, 30, Screen.width - 60, Screen.height
		- 60), "DISCONNECTED!");
		GUI.colour = Colour.white;
	}
}

if (primed == true)
{
	GUI.colour = Colour.green;
	GUI.Box(new Rect(200, 200, Screen.width - 400, Screen.height -
	400), "PRIMED!");
	GUI.colour = Colour.white;
}

if (soundCued == true)
{
	if (Time.time > timeFloat + 0.5)
	{
		soundCued = false;
		if (((cueArray[trialNumber % 9]) == 4) ||
		(cueArray[trialNumber % 9] == 5) || (cueArray[trialNumber
		% 9] == 6))
		{
			soundSourceStatic.Play();
			playingStaticSound = true;
		}
		if ((cueArray[trialNumber % 9] == 7) ||
		(cueArray[trialNumber % 9] == 8) || (cueArray[trialNumber
		% 9] == 9))
		{
			AudioCueTransform.position = headTrack.position;
			AudioCueScript.Fire();
			soundSourceDynamic.Play();
		}
	}
}

if (buttonPressedArray[0] == 0)
{
	if (GUI.Button(new Rect(10, Screen.height - 90, 120, 80), "Prep
	Auto Start"))
	{
		if (primed == true)
		{
			primed = false;
			PrintEventRow("21", "Removed Priming");
```

```
                }
                else
                {
                        if (showCue == false)
                        {
                                primed = true;
                                PrintEventRow("20", "Primed");
                        }
                }
        buttonPressedArray[0] = 10;
        }
}
else
{
        buttonPressedArray[0]--;
}
if (buttonPressedArray[1] == 0)
{
        if (GUI.Button(new Rect(150, Screen.height - 90, 120, 80), "Force
        Start"))
        {
                StartTrial();
                buttonPressedArray[1] = 10;
        }
}
else
{
        buttonPressedArray[1]--;
}


if (buttonPressedArray[2] == 0)
{
        if (GUI.Button(new Rect(290, Screen.height - 90, 120, 80), "End
        Trial"))
        {
                EndTrial();
                buttonPressedArray[2] = 10;
        }
}
else
{
        buttonPressedArray[2]--;
}

if (buttonPressedArray[3] == 0)
{
        if (GUI.Button(new Rect(430, Screen.height - 90, 120, 80), "Dead
        Trial"))
        {
                PrintEventRow("91", "Dead Trial");
                buttonPressedArray[3] = 10;
        }
}
else
{
        buttonPressedArray[3]--;
}

if (buttonPressedArray[4] == 0)
{
```

```
        if (GUI.Button(new Rect(570, Screen.height - 90, 120, 80), "Other
        Error"))
        {
                PrintEventRow("92", "Other Error");
                buttonPressedArray[4] = 10;
        }
}
else
{
        buttonPressedArray[4]--;
}

if (buttonPressedArray[5] == 0)
{
        if (GUI.Button(new Rect(710, Screen.height - 90, 120, 80),
        "Previous Trial"))
        {
                if (showCue == false)
                {
                        if (trialNumber > 0)
                        {
                        trialNumber--;
                        PrintEventRow("10", "Previous trial");
                                if (trialNumber % 9 == 8)
                                {
                                        ShuffleCueArray();
                                }
                        }
                buttonPressedArray[5] = 10;
                }
        }
}
else
{
        buttonPressedArray[5]--;
}

if (buttonPressedArray[6] == 0)
{
        if (GUI.Button(new Rect(850, Screen.height - 90, 120, 80), "Next
        Trial"))
        {
                if (showCue == false)
                {
                        if (trialNumber < 90)
                        {
                        trialNumber++;
                        PrintEventRow("11", "Next Trial");
                                if (trialNumber % 9 == 0)
                                {
                                        ShuffleCueArray();
                                }
                        }
                buttonPressedArray[6] = 10;
                }
        }
}
else
{
        buttonPressedArray[6]--;
}
```

```
        windowRect = GUI.Window(0, windowRect, DoMyWindow, "Glass Screen",
        blackWindowStyle);
}

//This is an overlay window which displays what the participant will be seeing
on the Glass. It is called each frame by OnGUI().
void DoMyWindow(int windowID)
{
        if (connected)
        {
                switch (cueArray[trialNumber % 9])
                {
                case 1: //None
                case 4:
                case 7:
                        // Do Nothing
                        break;
                case 2: //Map
                case 5:
                case 8:
                        if (showCue)
                        {
                                GUI.DrawTexture(new Rect(0, 0, 640, 360), texMap,
                                ScaleMode.StretchToFill);
                                GUI.DrawTexture(new Rect(mapXArray[trialNumber],
                                mapYArray[trialNumber], 70, 70), texHotSpot,
                                ScaleMode.ScaleAndCrop);
                        }
                        break;
                case 3: //Arrow
                case 6:
                case 9:
                        if (showCue)
                        {
                                GUI.Label(new Rect(0, 190, 640, 70),
                                distanceToTarget.ToString("F1") + "m", glassStyle);

                                if (rotationOffset > 0)
                                {
                                        GUI.DrawTexture(new Rect(320 + 280f *
                                        (rotationOffset / 180f), 263, 38, 75),
                                        texRightArrow, ScaleMode.ScaleAndCrop);
                                        GUI.DrawTexture(new Rect(320, 270, 280f *
                                        (rotationOffset / 180f), 60), texWhiteSpace,
                                        ScaleMode.StretchToFill);
                                        GUI.DrawTexture(new Rect(320 - 41, 260, 82,
                                        81), texCentralDot, ScaleMode.ScaleAndCrop);
                                }
                                else if (rotationOffset < 0)
                                {
                                        GUI.DrawTexture(new Rect(282 + 280f *
                                        (rotationOffset / 180f), 263, 38, 75),
                                        texLeftArrow, ScaleMode.ScaleAndCrop);
                                        GUI.DrawTexture(new Rect(320 + 280f *
                                        (rotationOffset / 180f), 270, 280f * (-
                                        rotationOffset / 180f), 60), texWhiteSpace,
                                        ScaleMode.StretchToFill);
                                        GUI.DrawTexture(new Rect(320 - 42, 260, 82,
                                        81), texCentralDot, ScaleMode.ScaleAndCrop);
                                }
                                else
                                {
```

```
                                        GUI.DrawTexture(new Rect(320 - 42, 260, 82,
                                        81), texCentralDot, ScaleMode.ScaleAndCrop);
                                }
                        }
                        break;
                }
        }
        else
        {
                GUI.Label(new Rect(0, 210, 640, 50), "Waiting for Connection",
                glassStyle);
        }

        GUI.DragWindow(new Rect(0, 0, 10000, 10000));
}

//This starts the trial. It plays the sound cue, if one is set for the trial,
begins the trial timer and records the start time in the log. Called by the
start button, hotkey, the priming section of Update()
void StartTrial()
{
        PrintEventRow("0", "Start Trial");
        showCue = true;
        primed = false;
        soundCued = true;
        timeFloat = Time.time;
        internaltrialNumber++;
}

//This ends the trial. It stops any sound cues, ends the trial timer and
records the end time in the log.
void EndTrial()
{
        PrintEventRow("1", "End Trial");
        showCue = false;
        buttonPressedArray[2] = 10;
        if (playingStaticSound)
        {
                playingStaticSound = false;
                soundSourceStatic.Stop();
        }
}

//Attempts the connection to the bridging device after the IP address has been
set.
public void Connect()
{
        byte error;
        connectionId = NetworkTransport.Connect(socketId, clientIP, socketPort,
        0, out error);
        Debug.Log("Connected to server. ConnectionId: " + connectionId);
        System.IO.File.AppendAllText(record.fileName1, Time.time * 1000 +
        ",7,ConnectScriptRan" + System.Environment.NewLine);
}

//Prints an error code to the log. Called by error buttons on the desktop and
on the phone.
void PrintEventRow (string errorCode, string str)
{
        System.IO.File.AppendAllText(record.fileName1, Time.time + "," +
        errorCode + "," + str + "," + trialNumber + "," + internaltrialNumber +
```

```
        "," + (Time.time - timeFloat) + "," + cueArray[trialNumber % 9] + "," +
        tableArray[trialNumber] + "," + positionArray[trialNumber] + "," +
        columnArray[trialNumber] + "," + rowArray[trialNumber] +
        System.Environment.NewLine);
}

//Runs once when the program begins. Sets the framerate.
void Awake()
{
        Application.targetFrameRate = 60;
}

//During trial 1, the number keys change the IP Address for connecting to the
bridging program.
void InputIP()
{
        if (Input.anyKeyDown)
        {

                if (Input.GetKeyDown(KeyCode.Alpha0))
                {
                        clientIP = clientIP + "0";
                }
                if (Input.GetKeyDown(KeyCode.Alpha1))
                {
                        clientIP = clientIP + "1";
                }
                if (Input.GetKeyDown(KeyCode.Alpha2))
                {
                        clientIP = clientIP + "2";
                }
                if (Input.GetKeyDown(KeyCode.Alpha3))
                {
                        clientIP = clientIP + "3";
                }
                if (Input.GetKeyDown(KeyCode.Alpha4))
                {
                        clientIP = clientIP + "4";
                }
                if (Input.GetKeyDown(KeyCode.Alpha5))
                {
                        clientIP = clientIP + "5";
                }
                if (Input.GetKeyDown(KeyCode.Alpha6))
                {
                        clientIP = clientIP + "6";
                }
                if (Input.GetKeyDown(KeyCode.Alpha7))
                {
                        clientIP = clientIP + "7";
                }
                if (Input.GetKeyDown(KeyCode.Alpha8))
                {
                        clientIP = clientIP + "8";
                }
                if (Input.GetKeyDown(KeyCode.Alpha9))
                {
                        clientIP = clientIP + "9";
                }
                if (Input.GetKeyDown(KeyCode.Period))
                {
                        clientIP = clientIP + ".";
```

```
		}
		if (Input.GetKeyDown(KeyCode.Backspace))
		{
			clientIP = "";
		}
	}
}

//During all other trials, the number keys change the cue type for the current
trial.
void InputCueType()
{
	if (Input.anyKeyDown)
	{
		if (Input.GetKeyDown(KeyCode.Alpha1))
		{
			cueArray[trialNumber % 9] = 1;
			cueArrayVisual[trialNumber % 9] = "None";
			cueArraySound[trialNumber % 9] = "None";
		}
		if (Input.GetKeyDown(KeyCode.Alpha2))
		{
			cueArray[trialNumber % 9] = 2;
			cueArrayVisual[trialNumber % 9] = "Map";
			cueArraySound[trialNumber % 9] = "None";
		}
		if (Input.GetKeyDown(KeyCode.Alpha3))
		{
			cueArray[trialNumber % 9] = 3;
			cueArrayVisual[trialNumber % 9] = "Arrow";
			cueArraySound[trialNumber % 9] = "None";
		}
		if (Input.GetKeyDown(KeyCode.Alpha4))
		{
			cueArray[trialNumber % 9] = 4;
			cueArrayVisual[trialNumber % 9] = "None";
			cueArraySound[trialNumber % 9] = "Static";
		}
		if (Input.GetKeyDown(KeyCode.Alpha5))
		{
			cueArray[trialNumber % 9] = 5;
			cueArrayVisual[trialNumber % 9] = "Map";
			cueArraySound[trialNumber % 9] = "Static";
		}
		if (Input.GetKeyDown(KeyCode.Alpha6))
		{
			cueArray[trialNumber % 9] = 6;
			cueArrayVisual[trialNumber % 9] = "Arrow";
			cueArraySound[trialNumber % 9] = "Static";
		}
		if (Input.GetKeyDown(KeyCode.Alpha7))
		{
			cueArray[trialNumber % 9] = 7;
			cueArrayVisual[trialNumber % 9] = "None";
			cueArraySound[trialNumber % 9] = "Dynamic";
		}
		if (Input.GetKeyDown(KeyCode.Alpha8))
		{
			cueArray[trialNumber % 9] = 8;
			cueArrayVisual[trialNumber % 9] = "Map";
			cueArraySound[trialNumber % 9] = "Dynamic";
		}
```

```
                        if (Input.GetKeyDown(KeyCode.Alpha9))
                        {
                                cueArray[trialNumber % 9] = 9;
                                cueArrayVisual[trialNumber % 9] = "Arrow";
                                cueArraySound[trialNumber % 9] = "Dynamic";
                        }
                }
        }
}
```

Appendix 5.3: Google Glass Receiver Program and Bridging Program for Experiment 5

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using System.Net;
using UnityEngine.Networking;

using LostPolygon.AndroidBluetoothMultiplayer;

public class FadeControl : MonoBehaviour
{
	public SpriteRenderer mainSprite;

	float rotationPercent;

	int outMessageType;
	int outMessageValue1;
	int outMessageValue2;
	int outMessageValue3;
	int outMessageValue4;

	int screenState = 0;
	int incomingMessageType;
	float incomingValue1 = 0;
	float incomingValue2 = 0;

	public Texture texCentralDot;
	public Texture texLeftArrow;
	public Texture texRightArrow;
	public Texture texWhiteSpace;
	public Texture texMap;
	public Texture texHotSpot;

	float rotationOffset;
	float distanceToTarget;

	int showCue = 0;
	bool primed = false;

	public GUIStyle style;
	public GUIStyle glassStyle;
	public GUIStyle buttonStyle;

	int[] buttonPressedArray = { 0, 0, 0, 0, 0, 0 };
	int[] tableArray = { 1, 6, 5, 4, 4, 3, 6, 6, 1, 6, 6, 5, 2, 1, 5, 6, 4, 2, 6,
	5, 3, 6, 1, 2, 4, 2, 2, 6, 1, 5, 3, 2, 1, 6, 6, 3, 2, 1, 1, 5, 1, 2, 3, 6, 6,
	6, 2, 2, 3, 2, 6, 6, 6, 5, 4, 2, 2, 1, 2, 2, 1, 2, 3, 1, 1, 4, 2, 1, 6, 5, 3,
	6, 2, 4, 6, 1, 1, 4, 3, 3, 1, 6, 3, 2, 6, 6, 6, 1, 6, 3 };
	int[] positionArray = { 6, 4, 4, 1, 9, 6, 12, 15, 15, 11, 14, 3, 2, 3, 1, 8, 7,
	10, 15, 3, 5, 11, 10, 11, 4, 15, 10, 5, 13, 4, 3, 6, 15, 5, 2, 1, 9, 6, 14, 1,
	9, 9, 2, 6, 15, 15, 11, 6, 8, 15, 1, 12, 2, 3, 9, 14, 11, 7, 10, 3, 15, 6, 6,
	15, 14, 2, 12, 14, 7, 2, 4, 5, 10, 6, 1, 2, 3, 8, 8, 1, 14, 1, 5, 4, 15, 8, 11,
	2, 13, 7 };
	int[] mapXArray = { 214, 331, 209, 364, 404, 299, 283, 355, 310, 259, 331, 184,
	186, 262, 184, 307, 364, 164, 355, 184, 279, 259, 310, 186, 364, 208, 164, 355,
	262, 209, 299, 208, 310, 355, 283, 259, 208, 214, 286, 184, 286, 208, 279, 259,
	355, 355, 186, 208, 279, 208, 259, 283, 283, 184, 404, 186, 186, 238, 164, 208,
	310, 208, 299, 310, 286, 384, 208, 286, 283, 209, 259, 355, 164, 404, 259, 238,
	262, 384, 279, 259, 286, 259, 279, 164, 355, 307, 259, 238, 307, 259 };
	int[] mapYArray = { 259, 52, 27, 206, 141, 145, 8, 8, 237, 8, 8, 27, 237, 281,
	52, 30, 141, 165, 8, 27, 145, 8, 259, 165, 174, 141, 165, 52, 237, 27, 177,
```

```
213, 237, 52, 52, 177, 189, 259, 237, 52, 259, 189, 177, 30, 8, 8, 165, 213,
112, 141, 52, 8, 52, 27, 141, 141, 165, 259, 165, 237, 237, 213, 145, 237, 237,
206, 165, 237, 30, 52, 145, 52, 165, 174, 52, 281, 281, 141, 112, 177, 237, 52,
145, 213, 8, 30, 8, 281, 8, 112 };
int trialNumber = 0;
int cue = 0;
float timeFloat = 0f;

int myReliableChannelId;
int socketId;
int socketPort = 8888;
int connectionId;
bool connected = false;
bool bConnected = false;
bool cConnected = false;
bool updated = false;

bool isServer = false;
bool isClient = false;

bool showIntroImage = true;

public GameObject ActorPrefab;
public Transform floatHolder;

string uuid = "b909e9da-e83a-4fdb-b96c-1fa7334b756d";
private const string kLocalIp = "127.0.0.1"; // An IP for Network.Connect(),
must always be 127.0.0.1
private const int kPort = 28000; // Local server IP. Must be the same for
client and server

private bool _initResult;
private BluetoothMultiplayerMode _desiredMode = BluetoothMultiplayerMode.None;

//Runs once at the start of the program. Sets the screen to stay awake for the
whole experiment. Sets the Glass as the client, or otherwise sets the device as
the server. If it's the server, it sets up the socket to socket Wi-Fi
connection.
void Start()
{
	Screen.sleepTimeout = SleepTimeout.NeverSleep;

	if (AndroidBluetoothMultiplayer.GetCurrentDevice().Name == "Mark Glass's
	Glass")
	{
		isClient = true;
	}
	else
	{
		isServer = true;
	}
	if (isServer)
	{
		NetworkTransport.Init();
		ConnectionConfig config = new ConnectionConfig();
		myReliableChannelId = config.AddChannel(QosType.StateUpdate);
		int maxConnections = 10;
		HostTopology topology = new HostTopology(config, maxConnections);
		socketId = NetworkTransport.AddHost(topology, socketPort);
		Debug.Log("Socket Open. SocketId is: " + socketId);
	}
}
```

```
//Runs every frame. Runs ServerUpdate() if this is the bridging device. Runs
ClientUpdate() if this is the Glass.
void Update()
{
	if (isServer)
	{
		ServerUpdate();
	}

	if (isClient)
	{
		ClientUpdate();
	}
}

//Runs every frame on the bridging device. First checks for oncoming
synchronisation messages and then passes them on to the Glass. If the transfer
game object does not exist, it creates one. If more than once exists it deletes
them all.
void ServerUpdate()
{
	ReceiveWifi();
	UpdateBluetooth();
	if (floatHolder != null)
	{
		NetworkTransform[] objects =
		FindObjectsOfType(typeof(NetworkTransform)) as
		NetworkTransform[];
		if (objects.Length > 1)
		{
			foreach (NetworkTransform obj in objects)
			{
				Destroy(obj.gameObject);
			}
		}
	}
	else
	{
		if (Network.isServer)
		{
			GameObject newObject = Network.Instantiate(ActorPrefab,
			Vector3.zero, Quaternion.identity, 0) as GameObject;
			floatHolder = newObject.GetComponent<Transform>();
		}
	}
}

//Runs every frame on the Glass. If a transfer game object exists, it takes the
values from it and stores them. Otherwise, puts up a disconnected screen and
searches for a new one.
void ClientUpdate()
{
	if (floatHolder != null)
	{
		screenState = Mathf.RoundToInt(floatHolder.position.x);
		incomingValue1 = floatHolder.position.y;
		incomingValue2 = floatHolder.position.z;
	}
	else
	{
		screenState = 0;
		incomingValue1 = 0;
```

```
                incomingValue2 = 0;
                NetworkTransform[] objects =
                FindObjectsOfType(typeof(NetworkTransform)) as
                NetworkTransform[];
                if (objects != null)
                {
                        floatHolder = objects[0].gameObject.transform;
                }
                else
                {
                        screenState = 99;
                }
        }
}

//Runs every frame on the bridging device when called by ServerUpdate(). This
takes in network messages such as connections, disconnections and synch
messages. When a synch message is received it updates the current screen
(screenState).
void ReceiveWifi()
{
        int recHostId;
        int recConnectionId;
        int recChannelId;
        byte[] recBuffer = new byte[24];
        int bufferSize = 24;
        int dataSize;
        byte error;
        NetworkEventType recNetworkEvent = NetworkTransport.Receive(out
        recHostId, out recConnectionId, out recChannelId, recBuffer, bufferSize,
        out dataSize, out error);
        connectionId = recConnectionId;
        switch (recNetworkEvent)
        {
        case NetworkEventType.Nothing:
                break;
        case NetworkEventType.ConnectEvent:
                connected = true;
                break;
        case NetworkEventType.DataEvent:
                updated = true;
                incomingMessageType = System.BitConverter.ToInt32(recBuffer, 0);
                incomingValue1 = (System.BitConverter.ToInt32(recBuffer, 4)) /
                1000f;
                incomingValue2 = (System.BitConverter.ToInt32(recBuffer, 8)) /
                1000f;
                trialNumber = (System.BitConverter.ToInt32(recBuffer, 12));
                cue = (System.BitConverter.ToInt32(recBuffer, 16));
                showCue = (System.BitConverter.ToInt32(recBuffer, 20));
                break;
        case NetworkEventType.DisconnectEvent:
                connected = false;
                break;
        }
        if (connected)
        {
                if (updated)
                {
                        updated = false;
                        if (incomingMessageType == 6)
                        {
                                rotationOffset = incomingValue1;
```

```
                        distanceToTarget = incomingValue2;
                    }
                }
                if (showCue == 1)
                {
                    screenState = cue;
                }
                else
                {
                    screenState = 1;
                }
            }
        }

        //Runs every frame on the bridging device when called by ServerUpdate(). It
        passes through the current screen, and the two values necessary to generate the
        cues through to the Glass.
        void UpdateBluetooth()
        {
            if (floatHolder != null)
            {
                switch (cue)
                {
                case 1: //None
                case 4:
                case 7:
                    floatHolder.position = new Vector3(0f, 0f, 0f);
                    break;
                case 2: //Map
                case 5:
                case 8:
                    if (showCue == 1)
                    {
                    floatHolder.position = new Vector3(2f,
                    mapXArray[trialNumber], mapYArray[trialNumber]);
                    }
                    else
                    {
                    floatHolder.position = new Vector3(0f, 0f, 0f);
                    }
                    break;
                case 3: //Arrow
                case 6:
                case 9:
                    if (showCue == 1)
                    {
                    floatHolder.position = new Vector3(1f, rotationOffset,
                    distanceToTarget);
                    }
                    else
                    {
                    floatHolder.position = new Vector3(0f, 0f, 0f);
                    }
                    break;
                }
            }
        }

        //Runs every frame. Runs ServerOnGUI() if this is the bridging device. Runs
        ClientOnGUI() if this is the Glass.
        private void OnGUI()
        {
```

```
        if (isServer)
        {
                ServerOnGUI();
        }

        if (isClient)
        {
                ClientOnGUI();
        }
}

//Runs every frame on the bridging device. The first block of code displays a
large amount of debug information and current trial information to the bridging
device so that the mobile experimenter can access this information at the same
time as the experimenter operating the desktop computer. The second block
provides 4 buttons. These buttons force a trial start, force a trial stop, mark
an error or note an incorrect card flip. The third block of code requests
Bluetooth permissions if they are not already given.
void ServerOnGUI()
{
        GUI.Label(new Rect(690, 50 * 0 + 20, 500, 80), "--INCOMING--", style);
        GUI.Label(new Rect(690, 50 * 1 + 20, 500, 80), "connected: " +
        connected, style);
        GUI.Label(new Rect(690, 50 * 2 + 20, 500, 80), "incomingMessageType: " +
        incomingMessageType, style);
        GUI.Label(new Rect(690, 50 * 3 + 20, 500, 80), "incomingValue1: " +
        incomingValue1, style);
        GUI.Label(new Rect(690, 50 * 4 + 20, 500, 80), "incomingValue2: " +
        incomingValue2, style);

        GUI.Label(new Rect(1220, 50 * 0 + 20, 500, 80), "--OUTGOING--", style);
        GUI.Label(new Rect(1220, 50 * 1 + 20, 500, 80), "connected: " +
        bConnected, style);
        if (floatHolder != null)
        {
                GUI.Label(new Rect(1220, 50 * 2 + 20, 500, 80), "screen: " +
                floatHolder.position.x, style);
                GUI.Label(new Rect(1220, 50 * 3 + 20, 500, 80), "outgoingValue1:
                " + floatHolder.position.y, style);
                GUI.Label(new Rect(1220, 50 * 4 + 20, 500, 80), "outgoingValue2:
                " + floatHolder.position.z, style);
                }
        else
        {
                GUI.Label(new Rect(1220, 50 * 2 + 20, 500, 80), "no floatHolder
                available", style);
        }

        GUI.Label(new Rect(690, 50 * 7 + 20, 500, 80), "--TRIAL INFO--", style);
        GUI.Label(new Rect(690, 50 * 8 + 20, 500, 80), "Current Trial: " +
        trialNumber, style);
        GUI.Label(new Rect(690, 50 * 9 + 20, 500, 80), "Current Block: " +
        (1+Mathf.FloorToInt((trialNumber*1f)/9)), style);
        GUI.Label(new Rect(690, 50 * 10 + 20, 500, 80), "Target Table: " +
        tableArray[trialNumber], style);
        GUI.Label(new Rect(690, 50 * 11 + 20, 500, 80), "Target Position: " +
        positionArray[trialNumber], style);
        GUI.Label(new Rect(690, 50 * 12 + 20, 500, 80), "Cue ID: " + cue,
        style);
        if (cue == 0)
        {
```

```
        GUI.Label(new Rect(690, 50 * 13 + 20, 500, 80), "Visual Cue: -",
style);
}
else if (cue % 3 == 1)
{
        GUI.Label(new Rect(690, 50 * 13 + 20, 500, 80), "Visual Cue:
        None", style);
}
else if (cue % 3 == 2)
{
        GUI.Label(new Rect(690, 50 * 13 + 20, 500, 80), "Visual Cue:
        Map", style);
}
else
{
        GUI.Label(new Rect(690, 50 * 13 + 20, 500, 80), "Visual Cue:
        Arrow", style);
}
if (cue == 0)
{
        GUI.Label(new Rect(690, 50 * 14 + 20, 500, 80), "Auditory Cue: -
        ", style);
}
else if (cue < 4)
{
        GUI.Label(new Rect(690, 50 * 14 + 20, 500, 80), "Auditory Cue:
        None", style);
}
else if (cue < 7)
{
        GUI.Label(new Rect(690, 50 * 14 + 20, 500, 80), "Auditory Cue:
        Static", style);
}
else
{
        GUI.Label(new Rect(690, 50 * 14 + 20, 500, 80), "Auditory Cue:
        Dynamic", style);
}
GUI.Label(new Rect(690, 50 * 15 + 20, 500, 80), "Showing Cue: " +
showCue, style);
if (showCue == 1)
{
        GUI.Label(new Rect(690, 50 * 16 + 20, 500, 80), "Trial Time: " +
        Mathf.FloorToInt(Time.time - timeFloat), style);
}
else
{
        GUI.Label(new Rect(690, 50 * 16 + 20, 500, 80), "Trial Time:
        Pending", style);
}



if (buttonPressedArray[0] == 0)
{
        if (GUI.Button(new Rect(30, 30, 300, 200), "Force" +
        System.Environment.NewLine + "Start", buttonStyle))
        {
                buttonPressedArray[0] = 10;
                outMessageType = 0;
                SendSocketMessage();
        }
```

```
}
else
{
	buttonPressedArray[0]--;
}

if (buttonPressedArray[1] == 0)
{
	if (GUI.Button(new Rect(360, 30, 300, 200), "Force" +
	System.Environment.NewLine + "End", buttonStyle))
	{
		buttonPressedArray[1] = 10;
		outMessageType = 1;
		SendSocketMessage();
	}
}
else
{
	buttonPressedArray[1]--;
}

if (buttonPressedArray[2] == 0)
{
	if (GUI.Button(new Rect(30, 260, 300, 200), "Note" +
	System.Environment.NewLine + "Error", buttonStyle))
	{
		buttonPressedArray[2] = 10;
		outMessageType = 2;
		SendSocketMessage();
	}
}
else
{
	buttonPressedArray[2]--;
}

if (buttonPressedArray[3] == 0)
{
	if (GUI.Button(new Rect(360, 260, 300, 200), "Incorrect" +
	System.Environment.NewLine + "Flip", buttonStyle))
	{
		buttonPressedArray[3] = 10;
		outMessageType = 3;
		SendSocketMessage();
	}
}
else
{
	buttonPressedArray[3]--;
}

if (_initResult)
{
	BluetoothMultiplayerMode currentMode =
	AndroidBluetoothMultiplayer.GetCurrentMode();
	if (currentMode == BluetoothMultiplayerMode.None)
	{
		if (AndroidBluetoothMultiplayer.GetIsBluetoothEnabled())
		{
			AndroidBluetoothMultiplayer.StartServer(kPort);
		}
		else
```

```
                    {
                        AndroidBluetoothMultiplayer.RequestEnableDiscoverab
                        ility(120);
                        _desiredMode = BluetoothMultiplayerMode.Server;
                    }
                }
            }
        }

        //Runs every frame on Glass. Brings up an error if the synchronising game
        object is lost. It runs the function to draw any active cues to the screen.
        Finally, it requests Bluetooth permissions if they are not already active.
        void ClientOnGUI()
        {
            if (floatHolder == null)
            {
                GUI.Label(new Rect(0, 100, 640, 50), "no floatHolder", style);
            }

            drawScreens();
            if (_initResult)
            {
                BluetoothMultiplayerMode currentMode =
                AndroidBluetoothMultiplayer.GetCurrentMode();
                if (currentMode == BluetoothMultiplayerMode.None)
                {
                    if (AndroidBluetoothMultiplayer.GetIsBluetoothEnabled())
                    {
                        BluetoothDevice[] bluetoothDeviceList =
                        AndroidBluetoothMultiplayer.GetBondedDevices();
                        AndroidBluetoothMultiplayer.Connect(bluetoothDevice
                        List[0].Address, kPort);
                    }
                    else
                    {
                        AndroidBluetoothMultiplayer.RequestEnableDiscoverab
                        ility(120);
                        _desiredMode = BluetoothMultiplayerMode.Client;
                    }
                }
            }
        }

        //Runs every frame on Glass when called by ClientOnGUI(). Draws the arrow cue
        or map cue if appropriate. If no connection is available, it displays a warning
        message instead.
        void drawScreens()
        {
            if (cConnected)
            {
                if (screenState == 1)
                {
                    GUI.Label(new Rect(0, 190, 640, 70),
                    incomingValue2.ToString("F1") + "m", glassStyle);

                    if (incomingValue1 > 0)
                    {
                        GUI.DrawTexture(new Rect(320 + 280f *
                        (incomingValue1 / 180f), 263, 38, 75),
                        texRightArrow, ScaleMode.ScaleAndCrop);
```

```
                        GUI.DrawTexture(new Rect(320, 270, 280f *
                        (incomingValue1 / 180f), 60), texWhiteSpace,
                        ScaleMode.StretchToFill);
                        GUI.DrawTexture(new Rect(320 - 41, 260, 82, 81),
                        texCentralDot, ScaleMode.ScaleAndCrop);
                }
                else if (incomingValue1 < 0)
                {
                        GUI.DrawTexture(new Rect(282 + 280f *
                        (incomingValue1 / 180f), 263, 38, 75),
                        texLeftArrow, ScaleMode.ScaleAndCrop);
                        GUI.DrawTexture(new Rect(320 + 280f *
                        (incomingValue1 / 180f), 270, 280f * (-
                        incomingValue1 / 180f), 60), texWhiteSpace,
                        ScaleMode.StretchToFill);
                        GUI.DrawTexture(new Rect(320 - 42, 260, 82, 81),
                        texCentralDot, ScaleMode.ScaleAndCrop);
                }
                else
                {
                        GUI.DrawTexture(new Rect(320 - 42, 260, 82, 81),
                        texCentralDot, ScaleMode.ScaleAndCrop);
                }
        }
        if (screenState == 2)
        {
                GUI.DrawTexture(new Rect(0, 0, 640, 360), texMap,
                ScaleMode.StretchToFill);
                GUI.DrawTexture(new Rect(incomingValue1, incomingValue2,
                70, 70), texHotSpot, ScaleMode.ScaleAndCrop);
        }
    }
    else
    {
        GUI.Label(new Rect(0, 210, 640, 50), "Waiting for Connection",
        glassStyle);
    }
}

//Called by the bridging device whenever a button press needs to send a message
back to the desktop program.
public void SendSocketMessage()
{
    byte error;
    byte[] buffer = new byte[24];

    buffer[0] = System.BitConverter.GetBytes(outMessageType)[0];
    buffer[1] = System.BitConverter.GetBytes(outMessageType)[1];
    buffer[2] = System.BitConverter.GetBytes(outMessageType)[2];
    buffer[3] = System.BitConverter.GetBytes(outMessageType)[3];

    int bufferSize = 24;

    NetworkTransport.Send(socketId, connectionId, myReliableChannelId,
    buffer, bufferSize, out error);
}

//The remaining functions register and handle the Bluetooth connection and
synchronisation event. These are provided by the AndroidBluetoothMultiplayer
plugin library.
private void Awake()
{
```

```
        // Setting the UUID. Must be unique for every application
        _initResult = AndroidBluetoothMultiplayer.Initialize(uuid);

        // Enabling verbose logging. See log cat!
        AndroidBluetoothMultiplayer.SetVerboseLog(true);

        // Registering the event delegates
        AndroidBluetoothMultiplayer.ListeningStarted +=
        OnBluetoothListeningStarted;
        AndroidBluetoothMultiplayer.ListeningStopped +=
        OnBluetoothListeningStopped;
        AndroidBluetoothMultiplayer.AdapterEnabled += OnBluetoothAdapterEnabled;
        AndroidBluetoothMultiplayer.AdapterEnableFailed +=
        OnBluetoothAdapterEnableFailed;
        AndroidBluetoothMultiplayer.AdapterDisabled +=
        OnBluetoothAdapterDisabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnabled +=
        OnBluetoothDiscoverabilityEnabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnableFailed +=
        OnBluetoothDiscoverabilityEnableFailed;
        AndroidBluetoothMultiplayer.ConnectedToServer +=
        OnBluetoothConnectedToServer;
        AndroidBluetoothMultiplayer.ConnectionToServerFailed +=
        OnBluetoothConnectionToServerFailed;
        AndroidBluetoothMultiplayer.DisconnectedFromServer +=
        OnBluetoothDisconnectedFromServer;
        AndroidBluetoothMultiplayer.ClientConnected +=
        OnBluetoothClientConnected;
        AndroidBluetoothMultiplayer.ClientDisconnected +=
        OnBluetoothClientDisconnected;
        AndroidBluetoothMultiplayer.DevicePicked += OnBluetoothDevicePicked;
}

// Don't forget to unregister the event delegates!
void OnDestroy()
{
        AndroidBluetoothMultiplayer.ListeningStarted -=
        OnBluetoothListeningStarted;
        AndroidBluetoothMultiplayer.ListeningStopped -=
        OnBluetoothListeningStopped;
        AndroidBluetoothMultiplayer.AdapterEnabled -= OnBluetoothAdapterEnabled;
        AndroidBluetoothMultiplayer.AdapterEnableFailed -=
        OnBluetoothAdapterEnableFailed;
        AndroidBluetoothMultiplayer.AdapterDisabled -=
        OnBluetoothAdapterDisabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnabled -=
        OnBluetoothDiscoverabilityEnabled;
        AndroidBluetoothMultiplayer.DiscoverabilityEnableFailed -=
        OnBluetoothDiscoverabilityEnableFailed;
        AndroidBluetoothMultiplayer.ConnectedToServer -=
        OnBluetoothConnectedToServer;
        AndroidBluetoothMultiplayer.ConnectionToServerFailed -=
        OnBluetoothConnectionToServerFailed;
        AndroidBluetoothMultiplayer.DisconnectedFromServer -=
        OnBluetoothDisconnectedFromServer;
        AndroidBluetoothMultiplayer.ClientConnected -=
        OnBluetoothClientConnected;
        AndroidBluetoothMultiplayer.ClientDisconnected -=
        OnBluetoothClientDisconnected;
        AndroidBluetoothMultiplayer.DevicePicked -= OnBluetoothDevicePicked;
}
```

```
void OnBackToMenu()
{
	// Gracefully closing all Bluetooth connectivity and loading the menu
	try
	{
		AndroidBluetoothMultiplayer.StopDiscovery();
		AndroidBluetoothMultiplayer.Stop();
	}
	catch
	{
	//
	}
}

private void OnBluetoothListeningStarted()
{
	Network.InitializeServer(4, kPort, false);
}

private void OnBluetoothListeningStopped()
{
	AndroidBluetoothMultiplayer.Stop();
}

private void OnBluetoothDevicePicked(BluetoothDevice device)
{
	AndroidBluetoothMultiplayer.Connect(device.Address, kPort);
}

private void OnBluetoothClientDisconnected(BluetoothDevice device)
{
	bConnected = false;
}

private void OnBluetoothClientConnected(BluetoothDevice device)
{
	bConnected = true;
}

private void OnBluetoothDisconnectedFromServer(BluetoothDevice device)
{
	cConnected = false;
	Network.Disconnect();
}

private void OnBluetoothConnectedToServer(BluetoothDevice device)
{
	cConnected = true;
	Network.Connect(kLocalIp, kPort);
}

private void OnBluetoothAdapterEnabled()
{
	switch (_desiredMode)
	{
	case BluetoothMultiplayerMode.Server:
		Network.Disconnect();
		AndroidBluetoothMultiplayer.StartServer(kPort);
		break;
	case BluetoothMultiplayerMode.Client:
		Network.Disconnect();
```

```
                BluetoothDevice[] bluetoothDeviceList =
                AndroidBluetoothMultiplayer.GetBondedDevices();
                AndroidBluetoothMultiplayer.Connect(bluetoothDeviceList[0].Addres
                s, kPort);
                break;
            }
            _desiredMode = BluetoothMultiplayerMode.None;
        }


        private void OnFailedToConnect(NetworkConnectionError error)
        {
            AndroidBluetoothMultiplayer.Stop();
        }

}
```

Appendix 5.4: Example Output and Data Processing for Experiment 5

Example data output for an Experiment 5 event log:

Start Time:	2016	11	2	12	9	49	675			
Time	EventType	EventDescription	TrialID	TrialCounter	TimeSinceLastStartEvent	CueID	TableID	PositionID	ColumnID	RowID
1.057033	7	ConnectScriptRan								
1.073652	8	Connected								
174.2734	20	Primed	0	0	174.2734	8	1	6	1	2
193.6686	0	Start Trial	0	0	193.6686	8	1	6	1	2
199.5344	3	Incorrect Flip	0	1	5.865814	8	1	6	1	2
200.2852	3	Incorrect Flip	0	1	6.616547	8	1	6	1	2
201.1667	1	End Trial	0	1	7.498047	8	1	6	1	2
209.7302	11	Next Trial	1	1	16.0616	4	6	4	4	1
212.8459	20	Primed	1	1	19.17732	4	6	4	4	1
226.9595	0	Start Trial	1	1	33.29094	4	6	4	4	1
238.4226	3	Incorrect Flip	1	2	11.46309	4	6	4	4	1
240.6386	3	Incorrect Flip	1	2	13.67902	4	6	4	4	1
241.3735	3	Incorrect Flip	1	2	14.41399	4	6	4	4	1
246.97	3	Incorrect Flip	1	2	20.0105	4	6	4	4	1
247.7202	3	Incorrect Flip	1	2	20.76062	4	6	4	4	1
248.8194	3	Incorrect Flip	1	2	21.8598	4	6	4	4	1
249.4525	3	Incorrect Flip	1	2	22.49295	4	6	4	4	1
250.6522	3	Incorrect Flip	1	2	23.69269	4	6	4	4	1
253.5682	1	End Trial	1	2	26.60864	4	6	4	4	1
258.9662	11	Next Trial	2	2	32.00667	9	5	4	2	2
275.5781	20	Primed	2	2	48.61855	9	5	4	2	2
291.5574	0	Start Trial	2	2	64.59789	9	5	4	2	2
310.152	3	Incorrect Flip	2	3	18.5946	9	5	4	2	2
...	...	...	...	...	...	...	...	...	...	...

The experiment 5 event log lists the time in seconds from the start of the program, an event code for sorting events, a description of the event, the trial number where it falls in the experiment order, a second unique trial number (for if a trial needs to be run twice or otherwise repeated and then later differentiated), the current cue type and the location of the card within the room. The log data as compressed into usable row data using the following macro:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
Sub Extraction()

Dim CurrentRow As Integer
Dim PrintColumn As Integer
Dim IncorrectFlips As Integer
Dim Looping As Boolean


CurrentRow = 5
IncorrectFlips = 0
Looping = True
PrintColumn = 15
Cells(2, PrintColumn - 1).Value = "TrialOrder"
```

```
Cells(3, PrintColumn - 1).Value = "TrialID"
Cells(4, PrintColumn - 1).Value = "CompletionTime"
Cells(5, PrintColumn - 1).Value = "IncorrectFlips"
Cells(6, PrintColumn - 1).Value = "AudioCue"
Cells(7, PrintColumn - 1).Value = "VisualCue"
Cells(8, PrintColumn - 1).Value = "TableID"
Cells(9, PrintColumn - 1).Value = "TableColumn"
Cells(10, PrintColumn - 1).Value = "TableRow"
Cells(11, PrintColumn - 1).Value = "StartTime"
Cells(12, PrintColumn - 1).Value = "Overflow Flips"


Do While (Looping = True)
   If (Cells(CurrentRow, 1).Value > 0) Then
      If (Cells(CurrentRow, 2).Value = 1) Then
         Cells(2, PrintColumn).Value = Cells(CurrentRow, 5).Value
         Cells(3, PrintColumn).Value = Cells(CurrentRow, 4).Value
         Cells(4, PrintColumn).Value = Cells(CurrentRow, 6).Value
         Cells(5, PrintColumn).Value = IncorrectFlips
         IncorrectFlips = 0
         Cells(6, PrintColumn).Value = Application.WorksheetFunction.Floor((Cells(CurrentRow,
7).Value - 1) / 3, 1)
         Cells(7, PrintColumn).Value = (Cells(CurrentRow, 7).Value - 1) - 3 * Cells(6, PrintColumn).Value
         Cells(8, PrintColumn).Value = Cells(CurrentRow, 8).Value
         Cells(9, PrintColumn).Value = Cells(CurrentRow, 10).Value
         Cells(10, PrintColumn).Value = Cells(CurrentRow, 11).Value
         Cells(11, PrintColumn).Value = Cells(CurrentRow, 1).Value - Cells(CurrentRow, 6).Value

         PrintColumn = PrintColumn + 1
      End If
      If (Cells(CurrentRow, 2).Value = 0) Then
         If (Cells(12, PrintColumn - 1).Value = "Overflow Flips") Then

         Else
            Cells(12, PrintColumn - 1).Value = IncorrectFlips
            IncorrectFlips = 0
         End If
      End If
      If (Cells(CurrentRow, 2).Value = 3) Then
         IncorrectFlips = IncorrectFlips + 1
      End If
      If (Cells(CurrentRow, 2).Value = 9) Then
         Cells(2, PrintColumn).Value = "Disconnect"
         Cells(3, PrintColumn).Value = Cells(2, PrintColumn).Value
         Cells(4, PrintColumn).Value = Cells(2, PrintColumn).Value
         Cells(5, PrintColumn).Value = Cells(2, PrintColumn).Value
         Cells(6, PrintColumn).Value = Cells(2, PrintColumn).Value
         PrintColumn = PrintColumn + 1
```

```
    End If
    If (Cells(CurrentRow, 2).Value = 91) Then
      Cells(2, PrintColumn).Value = "DeadTrial"
      Cells(3, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(4, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(5, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(6, PrintColumn).Value = Cells(2, PrintColumn).Value
      PrintColumn = PrintColumn + 1
    End If
    If (Cells(CurrentRow, 2).Value = 92) Then
      Cells(2, PrintColumn).Value = "OtherError"
      Cells(3, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(4, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(5, PrintColumn).Value = Cells(2, PrintColumn).Value
      Cells(6, PrintColumn).Value = Cells(2, PrintColumn).Value
      PrintColumn = PrintColumn + 1
    End If
    If (CurrentRow = 1000) Then
      Looping = False
      Cells(2, 1).Value = "Overflow"
    End If
    CurrentRow = CurrentRow + 1
  Else
    Looping = False
  End If

Loop


End Sub
```

//

This macro rearranges the data so that each column represents a single trial. The rearranged data looks as follows:

TrialID	0	1	2	3	4	5	6	7	8	9	10	...
CompletionTime	7	27	22	10	42	31	34	18	5	18	15	...
IncorrectFlips	2	8	3	0	24	9	10	0	0	3	5	...
AudioCue	2	1	2	1	2	0	1	0	0	1	1	...
VisualCue	1	0	2	1	0	0	2	2	1	0	2	...
TableID	1	6	5	4	4	3	6	6	1	6	6	...
TableColumn	1	4	2	1	3	3	2	5	5	1	4	...
TableRow	2	1	2	1	3	2	3	3	3	3	3	...
StartTime	194	227	292	337	371	450	518	582	626	654	105	...

The data is then manually collated into SPSS accessible sheets collected by a single variable. For example, below is the collated sheet for CompletionTime:

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	7	27	22	10	42	31	34	18	5	18	15	15	41	22	9	40	6	15	22
2	49	31	41	41	6	41	8	16	5	12	43	41	5	15	7	8	13	12	40
3	54	11	27	7	18	41	41	8	46	39	8	15	25	8	6	42	12	7	32
4	35	16	46	17	8	8	54	22	5	24	9	10	42	20	7	9	12	41	12
5	42	29	14	20	10	42	41	15	17	24	19	42	28	18	10	18	18	36	20
6	64	11	11	41	14	42	18	9	13	40	10	24	16	5	8	16	10	23	8
7	36	21	41	10	7	12	15	7	4	11	9	41	14	9	6	8	12	9	19
8	29	16	9	15	41	16	41	9	136	21	15	15	6	41	17	41	6	28	29
9	20	41	41	7	30	20	11	30	19	10	41	9	20	33	7	21	16	18	8
10	88	13	42	10	39	8	68	42	40	10	41	28	7	12	20	16	16	16	41
11	15	21	36	6	23	6	35	41	17	32	12	6	28	15	14	12	9	6	27
12	24	11	9	93	25	83	41	41		42	21	7	5	28	23	41	10	35	19
13	26	43	29	17	12	37	22	29	9	17	41	10	12	7	11	15	24	46	41
14	61	24	14	21	9	70	14	20	41	12	9	8	15	6	16	8	40	42	41
15	31	11	8	6	8	12	11	19	7	7	8	20	30	12	12	23	8	10	15
16	11	41	20	14	45	6	17	8	13	14	11	12	20	3	6	31	6	20	11
17	13	34	26	7	37	40	36	23	9	42	14	41	7	9	9	10	26	14	9
18	30	13	10	41	28	8	12	41	5	9	20	10	4	3	13	30	19	44	15
19	11	37	34	48	56	20	40	9	9	13	26	9	12	8	7	9	11	5	15
20	8	22	41	10	6	12	36	17	4	6	8	5	12	14	6	10	19	11	41
21	46	15	21	41	17	6	9	41	13	41	26	13	24	14	13	24	6	7	7
22	41	24	48	9	41	9	27	25	44	20	18	25	16	6	43	19	11	7	40
23	42	20	43	8	15	22	10	21	13	9	48	12	7	14	17	49	16	7	10
24	7	15	7	12	47	14	42	28	8	11	10	9	7	5	41	15	27	7	31
25	30	118	64	42	8	41	63	11	23	41	23	19	4	29	19	10	6	41	8
26	77	43	31	48	44	32	15	14	18	21	19	10	7	15	19	28	9	32	9
27	42	41	13	7	7	12	28	13	42	16	17	39	5	41	15	12	6	9	17
28	15	12	18	41	13	13	32	9	24	26	14	10	5	19	8	16	23	42	19
29	33	9	60	9	12	27	6	20	17	5	15	11	9	24	5	41	6	41	16
30	14	8	9	14	13	8	41	17	22	5	19	11	13	5	41	6	18	21	5
31	45	14	7	40	10	10	7	7	7	10	18	8	3	3	5	26	40	8	41
32	21	13	45	10	8	25	31	60	43	19	41	15	6	10	36	32	10	16	29
33	27	14	15	16	15	15	41	9	6	19	9	10	27	5	34	8	10	18	11
34	7	34	9	26	7	10	22	38	41	9	9	14	41	5	12	41	17	6	41
35	14	26	9	6	18	30	20	38	30	9	11	6	33	4	9	42	11	8	48
36	7	41	37	14	7	41	9	20	12	26	11	41	15	2	11	8	12	6	9
AVERAGE	31	25	27	22	21	24	28	22	22	19	19	17	16	14	15	22	15	20	22
VARIANCE	415.1033	377.8874	263.5756	352.0891	220.8894	334.3728	262.5298	175.0258	578.8523	138.3339	139.545	143.9761	132.1137	107.5872	117.6711	170.3523	75.23479	191.2	174.3979
STANDARD DEVIATION										11.76154	11.81292	11.999	11.49407	10.37243	10.84763	13.05191	8.673799	13.82751	13.20598
TableID	1	6	5	4	4	3	6	6	1	6	6	5	2	1	5	6	4	2	6
TableColumn	1	4	2	1	3	3	2	5	5	1	4	1	2	3	1	3	1	1	5
TableRow	2	1	2	1	3	2	3	3	3	3	3	2	1	1	1	2	3	4	3

The first block of trials is marked in yellow because these were not used in the analysis. The data analyzed in this form using SPSS. Average cue effects were extracted by fitting a model of

$$\text{expected response time} = \text{average table response time} + \text{cue effect}_{[1\text{-}9]}$$

using the Excel Solver function. The results of this fit are shown below. Results are in seconds. This data were used to compare the effects of the cues on search time.

Model 1 Parameters			T1 Mean	T2 Mean	T3Mean	T4 Mean	T5Mean	T6 Mean	
			13.42253	16.22467	17.7611	15.03116	15.83036	19.28604	
CueType	1	2	3	4	5	6	7	8	9
1									
2	21.7	-10.0	-1.1	1.5	-10.6	1.6	16.9	-7.2	-2.8
3	12.8	-9.6	-2.0	-5.7	-11.9	5.7	0.6	-9.6	-0.2
4	12.1	-10.6	-5.5	1.3	-9.7	-6.5	19.7	-8.5	-5.4
5	21.9	-7.6	7.7	18.1	-5.2	4.1	8.2	-5.4	5.9
6	17.2	-8.9	-3.3	-4.2	-9.6	-5.6	9.3	-9.1	1.9
7	16.6	-9.5	-5.7	-1.0	-12.0	-4.3	6.0	-10.8	-3.9
8	21.7	-9.8	-4.1	11.6	-9.4	-2.7	17.5	-6.2	-2.5
9	13.7	-10.2	0.7	5.5	-7.8	4.6	18.4	-9.2	4.2
10	17.8	-9.0	-1.7	3.8	-9.8	8.6	21.9	-8.2	1.1
11	12.2	-11.0	-0.8	-6.7	-8.5	-5.1	16.4	-11.6	6.7
12	20.7	-6.7	-1.5	15.9	-7.7	0.7	19.3	-8.7	7.6
13	15.9	-6.5	12.9	7.8	-6.4	9.1	14.4	-6.5	-1.2
14	11.8	-8.7	-0.7	8.2	-10.7	0.0	18.9	-9.4	-1.5
15	1.5	-10.8	-3.9	0.2	-10.0	-6.3	5.4	-10.0	-2.8
16	9.8	-10.5	4.5	8.3	-10.1	-7.3	7.2	-9.8	-1.8
17	25.2	-8.5	-1.5	6.4	-9.0	-4.9	10.9	-7.3	-2.5
18	19.2	-9.6	-5.5	2.7	-10.0	-3.8	13.5	-11.0	-5.1
19	6.7	-11.2	-3.7	1.4	-11.9	4.5	18.1	-8.2	-4.9
20	-3.2	-12.4	-6.4	-9.5	-11.1	-7.1	9.5	-9.6	-8.8
21	8.7	-11.5	2.2	11.9	-5.0	4.3	10.0	-7.3	7.8
22	13.9	-8.2	2.2	-2.1	-8.3	3.6	12.6	-6.1	1.3
23	22.6	-8.3	0.4	-3.7	-9.6	0.9	20.7	-2.8	-2.5
24	18.5	-8.7	-6.4	0.0	-9.2	-6.8	18.5	-10.6	-8.0
25	13.4	-11.7	2.2	1.6	-6.0	7.0	10.5	-6.7	-0.5
26	7.3	-8.5	1.6	-6.0	-9.4	4.1	-5.0	-6.8	3.4
27	25.6	-7.3	4.2	14.5	-10.8	-5.4	15.6	-10.5	-3.8
28	22.8	-7.9	-5.6	-3.0	-8.4	3.9	9.8	-10.4	-3.8
29	8.0	-10.7	-2.0	2.6	-11.7	-3.3	12.3	-9.9	-4.6
30	4.8	-9.3	-6.8	-2.8	-10.5	0.6	22.9	-14.5	-2.3
31	17.2	-11.3	-9.7	-5.7	-10.6	-7.4	13.9	-10.7	-8.4
32	16.2	-5.5	-2.5	4.9	-9.2	-0.7	17.2	-9.8	-0.9
33	8.3	-9.8	-6.5	-1.5	-10.5	-7.1	10.0	-10.7	-5.3
34	19.9	-10.3	-5.3	-1.3	-8.9	-7.2	4.3	-11.6	4.0
35	21.3	-9.6	-7.6	-5.7	-11.3	-4.9	10.5	-10.6	-3.3
36	17.8	-9.6	-4.5	-3.2	-11.3	-5.9	3.0	-10.9	-4.7

No Sound			Static Sound			Dynamic Sound		
None	Map	Arrow	None	Map	Arrow	None	Map	Arrow
14.9	-9.4	-1.9	1.9	-9.5	-1.1	12.5	-9.0	-1.3

Appendix 5.5: Paper Materials for Experiment 5

Consent form for Experiment 5.

HIT Lab NZ & Department of Psychology

Telephone: +64 3 364 2349

Email: amit.barde@pg.canterbury.ac.nz

matt.ward@pg.canterbury.ac.nz

A REAL WORLD SEARCH TASK USING SPATIALISED AUDITORY AND VISUAL CUES

CONSENT FORM FOR PARTCIPANTS

- ☐ I have been given a full explanation of this project and have had the opportunity to ask questions.
- ☐ I understand what is required of me if I agree to take part in the research.
- ☐ I understand that participation is voluntary and I may withdraw at any time without penalty. Withdrawal of participation will also include withdrawal of any information I have provided that this should remain practically achievable.
- ☐ I understand that any information or opinions I provide will be kept confidential to the researchers Amit Barde & Matt Ward, and that any published or reported results will not identify participants.
- ☐ I understand that a thesis is a public document and will be available through the UC Library.
- ☐ I have been informed of and understand the risks associated with taking part in this research study.

Please Turn Overleaf..

- ☐ I understand that all data collected for the study will be kept in locked ad secure facilities and/or in password protected electronic form and will be destroyed after ten years.
- ☐ I understand that I can contact the researchers, Amit Barde & Matt Ward or their supervisors listed below for further information.
  Amit Barde: amit.barde@pg.canterbury.ac.nz
  Matt Ward: matt.ward@pg.canterbury.ac.nz
  Mark Billinghurst: mark.billinghurst@canterbury.ac.nz
  Rob Lindeman: gogo@hitlabnz.org
- ☐ I would like a summary of the results of the project.

- ☐ By signing below, I agree to participate in this research project.

Name: _____ Signed: _____ Date: __ __ / __ __ / 2016

Email address (for report of findings, if applicable): _____

Please return the consent form once you've checked all the boxes and signed it to the researcher present.

Matt Ward

Amit Barde

Information sheet for Experiment 4

HIT Lab NZ & Department of Psychology
Telephone: +64 3 364 2349
Email: amit.barde@pg.canterbury.ac.nz
matt.ward@pg.canterbury.ac.nz

A REAL WORLD SEARCH TASK USING SPATIALISED AUDITORY AND VISUAL CUES

INFORMATION SHEET FOR PARTCIPANTS

Welcome! You are invited to take part in this study which will form a part of the investigators, Amit Barde & Matt Ward's, doctoral theses. The purpose of this study is to understand and evaluate the use of auditory and visual cues in a real world search task. This study will form a part of an ongoing investigation in to the design and use of hybrid wearable interfaces.

As part of this study you will be required to perform a simple search task. The sequence of the experimental procedure will be as follows:

- You will be presented with a bone conduction headset and the Google Glass, both of which you will be required to wear.
- Visual cues will be presented to you via the Google Glass, while auditory cues will be delivered over the bone conduction headset.
- You are to navigate within the test space using these cues and find a target object.
- To help familiarise you with the experiment, a trial run with all the cues presented once will be carried out.
- The main experiment will run for 50 minutes.
- You will receive a $10 Westfield Voucher as compensation.
- Your participation in this study is voluntary.
- You have the right to withdraw at any stage without penalty up until the point the data is entered in to the computer.
- If you choose to withdraw, all data relating to your participation in the study will be discarded.

Please Turn Overleaf..

- Your participation in this study and the data generated as a result will be treated with the utmost regard to anonymity and confidentiality.
- No personal information will be collected. Only population demographics such as age, sex and frequency of mobile and/or wearable device usage will be collected.
- You are entitled to receive a copy of the results by contacting the researchers at the conclusion of this study. The results of this study may be published. Any publication of the results will not involve divulging participant details.
- Only demographic data that has been collected will be made public as required for publication. All data generated as part of this study will be kept in a secure location i.e. in the research's personal locker and/or a password protected computer located in a secure, access controlled facility. Only the researcher and this supervisor(s) will have access to this data.
- The data will be archived and stored for 10 years before being destroyed as university regulation governing the storage and destruction of data accumulated as part of a doctoral thesis stipulates. The results of this study will be used in part or full, as part of the researcher, Amit Barde's, doctoral thesis.

Please indicate to the researcher on the consent form if you would like to receive a copy of the summary of the results of this project.

This project is being carried out by the researchers, Amit Barde & Matt Ward, as a requirement for PhD degree under the supervision of Dr. Rob Lindeman, Dr. Gun Lee and Dr. Mark Billinghurst. They can all be contacted via their respective email addresses given below.

Dr. Rob Lindeman: gogo@hitlabnz.org
Dr. Gun Lee: gun.lee@canterbury.ac.nz
Dr. Mark Billinghurst: mark.billinghurst@canterbury.ac.nz

They will be pleased to discuss any concerns you may have about participation in the project.

This project has been reviewed and approved by the University of Canterbury Human Ehtics Committee, and participants should address any complaints to The Chair, Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ehtics@canterbury.ac.nz).

If you agree to participate in the study, you are asked to complete the consent form and return it to the experimenters.

Amit Barde & Matt Ward.

Please take this information sheet with you when you leave

Pre-study questionnaire for Experiment 5:

P

A REAL WORLD SEARCH TASK USING SPATIALISED AUDITORY AND VISUAL CUES

PRE-TASK QUESTIONNAIRE

1. Gender: Male ☐ Female ☐ Other ☐

2. Age: ☐☐ years

3. Do you have any previous experience with auditory experiments? Conducting, participating or assisting.
   Yes ☐ No ☐

4. Do you own a mobile device? I.e. cell phone, tablet, mp3 player etc.
   Yes ☐ No ☐

5. If you've ticked **Yes** in Q4, how often do you use one or more of these devices?
   Everyday ☐ Few times a week ☐ Few times a month ☐ Few times a year ☐

6. Do you use headphones/earphones to listen to music and/or other material?
   Yes ☐ No ☐

7. If you've ticked **Yes** in Q6, how often do you use headphones/earphones?
   Everyday ☐ Few times a week ☐ Few times a month ☐ Few times a year ☐

8. How often do you use headphones/earphones to listen to material on any of your mobile devices?
   Everyday ☐ Few times a week ☐ Few times a month ☐ Few times a year ☐

9. Do you have normal hearing in both ears?
   Yes ☐ No ☐

Please Turn Overleaf...

P

10. Are you familiar with a wearable computing device? E.g. Google Glass.

Yes ☐ **(Go to Q.11)** No ☐ **(Return the questionnaire)**

11. How often have you used such a device?

Once ☐
Everyday ☐
Few times a month ☐
Few times a year ☐

12. How would you describe your experience using such a device?

Excellent, I found it useful ☐
Good, with potential to be a lot more useful ☐
Neutral, the novelty of using such a device wore off quickly ☐
Bad, I do not see how such a device could be helpful ☐
Other (Please explain the experience in your own words) ☐

Please Turn Overleaf...

Post-study questionnaire for Experiment 5:

P

A REAL WORLD SEARCH TASK USING SPATIALISED AUDITORY AND VISUAL CUES

POST-STUDY QUESTIONNAIRE

1. Which visual cue did you prefer?
   Environment Mapped ☐ Dynamic Arrow ☐

2. Which auditory cue did you prefer?
   Static (Constant Ping) ☐ Dynamic ☐

3. Which cueing condition did you most prefer? (**1 – Best, 3 – Worst**)

Visual Cue	Auditory Cue	A combination of the two
☐	☐	☐

4. You were fastest at finding the target using the (**1 – Fastest, 3 – Slowest**)

Visual Cue	Auditory Cue	A combination of the two
☐	☐	☐

5. Would you use any of these cues in a real-world scenario? E.g. To find an address or while driving etc.
   Yes ☐ **Go to Question 6** No ☐ **Go to Question 7**

P

6. Which one?

   Visual Cue ☐ Auditory Cue ☐ A combination of the two ☐

   Depends on the scenario (driving, walking etc.) ☐

   Go to Question 7

7. Why? (Provide a short explanation for your choice)

Appendix 6.0: Ethics, Programming and Paper Materials for Experiment 6

The University of Canterbury Human Ethic Approval for this experiment is presented in Appendix 6.1.

Experiment 6 shares its design and code with experiment 5. A new type of cue was added, the Ring Map cue. The updated DrawScreens() function for the receiver program is available in appendix 6.2. Similarly, analysis methods are also shared. See appendices 5.2, 5.3, and 5.4 for details of the experiment and analysis design and procedure.

Appendix 6.3 contains the paper materials used in this experiment. These include an information sheet, a consent form and a post-experiment questionnaire.

Appendix 6.1: Human Ethics Approval for Experiment 6

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE

Secretary, Rebecca Robinson
Telephone: +64 03 369 4588, Extn 94588
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2017/02/LR-PS

31 January 2017

Matt Ward
Psychology
UNIVERSITY OF CANTERBURY

Dear Matt

Thank you for submitting your low risk application to the Human Ethics Committee for the research proposal titled "OK Google, Where Are My Keys? The Interaction of Physical and Augmented Reality Visual Cues When Searching a Real World Environment".

I am pleased to advise that this application has been reviewed and approved.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 24th January 2017.

With best wishes for your project.

Yours sincerely

pp. R. Robinson

Associate Professor Jane Maidment
Chair, Human Ethics Committee

Appendix 6.2: Update to the experiement 5 receiver program.

```
// This is the updated DrawScreen() function for the 6th experiment. It adds a third cue
type, a map where the target could fall anywhere inside a region rather than at the
central point. Labels and other text was also changed, but the programming otherwise
stayed the same as in the previous experiment. See Appendix 5.3 for more details.
void drawScreens()
{
	if (cConnected)
	{
		if (screenState == 1)
		{
			GUI.Label(new Rect(0, 190, 640, 70),
			incomingValue2.ToString("F1") + "m", glassStyle);
			if (incomingValue1 > 0)
			{
				GUI.DrawTexture(new Rect(320 + 280f * (incomingValue1 /
				180f), 263, 38, 75), texRightArrow,
				ScaleMode.ScaleAndCrop);
				GUI.DrawTexture(new Rect(320, 270, 280f * (incomingValue1
				/ 180f), 60), texWhiteSpace, ScaleMode.StretchToFill);
				GUI.DrawTexture(new Rect(320 - 41, 260, 82, 81),
				texCentralDot, ScaleMode.ScaleAndCrop);
			}
			else if (incomingValue1 < 0)
			{
				GUI.DrawTexture(new Rect(282 + 280f * (incomingValue1 /
				180f), 263, 38, 75), texLeftArrow,
				ScaleMode.ScaleAndCrop);
				GUI.DrawTexture(new Rect(320 + 280f * (incomingValue1 /
				180f), 270, 280f * (-incomingValue1 / 180f), 60),
				texWhiteSpace, ScaleMode.StretchToFill);
				GUI.DrawTexture(new Rect(320 - 42, 260, 82, 81),
				texCentralDot, ScaleMode.ScaleAndCrop);
			} else {
				GUI.DrawTexture(new Rect(320 - 42, 260, 82, 81),
				texCentralDot, ScaleMode.ScaleAndCrop);
			}
		}
		if (screenState == 2)
		{
			GUI.DrawTexture(new Rect(0, 0, 640, 360), texMap,
			ScaleMode.StretchToFill);
			GUI.DrawTexture(new Rect(incomingValue1, incomingValue2, 70, 70),
			texHotSpot, ScaleMode.ScaleAndCrop);
		}
		if (screenState == 3)
		{
			GUI.DrawTexture(new Rect(0, 0, 640, 360), texMap,
			ScaleMode.StretchToFill);
			GUI.DrawTexture(new Rect(incomingValue1, incomingValue2, 100,
			100), texHotRing, ScaleMode.ScaleAndCrop);
		}
	}
	else
	{
		GUI.Label(new Rect(0, 210, 640, 50), "Waiting for Connection",
		glassStyle);
	}
}
```

Appendix 6.3: Paper Materials for Experiment 6

Consent form for experiment 6:

Psychology
Telephone: +64 22 695 0995
Email: matt.ward@pg.canterbury.ac.nz

Ok Google, Where Are My Keys? The Interaction of Physical and Augmented Reality Visual Cues When Searching a Real World Environment
Consent Form for Participants

Include a statement regarding each of the following:

- ☐ I have been given a full explanation of this project and have had the opportunity to ask questions.
- ☐ I understand what is required of me if I agree to take part in the research.
- ☐ I understand that participation is voluntary and I may withdraw at any time without penalty. Withdrawal of participation will also include the withdrawal of any information I have provided should this remain practically achievable.
- ☐ I understand that any information or opinions I provide will be kept confidential to the researcher and his supervisors and that any published or reported results will not identify the participants. I understand that a thesis is a public document and will be available through the UC Library.
- ☐ I understand that all data collected for the study will be kept in locked and secure facilities and/or in password protected electronic form and will be destroyed after ten years.
- ☐ I understand the risks associated with taking part and how they will be managed.
- ☐ I understand that I can contact the researcher Matt Ward (matt.ward@pg.canterbury.ac.nz) or supervisor Deak Helton (deak.helton@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Human Ethics Committee, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz)
- ☐ I would like a summary of the results of the project.
- ☐ By signing below, I agree to participate in this research project.

Name:_____ Signed:_____ Date:_____

Email address *(for report of findings, if applicable)*:_____

Please return this sheet to the experimenter when complete.

Information sheet for experiment 6:

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Department of Psychology
Email: matt.ward@pg.canterbury.ac.nz

OK GOOGLE, WHERE ARE MY KEYS? THE INTERACTION OF PHYSICAL AND AUGMENTED REALITY VISUAL CUES WHEN SEARCHING A REAL WORLD ENVIRONMENT.

INFORMATION SHEET FOR PARTCIPANTS

Welcome! You are invited to take part in this study which will form a part of the investigators, Amit Barde & Matt Ward's, doctoral theses. The purpose of this study is to understand and evaluate the use of auditory and visual cues in a real world search task. This study will form a part of an ongoing investigation in to the design and use of hybrid wearable interfaces.

As part of this study you will be required to perform a simple search task. The sequence of the experimental procedure will be as follows:

- You will be presented with a Google Glass headset, which you will be required to wear.
- Virtual visual cues will be presented to you via the Google Glass.
- You are to navigate within the test space using these cues and find a target object.
- To help familiarise you with the experiment, a trial run with all the cues presented once will be carried out.
- The main experiment will run for 50 minutes.
- You will receive a $10 Westfield Voucher as compensation.
- Your participation in this study is voluntary.
- You have the right to withdraw at any stage without penalty up until the point the data is entered in to the computer.
- If you choose to withdraw, all data relating to your participation in the study will be discarded.
- Your participation in this study and the data generated as a result will be treated with the utmost regard to anonymity and confidentiality.

Please Turn Overleaf...

- No personal information will be collected. Only population demographics such as age, sex and frequency of mobile and/or wearable device usage will be collected.
- You are entitled to receive a copy of the results by contacting the researchers at the conclusion of this study. The results of this study may be published. Any publication of the results will not involve divulging participant details.
- Only demographic data that has been collected will be made public as required for publication. All data generated as part of this study will be kept in a secure location i.e. in the research's personal locker and/or a password protected computer located in a secure, access controlled facility. Only the researcher and this supervisor(s) will have access to this data.
- The data will be archived and stored for 10 years before being destroyed as university regulation governing the storage and destruction of data accumulated as part of a doctoral thesis stipulates. The results of this study will be used in part or full, as part of the researcher, Matt Ward's, doctoral thesis.

Please indicate to the researcher on the consent form if you would like to receive a copy of the summary of the results of this project.

This project is being carried out by the researchers, Matt Ward, as a requirement for PhD degree under the supervision of Dr. Deak Helton (deak.helton@canterbury.ac.nz).

They will be pleased to discuss any concerns you may have about participation in the project.

This project has been reviewed and approved by the University of Canterbury Human Ehtics Committee, and participants should address any complaints to The Chair, Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ehtics@canterbury.ac.nz).

If you agree to participate in the study, you are asked to complete the consent form and return it to the experimenters.

Matt Ward.

Please take this information sheet with you when you leave

Post study questionnaire for experiment 6:

Matt Ward, Experiment 6

Post Study Questionnaire

Study ID: _____

Gender: M / F / O (circle one)

Age: _____

Preferred AR Cue: Map (Dot) / Map (Ring) / Arrows / No Preference (circle one)

For which of the following do you use a smart phone?:

□ Making calls
□ Sending texts
□ Accessing email
□ Social Networking
□ Getting directions
□ Work-specific apps
□ Playing games
□ Watching video
□ Scheduling/ booking appointments
□ Accessing the internet

Notes: _____