Audio Screen: Unsighted Game Mechanics for Mobile Devices

Jonathon Brammer

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science & Software Engineering

University of Washington

2016

Committee:

William Erdly

Min Chen

Kelvin Sung

Program Authorized to Offer Degree:

Computing and Software Systems – Bothell

University of Washington

Abstract

Audio Screen: Unsighted Game Mechanics for Mobile Devices

Jonathon Brammer

Chair of the Supervisory Committee:
William Erdly, Ph.D., Associate Professor
Computing and Software Systems

Historically, most video games lean primarily on visual communication, making them inaccessible to the blind. Non-sighted gaming is beginning to gain some momentum, particularly with the proliferation of mobile devices, whose multiple sensor inputs allow for the exploration of inventive control schemes. The arrival of built-in 3D audio support on iOS also represents a significant opportunity to easily explore audio-only gaming on the mobile platform. This study tests two variations on a paradigm for aurally representing a 2D screen in a virtual 3D audio world, as well as the effects of three input control schemes and a game mechanic called "auto-lock." Users take on the simple task of finding and selecting stationary targets on the screen. The resulting data support the hypothesis that the "Flat" environment permits higher accuracy than

the “Curved” environment, and also shows improvements in speed as well as subjective metrics of usability, enjoyableness, and immersion.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge the members of my supervisory committee, Bill, Min, and Kelvin, for their support and guidance throughout my graduate education at the University of Washington. I approached them each specifically for their clear passion and skill in the art of teaching as well as their dedication to students, and I am proud to have them on my committee. I would also like to thank my family and friends for their support through this challenging process, as well as all the volunteers who gave up their time to help test these hypotheses.

DEDICATION

This work is dedicated to my wife, Kaley,
without whom my post-graduate education would not have been possible nor survivable.

Chapter 1. INTRODUCTION

Video games provide access to increasingly complex and exciting virtual worlds. In these worlds, we can experience adventures that are either not possible, not affordable, or too dangerous in the real world. Unfortunately, these experiences are lost on one particular group who one could argue could benefit from them the most: people who can't see.

Video games rely primarily on visuals, as implied by the phrase itself. Home consoles and custom-built gaming PCs are described and compared in terms of their graphical capabilities. Countless games have been advertised and described with phrases like "cutting edge graphics." This is to be expected, since vision is so fundamental to the life experience of the vast majority of humans that we rely on it without thinking, but in the US, nearly 700,000 people from the ages of 4 to 20 and roughly 3.8 million people from the ages of 18 to 64 suffer from vision loss significant enough that that they can't perform sight-related functions without alternative methods, even when wearing corrective lenses ("Statistical Facts about Blindness in the United States," 2015). Imagine driving a car, playing basketball, or just watching a movie with your eyes closed. These and many other activities are inaccessible to the blind, in part due to rules of the world that we can't change, but in video games, we make our own worlds, and we make our own rules. And yet, we rely primarily on graphics. This does not need to be the case.

The idea of accessible games is not a new concept, but is growing in popularity as more technology becomes widely available, allowing people to experiment with alternative interfaces and game designs. The mobile gaming industry in particular stands poised to meet the needs of this underserved group, and some interesting forays into this realm are already being done. This study focuses on bringing 2-dimensional games to life for non-sighted players through a

paradigmatic application of a 3-dimensional audio environment. Two variations on this environment (referred to as “Flat” and “Curved”) are evaluated and combined with variations of two other factors, the control scheme and an “auto-lock” ability, to comprise a 3-factor study of eyes-free game mechanics. The primary hypothesis is that the “Flat” environment will objectively outperform the “Curved” environment in general, but that human opinion may vary with regard to immersion and enjoyableness. The other two factors (control scheme and auto-lock) will have impacts as well, while also serving as controls for the primary main effects (the environment).

Chapter 2. BACKGROUND

2.1 Literature Review

While still an underexplored field, the idea of accessible gaming for the vision-impaired has become an increasingly popular topic in the last ten years, so before designing this study, existing research in the field of audio gaming was done to determine how to best contribute to the existing body of work. This section discusses a selection of the most interesting and relevant results found, categorized into subsections based on observed similarities of approach.

2.1.1 *Modifications of Existing Games*

One approach has been toward modifying existing mainstream games to make them accessible to blind players. AudioQuake, a blind-accessible version of the PC game Quake, was demoed as early as 2004. This modified version used the idea of earcons: "Structured sounds, often obeying musical conventions, that are designed to alert the user to an object or event. They do not 'sound like' their referents." (Atkinson, Gucukoglu, Machin, & Lawrence, 2006). Essentially, different variations of short, distinct sounds are used to efficiently communicate different types of enemies, approaching corners, and other necessary information. This approach has the benefit of communicating a lot of information quickly, but comes with a learning curve, as users must learn to recognize the various earcons and accurately interpret their meaning. The project has since been turned over to the GitHub community, and is known as The AGRIP project: Accessible Gaming Rendering Independence Possible ("AGRIP AudioQuake and Level Description Language (LDL)," n.d., "The AGRIP Project," n.d.).

In 2008, researchers created Blind Hero, a modified version of Frets on Fire (an open-source game similar to Guitar Hero) that uses a custom haptic glove as a substitute for visual feedback.

The user wears the haptic glove while holding the standard guitar controller, and vibrations on each finger indicate which button needs to be pressed. They disabled the fifth button of the controller for simplicity and straightforward mapping to the four non-thumb fingers. One of the main problems they found with this interface was a lack of "lookahead" ability. In the visual version of the game, the player can see the note approaching for a longer amount of time before needing to play it. With this simple haptics interface, the player only finds out about the note at a single point: right before the button must be pressed. They explored the idea of adding additional haptics motors for lookahead ability, but deemed it too complicated and more difficult to learn (Yuan & Folmer, 2008). In 2009, the creators of Rock Vibe used an arduino processor and custom haptic arm and leg bands to make the drum interface of Rock Band accessible in a similar way (Allman, Dhillon, Landau, & Kurniawan, 2009). Naturally, the problems with lookahead in Blind Hero would apply to this interface as well, although they are not specifically mentioned. A more challenging hurdle to the approach of both of these games is the use of custom hardware. While the makers of Rock Vibe in particular focused on the use of cheap components, the custom nature of these interfaces and their high coupling to these specific games hinder both their availability and wide applicability.

In 2010, Morelli et al created VI Bowling and VI Tennis, blind-accessible games based on Wii Bowling and Wii Tennis. The focus on these games was in modifying the audio and tactile cues to provide more useful information to non-sighted players. The researchers separate the various visual, audio, and tactile cues into "primary" and "secondary" categories. Primary cues elicit the player to respond in a particular way. Secondary cues provide "reinforcement feedback," indicating whether the player's provided response was correct. In Wii Bowling and Tennis, the audio and haptic feedback tend to be used for secondary cues, whereas visual feedback is used for

primary cues. VI Bowling and Tennis make better use of the two non-visual categories by translating primary visual cues into auditory or tactile feedback, thus making both games more accessible to non-sighted players. VI Bowling explored a mechanic they call "tactile dowsing," where haptic feedback pulses increase in frequency as the player approaches the target (in this case, the ideal direction to throw the ball). While player enjoyment was not measured for VI Tennis, since their focus was on active energy expenditure, players of VI Bowling found it easy and fun to play (Morelli, Foley, Columna, Lieberman, & Folmer, 2010; Morelli, Foley, & Folmer, 2010). While these games, like Blind Hero and Rock Vibe, also use a relatively custom conglomeration of tools (the games were written for PC using Microsoft's XNA framework, and communicate with the Wiimote via Bluetooth), they do explore portable mechanics that could easily be implemented on the Wii itself or PlayStation Move without the need for additional custom hardware or extensive setup.

One group tackled the problem of modifying proprietary source code by developing a concept they call "Real Time Sensory Substitution." Using a separate PC, they analyzed the video feed from Kinect Sports running on an Xbox 360, and provided supplementary haptic feedback through a Wiimote (Morelli & Folmer, 2011). While this is a very unique approach, and does solve an interesting problem in trying to make commercial games accessible without access to the source code, it still suffers when it comes to wide applicability. It must be implemented and/or tuned for a given game to work, and many games simply don't have the kind of simple and predictable visuals that are required for the image processing to work reliably, quickly, or in some cases, at all.

2.1.2 *Inclusive Games*

Others have focused specifically on making "inclusive" games: games that are fun and accessible to both sighted and non-sighted players (Cheiran, Nedel, & Pimenta, 2011; Östblad, Engström, Brusk, Backlund, & Wilhelmsson, 2014). This approach focuses on trying to balance the difficulty of a game for sighted players and unsighted players. One group explored navigating a flat 3D environment of boxes, trying to locate the one containing the source of a sound. Although the results show that many players were confused by the interface and found it difficult to navigate, and it still represents a somewhat custom multi-platform solution, using a Wiimote and PC, they experimented with one particular mechanic that is worth mentioning: the idea of a "virtual cane." It is possible that the simplicity of the implementation may have caused some of the confusion. The virtual cane is simply "activated," either by pressing a button or swinging the Wiimote, and a combination of audio and haptic feedback is given if the user is within a certain range of a box. The angle of the Wiimote does not appear to be a factor, so it acts more like a radar than a real cane. Still, it is an interesting idea that warrants further research. Another group who focused on inclusive games explored the navigation of 2D point-and-click adventure games using a touch-screen and audio. Each interactive area of the point-and-click adventure emitted a particular sound using "3D" audio, and users used the direction of the audio to find the interactive object. In this case, 3D audio means simple stereo panning for left-right orientation, and volume for vertical proximity. 3D audio will be discussed more later, but one of the problems this group ran into was preventing the overall audio mix from becoming too messy to navigate.

2.1.3 *Original Content*

Still others have turned toward developing brand new games featuring either mostly audio, or even audio-only interfaces. This last category is the subject of this paper. Audio-only games have

existed for some time; Liam Erven, a blind programmer, has been making audio-only games for the PC for over ten years under the label "LWorks" (Liam Erven, n.d.-a). I had the opportunity to test two of his games. The first, Zompocalypse, was a demo of a work in progress, a "sidescrolling hack and slash." (Liam Erven, n.d.-b) It uses simple arrow key control for side to side movement and jumping. The sounds of zombies move from right to left as the player passes them. The second one, The Great Toy Robbery, is a simple game where you try to grab toys and avoid enemies. The arrows move the player around a discrete, 2D grid, with no rotation, and the audio is rendered from a first-person perspective. Toys and enemies make characteristic sounds that come from their corresponding direction. It is unclear what type of rendering algorithm was used for the 3D audio, but it seemed more advanced than simple left-right panning, as it was relatively easy to distinguish the difference between objects in front of and behind the player. The player always faces the same direction, avoiding the orientation issues that come with a rotatable perspective.

2.1.4 *Current iOS Games*

More recently, the advent of casual gaming on mobile devices and the growing prominence of indie game development has helped expand the market and push the desire for more original video game ideas to the forefront of the industry. This growth of mobile gaming and demand for innovation seems to have helped bolster the idea of audio games. The offerings are still meager, but I was able to find a small collection of games on the iOS App Store to test. These particular games were not selected on any criteria other than my ability to find them, and their self-identification as being audio-based and/or blind-accessible. Some were found through generic searches in the App Store, while others were found referenced in papers during the literature review. The table below summarizes my findings, and is followed by a more in-depth discussion of each title.

Table 1: iOS Game Testing Summary

Title	**Developer**	**Year**	**Type**	**Notable Features**	**Difficulties Encountered**
Papa Sangre	Somethin' Else	2010	360° Rotation & Movement	Tap left-right feet to walk	Voiceover interference, orientation difficulties
The Nightjar	Somethin' Else	2011	360° Rotation & Movement	Tap left-right feet to walk	Voiceover interference, orientation difficulties
Audio Defence: Zombie Arena	Somethin' Else	2014	360° Rotation	Gyro mode: very intuitive	None
Blindside	Epicycle	2012	360° Rotation & Movement	Can scrape along walls	Poor voice acting, orientation difficulties, buggy sound
Audio Archery	Liam Erven	2013	Timing-based arcade shooter	Swipe down to draw, release to fire	No directional feedback, gets boring fairly quickly
Audio Invaders	Stormy Studio	2012	Vertical scroller	Tilt to strafe: very intuitive	Inconsistent instructions
SpellStack	Ananse Productions	2011	Grid-based word puzzle	Entirely voiceover-based accessibility	Difficult/impossible with no vision
StemStumper	Ananse Productions	N/A	Grid-based connection puzzle	Entirely voiceover-based accessibility	Difficult/impossible with no vision
Sonic Tennis	Stefano Baldan	2013	Motion-sensing sports simulation	Uses device as virtual racket	No single-player, must type match name before starting

Developer “Somethin’ Else” managed to garner enough praise for their 2010 game “Papa Sangre” to warrant both a second title on the same engine (“The Nightjar,” featuring Benedict Cumberbatch), as well as a sequel to Papa Sangre (“Papa Sangre II,” featuring Sean Bean). Both Papa Sangre and The Nightjar exhibit a much higher production value than nearly every other game on this list. The player navigates a 3D environment by dragging across the top of the screen

to rotate, and tapping on the left and right portions of the bottom half of the screen to step forward. This explicit mapping of steps to taps, as well as the audio feedback of one's own footsteps, made it easy to grasp a sense of movement speed. In both games, they build an immersive narrative around a simple mechanic. The player starts at an entrance to a room, and has to navigate their way towards various sounds while avoiding others. Papa Sangre has the player navigating the "land of the dead," following musical notes and avoiding enemies. The Nightjar is a sci-fi story wherein the player is left behind on a spaceship and needs to escape. Despite their relative high production value, both are still difficult to play eyes-free, mainly due to two problems. First, the instructions emphasize that swiping the full length of the screen left to right will turn the player one-quarter of the way around, but findings indicate it is much more sensitive. This discrepancy often made orientation difficult. Second, the built-in Voiceover function on iOS is necessary to navigate the menus, but interferes with the controls when playing the actual game.

Somethin' Else's latest title is "Audio Defence: Zombie Arena." This game fixes the player in place while they defend against approaching zombies using various guns and melee weapons. There are three different control options for rotating: gyro, swipe, and tilt. In the gyro version, the player simply turns their body (and the device) in the direction they want the character to face, and the sounds "rotate" around the player as if they were static. When played in a swivel chair, the level of immersion is unprecedented, and this control scheme was the inspiration for the "Point" controls used in this study. This game also works well with Voiceover, which integrates perfectly with the menu system and stays out of the way during the game.

In "Blindside", the player wakes up unexpectedly blind in a post-apocalyptic scenario with strange dangerous creatures lurking throughout his house. This game takes on the challenge of full 3D navigation, just like the Papa Sangre games, but findings indicate it was not executed as well.

The voice acting is significantly more amateur, the tutorial relies on lengthy exposition, and the 3D audio was extremely buggy, with some sounds actually cutting in and out. The latter may be due to a lack of continuing support for new devices and version of iOS, but it cripples the presentation nonetheless. One mechanic worth mentioning is the ability to "scrape" along objects, which made navigation easier in an otherwise disorienting environment.

Liam Erven (the blind game programmer mentioned previously) was featured in a Polygon article (Moss, 2013) following his release of Audio Archery on iOS. In this simple arcade game, the player drags down on the screen to ready an arrow, and attempts to time the release to hit the center of a target that slides from left to right, as communicated through a scraping sound that pans from left to right. This game is simple, but fun and easy to pick up. Its only real fault is a lack of feedback on which side of the bullseye the player erred. It also requires that voiceover be disabled, but is clear about this up front, and implements its own fully-narrated menus that are easy to navigate. It doesn't hold interest very long, but it executes well on the intended concept.

Yet another simple entry is "Audio Invaders," a Space Invaders style game. Players can tilt the device left or right or tap the left and right sides of the screen to slide their ship across the bottom of the screen. Enemies approach from the top, and simple stereo panning is used to locate them. The player taps the screen to fire, and gets feedback when targets are being hit. The tilt controls are particularly immersive, encouraging the player to lean into the gameplay. Unfortunately, this game is better described as audio-mostly; while the game and its mechanics are accessible, the interface is not. It employs its own narration (in lieu of Voiceover), but is not consistent with it, often leaving the player confused.

SpellStack and StemStumper are two puzzle games that claim to be blind-accessible, but were found to be confusing at best. SpellStack is a grid-based word puzzle game that simply has added

Voiceover support. It seems to be nearly impossible to play blind, but perhaps the Voiceover features are intended for low-vision players rather than blind players. StemStumper is similar, in that it is a grid-based connection game with Voiceover support that appears similarly inaccessible with no vision, but again perhaps the Voiceover features are intended to assist low-vision players. It should be noted that, as the tester in this scenario, I have no visual impairments, and am simply playing with my eyes closed. It is possible that a player who was blind from birth or a young age would be more accustomed to overcoming the necessary barriers to playing games like these.

One last game worth mentioning is "Sonic Tennis," a unique game which leverages the unused display by using the entire device as a Wiimote-like motion controller in an audio-only two-player tennis game (Baldan, de Götzen, & Serafin, 2013). The ball approaches towards one side of the player or the other (indicated with stereo panning), and the player must execute a forehand or backhand swing accordingly. The game requires two players, and is difficult to get started, since it requires the player to name the match using the keyboard (one of the more difficult actions on iOS). However, once the game is launched, it offers simple, intuitive, and fun gameplay for two people.

2.2 Study History & Development

In light of the diverse existing body of work summarized above, it is difficult to determine the best place to start in order to make a relevant contribution. This section tells the story of how that determination was made, which starts by narrowing the focus to mobile.

2.2.1 *The Case for Mobile*

Mobile devices have some particular advantages when it comes to audio-only games, not the least of which is the proliferation of the technology itself. The ubiquity of mobile devices and the

increasing number of sensors they carry allows for the possibility of many alternative methods of interaction and feedback without the need for users to purchase custom hardware. In fact, most smartphones today carry most or all of the following hardware features:

Table 2: Typical Mobile Hardware Features

Feature	**Description**	**Application**
3-Axis Accelerometer	Measures movement	Motion-based gestures
3-Axis Gyroscope	Measures rotation	Gyroscopic aiming
Magnetometer	Measures compass direction	Real-world orientation
Vibration motor	Vibrates the device	Haptic feedback

In particular, Apple's iOS environment comes with a number of software features that makes it particularly friendly to both blind users and developers seeking to build blind-accessible games (see Table 3 below). The Voiceover accessibility feature allows blind users to navigate the OS, settings, and built-in apps. Additionally, third-party apps can support Voiceover through Apple's UI Accessibility protocol. In fact, Apple's UIKit (the primary way of building interfaces for third-party apps on iOS) implements this protocol by default ("Understanding Accessibility on iOS," n.d.). For games, Apple's new AVAudioEngine, released with iOS 8, allows for real-time 3D audio mixing and filtering, using a simple 3D coordinate system. HRTF is a particular rendering algorithm available in iOS that is discussed further in the next section. Additionally, at the time this research was begun in 2015, third-party developer Superpowered had found through their testing application that Android devices had significantly higher audio latency than iOS devices in general ("iOS and Android Audio Latency Test App," 2015). The extensive list of devices they have measured also illustrates the diversity of hardware that could make it difficult to achieve predictable performance across potential future widespread deployment, despite Android's tremendous market share.

Table 3: iOS Features Relevant to Blind-Accessible Game Design

Feature	**Description**	**Application**
Voiceover	Native screen reader	OS Navigation
UIAccessibility	Voiceover support protocol	Third-party app menu navigation
AVAudioEngine	Audio Engine API	Easy audio mixing and rendering
HRTF	Advanced 3D audio algorithm	High-resolution 3D audio

2.2.2 *Head-Related Transfer Functions*

Among the multiple audio rendering algorithms available in iOS 8 is one that merits some additional discussion, and makes this project possible: Head-Related Transfer Functions (HRTF). In general, transfer functions can be used to represent any linear time-invariant (LTI) system. In audio, transfer functions are a way of representing the frequency-dependent filtering effects of any given step in a signal chain. For example, a simple low-pass or high-pass filter could be represented as a transfer function. Humans determine the direction of a sound source using multiple sources of information, the two obvious ones being the overall loudness difference between the ears, and the slight time difference of the sound's arrival at the two different ears. While these two factors can serve to approximate left-right positioning, one can easily see that they provide no information for vertical positioning (along the sagittal plane). In order to achieve any vertical positioning, one must account for the differences in filtering imposed on sounds arriving from different angles along the sagittal plane. Doing the same for the horizontal plane will improve left-right positioning as well. By measuring the impulse response of a sound traveling from a specific direction in 3D space to two microphones placed at the eardrum positions of an anatomically-correct model of a human head, and performing a Fourier transform of the results, a pair of transfer functions can be generated for each direction of approach. The resulting set are known as head-related transfer functions (HRTFs), which can then be used to simulate surround sound in a binaural system. (Blauert, 1997)

The idea behind HRTF is a relatively old concept that is only recently becoming more widespread, due to the processing power required to implement it. Apple only added built-in support for HRTF as recently as iOS 8.0, and notes in its developer documentation that it is "a cpu [sic] intensive algorithm." ("AVAudio3DMixing Protocol Reference," n.d.) While battery life is a design factor that should be considered, audio games' lack of need for 3D graphics or any significant image processing whatsoever should help make up for the audio rendering demands.

2.2.3 *Problem Statement*

All of the above make mobile devices, and Apple's iOS platform in particular, ripe for the development of audio games. And while the sequential presentation of some of the previous examples (see section 2.1) may paint an optimistic picture of the state of the industry in terms of accessible gaming, it is important to keep in mind that these represent some of the highest-quality audio game offerings on iOS, and many of them still suffer from debilitating bugs that significantly increase the barrier to entry for unsighted players, despite their being the target audience. Moreover, even if these games were flawless, when compared to the hundreds of thousands of games available on the App Store, the audio-based medium remains severely underexplored. The handful of games breaking ground in this field have surely only begun to scratch the surface of mechanics for this genre. Even excluding mechanics, a majority of the audio games available leverage the human fear of darkness and wind up falling in or near the "horror" category. Setting aside whether or not this is exploitative, it certainly represents a very narrow view of the possibilities of the genre.

Still, despite the small library of current offerings, the wealth of *directions* to explore illustrated by the games in section 2.1 make it difficult to decide where to start. Based on the opportunities discussed in sections 2.2.1 and 2.2.2, the first restriction chosen was to focus on

mobile and to leverage the availability of HRTF on iOS, which prevents the need for custom hardware. Second, rather than adapting a specific game (see section 2.1.1) or crafting a particular unique experience (see section 2.1.3), of which there are several examples, we chose to pursue the study of mechanics in general that might be applicable to multiple types of games, new or existing. This was inspired in part by the spirit of inclusivity promoted by some of the games in section 2.1.2. Of course, pursuing mechanics that will apply to all games is not realistic, so the last restriction imposed was to focus on 2-dimensional games. The particular benefits of this restriction are explained in the next section, which gives an overview of the original attempt at this approach, followed by its transformation into this study.

2.2.4 *AudioScreen 1.0: The Idea*

This study hopes to contribute to the underserved market of blind-accessible gaming on mobile devices, but before it was a study, it started as a game project based on a new framework. The primary goal was (and remains) to use 3-dimensional audio to facilitate interaction with 2-dimensional games on iOS. This restriction to 2-dimensional games is a conscious limitation. Games that employ free-roam 3D navigation from the perspective of the player have unique interface requirements, as the player must be able to both move and rotate, and the developers at Somethin' Else are already making great strides in this area. The framework explored here focuses on a simpler "arcade-style" interface, where 3D audio and the various sensors of iOS devices are used to take the idea of 2D screen, and map it in front of the player with 3D sound. Elements placed at a location on the screen will sound as if they are coming from that direction on the virtual screen. By restricting the dimensionality of the game space, we can leverage the dimensionality of the audio space to make this particular segment of games accessible. Despite the arrival of 3D games decades ago, 2D games remain a significant segment of the market, particularly on mobile.

Potential applications of this framework include any 2D games with a primarily static screen, such as fixed shooter games, hidden object games, various puzzle games, tower defense games. Depending on its performance, it's also possible that future research could expand this paradigm to apply to 2D games with a moving screen, such as side-scrolling games, vertical-scrolling games, and games with top-down free movement.

The original idea was to curve the screen around the player, both horizontally and vertically, so that the directional cues from the 3D audio could be used to determine the player's location relative to targets on the screen. This approach is what this study now refers to as the "Curved" environment (see the Independent Variables section for details on this environment), and is a very natural approach to the problem. However, the developer found in early prototypes that the directional cues were not felt as strongly as expected, and orientation in the vertical dimension in particular seemed challenging.

2.2.5 *AudioScreen 2.0: The Study*

After considering several game modifications and additional mechanics in an attempt to patch over the shortcomings of the 3D audio, a change in the overall approach was deemed necessary, resulting in the design of a second audio mapping. This second mapping, which is referred to in this study as the "Flat" environment, takes advantage of the distance attenuation component of the 3D audio. However, it is possible that this approach could feel less natural than the "Curved" approach, since the movements of the player in the virtual audio world do not mimic those experienced in everyday life (see the Independent Variables section for details on this environment).

It is tempting but dangerous to decide arbitrarily that the new approach is "better" based on a single data point, especially since the overall performance of generic HRTF is likely to vary from

player to player, possibly unjustly maligning the "Curved" environment in the early prototypes. Thus, the project was converted to a study to accurately assess the differences between these approaches. Since the control scheme used is likely another contributing factor to usability, three variations are included as the second factor in the study, and one additional mechanic, "auto-lock," is included as a third factor, to see if the Curved environment can be salvaged through additional feedback. As a preliminary test of this paradigm, this study tackles the basic task of finding and selecting solitary static elements on the screen.

2.3 HYPOTHESES

In summary, this is a 2x3x2 factorial study in unsighted game mechanics. As mentioned above, our primary prediction is that the Flat environment will be more effective for finding and selecting targets than the Curved environment, but since there is no accounting for user preference, surveys will give insight into their qualitative aspects. The use of three control schemes will help prevent confounding their effects with that of the environment while also exploring their general effectiveness, and auto-lock will give each environment an opportunity to be improved through additional feedback. See the forthcoming Methods section for detailed discussion of all three factors and their values.

Chapter 3. METHODS

This study evaluates users' ability to find and select a sequence of static targets on a 2D screen while blindfolded, using a 3D audio environment. The player moves a **reticle** around the screen to aim at the **targets**, and taps on the screen to fire at the target. Targets are represented by a looped scratching sound, which the player is trying to find.

3.1 STUDY FACTORS (INDEPENDENT VARIABLES)

This is a three-factor study, consisting of two variations on the audio environment, three different control schemes, and the absence or presence of a mechanic called "auto-lock," resulting in a 2x3x2 factorial design. The table below lists the three independent factors and their values, which are explained in the subsequent sections.

Table 4: Three Study Factors and Their Values

Environment	Control Scheme	Auto-Lock
Curved Flat	Touch Tilt Point	On Off

3.1.1 *Environment*

The core of this paradigm is the mapping of the real 2D screen into the virtual 3D audio screen. Two ways of doing this are tested. The first method, called the **Curved** environment, wraps the screen around the player, by simply taking the x-y range of the real screen, and linearly mapping x and y from Cartesian coordinates to angles. In the case of the iPad, the aspect ratio of 4x3 is mapped to a range of 120°x90°. In other words, the horizontal range of the audio screen would extend ±60° to the left and right, and ±45° up and down, relative to directly in front of the player. **Targets** on the real screen are mapped onto the audio screen, while the **reticle** is mapped to the player's head orientation in the virtual audio world. The idea of "moving" the reticle on the screen

is then represented by a virtual rotation of the listener's head in the audio space. Moving left or right rotates their head in the virtual world left and right. Moving up and down rotates their virtual head up and down. Thus, when the reticle is directly over a target, it will sound as if it is directly in front of them. Virtually speaking, they will be "looking" directly at it. The **screen depth** (distance from the screen to the player) is 1 meter, which, due to the angular mapping of the environment, is the same for all points on the screen. See Figure 1 below for a representation of the Curved screen.

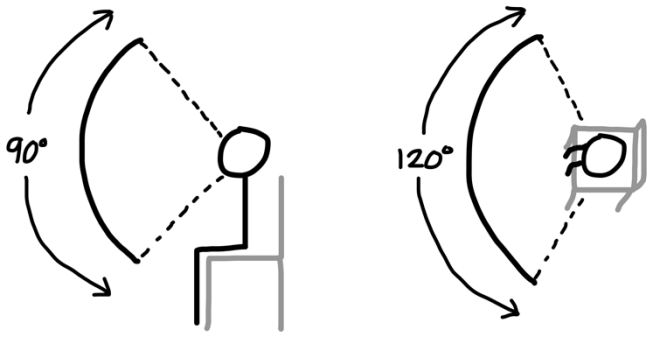

Figure 1: Curved Environment Mapping

This type of mapping has some natural limitations. The choice of 120ºx90º is not entirely arbitrary. First, the horizontal range is less than 180, which keeps the action primarily in front of the player, supporting the mental model that this 3D environment represents a 2D screen, and avoiding the problem of the left and right edges of the screen growing closer to each other as they wrap behind the player. This also restricts the necessary horizontal rotation required by the user when using the Point controls (discussed later). Second, the vertical range is *significantly* less than 180, which also keeps the action in front of the player, but more importantly, prevents the collision of x and y at the north and south poles. For example, if the range of both axes were exactly 180, the entire top edge of the screen would be mapped to a single point directly above the player, and the bottom edge would be mapped to a single point directly below the player. By truncating the axes in this way, the simple design language of a "curved screen" can be used without confronting

the challenges of a complete spherical coordinate system. The choice of 120ºx90º also allows for future mapping of 16x9 screens (another popular aspect ratio of mobile devices) to be easily mapped to 160ºx90º, while maintaining the desired restrictions mentioned above.

The second method is called the **Flat** environment. As the name suggests, the virtual screen is a flat screen in front of the player, and extends up, down, left, and right. One might naturally compare this to sitting in front of a regular TV, but there are two important distinctions here. The first is that this is an extremely large screen that is very close (30 meters tall, 40 meters wide, and 1 meter away). The second is that instead of reticle (player) movement being represented by virtual head rotation, it is represented by virtual movement in a plane parallel to the screen. In the virtual audio world, the user is always facing forward, but as they move the reticle, the virtual player moves up, down, left, and right in a plane parallel to the screen. As in the Curved environment, the screen is 1 meter away, but since the screen is flat, this is only true of one point on the screen at a time: the point of the screen that is currently directly in front of the player.

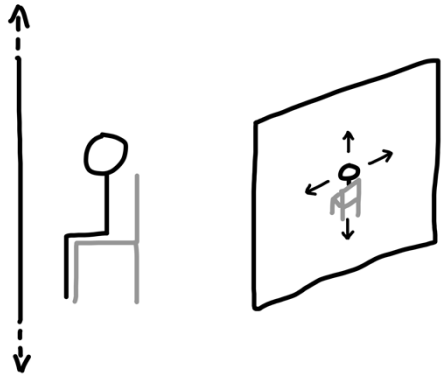

Figure 2: Flat Environment Mapping

This mapping is potentially less natural, but has the potential to be more accurate. Sound localization, the ability to detect the direction a sound is coming from, is difficult for humans in the vertical dimension, even in the real world. Furthermore, inevitable inaccuracies in the implementation of generic HRTFs, such as those used in Apple's AVAudioEngine, compounds

this deficiency. Informal preliminary testing shows that achieving sufficient vertical resolution using the Curved mapping will likely be difficult. The Flat mapping has the advantage of distance attenuation as an additional variable. Not only will the apparent direction of the sound change as the user moves around the screen, it will get louder as they move closer, since the distance decreases as a result of the Flat environment. Note that the size of the virtual screen is not inherently limited in the same way as the Curved environment. Theoretically, the Flat screen could extend to any size, but practically, it must be small enough so that a target on one side of the screen could still be easily heard from the other side of the screen without the target then being uncomfortably loud when directly in front of the player. This evaluation is necessarily somewhat subjective, but was determined through preliminary testing by the researcher. Users have the ability to adjust the overall volume before the start of the test and at any time to ensure it suits their hearing and preference.

Note that from the users' perspective, the environments are referred to as **Environment A** and **Environment B**, so as to avoid preconceived notions and more accurately assess their inherent intuitiveness. In all cases, **Environment A** is the **Flat** environment, and **Environment B** is the **Curved** environment.

3.1.2 *Control Scheme*

The second factor defines how the user moves themselves (the reticle) around the virtual screen. This study tests three methods of doing so. None of these methods are new, and all of them are likely have strengths and weaknesses for different types of games. Our goal is to evaluate them for the generic purpose of finding static targets on a 2D screen you can't see.

The most obvious control scheme is **Touch**. The user moves their finger around the screen to move the reticle (see Figure 3 below). The reticle follows their finger's absolute coordinates,

meaning that wherever they place their finger, that is where the reticle will be. Quickly tapping the screen anywhere will trigger a shot from the reticle's current location. The primary challenge with designing a touch control for this task is that it uses the same input (touch) for both movement and firing. Whenever the user taps the screen, the control needs to decide whether they are moving or firing. This determination is made using a 200ms delay timer and a 10pt tolerance radius. When the user's touch is first detected, it is ignored until either 200ms elapses or they move their finger more than 10pt. If either of these conditions are met, the target immediately snaps to their finger's current location, and tracks it until the finger is lifted. If the finger is lifted within 200ms without moving more than 10pt, it is treated as a tap, the reticle is not moved, and a shot is fired from the reticle's location. This allows the user to fine-tune their aim without then having to lift and tap in the precise location they just found. The choice of delay length and tolerance radius were determined based on preliminary testing on the part of the researcher and feedback from two other preliminary test volunteers.

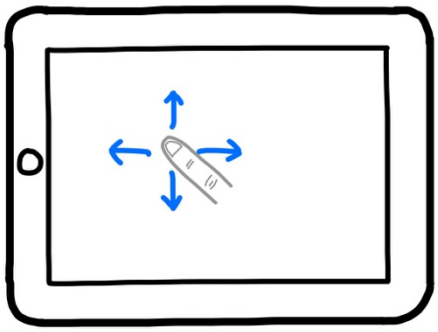

Figure 3: Touch Controls

The **Tilt** control scheme is similar to a flight yoke or joystick. The device is held in front of the user, with the screen facing them. Tilting the device left or right like a steering wheel will start moving the reticle left or right on the screen. Tilting the front of the device towards them will start

moving the reticle up, and tilting the front of the device away from them will start moving them down. (See Figure 4 below.) The angle of tilt sets the velocity of the player using a linear mapping. The sensitivity of the control is likely preferential based on the player, but must be fixed for the purposes of this test to prevent it from confounding the results. Originally, it was based on the default sensitivity of Audio Defence: Zombie Arena (see Table 1), one of the games tested in the literature review that employed Tilt controls, but was found to be too sensitive. After preliminary testing by the researcher and two other volunteers, the sensitivity was reduced and fixed at a rate of 2 degrees per second per degree, using the Curved environment as a reference. In other words, rotating the device by 2 degrees will start rotating the player by 2 degrees per second in the Curved environment. The sensitivity of the Curved environment mapping is then used to translate this into points (pt) on the screen for consistent sensitivity in both environments.

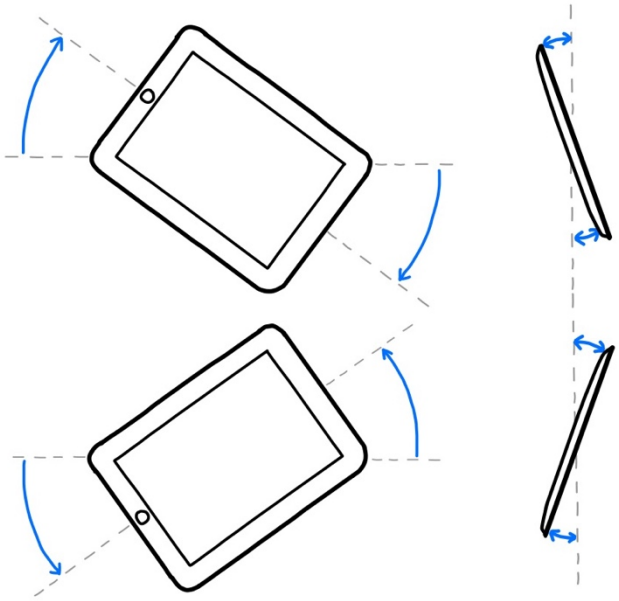

Figure 4: Tilt Controls

The last control scheme is **Point**. Similar to Tilt, the device is held in front of the user, with the screen facing them, but in this case, they simply point the back of the device to the place they

want to be on the virtual screen (see Figure 5 below), and the movement happens immediately in real time. This method adapts particularly naturally to the Curved environment, where the user's orientation in the real world maps directly to their orientation in the virtual audio world. For example, rotating 45º to the right and 30º up in the *real* world will rotate them 45º to the right and 30º up in the *virtual* world as well. However, it is possible that this control scheme may suffer in the Flat environment, where, for example, *rotating* themselves the left in the real world will actually *move* them to the left in the virtual world, while their virtual orientation remains fixed, facing the screen.

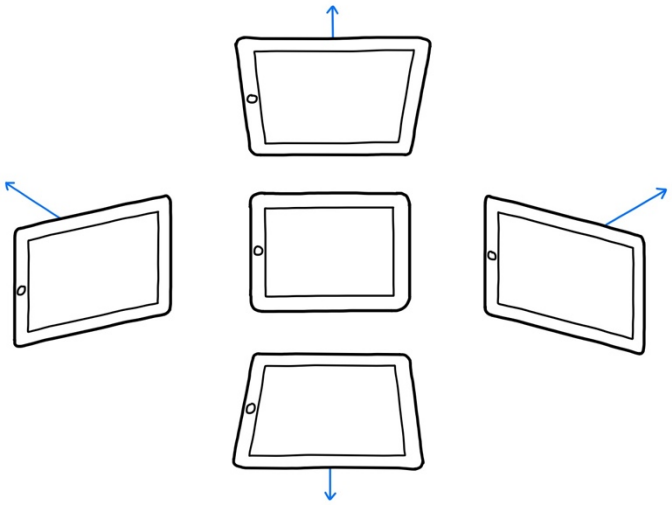

Figure 5: Point Controls

3.1.3 *Auto-Lock*

Lastly, this study explores the addition of one particularly helpful game mechanic for the task at hand: auto-lock. This feature, when enabled, will play a repeating beeping tone that gets faster as the user approaches the target, and goes solid (constant) when the user touches the target. The user is now "locked on" to the target's center and can proceed to fire at any point for a perfect shot. This mechanic separates the task of finding from that of selecting. We expect it to improve

objective performance, particularly in the Curved environment if the directional cues from the 3D audio are insufficient as predicted. However, it is unclear when the qualitative impact of this mechanic will be, which will be measured through user surveys.

3.2 Dependent Variables

Users attempt to find and select a sequence of 10 targets in all combinations of the 3 factors discussed above (2 environments, 3 controls, 2 auto-lock states), for a total of 12 tests and 120 targets. During each test, the device collects testing data for evaluating the performance of each combination. Users also complete surveys to collect demographic data as well as feedback on their experiences during the different tests.

3.2.1 *Test Data*

The device automatically collects two types of data during testing. First, since the testing order of the values of each factor are randomized for each participant, it collects data for each player about the order in which they were tested. The testing app writes one line for each player containing this information, as shown in the table below.

Table 5: Player Data

UserID	User ID of the player
Environment 1	The environment tested first
Environment 2	The environment tested second
Control 1	The control scheme tested first
Control 2	The control scheme tested second
Control 3	The control scheme tested third
AutoLock 1	The auto-lock state tested first
AutoLock 2	The auto-lock state tested second

Second, for every shot fired by any player, it collects data about that shot, as shown in the table below.

Table 6: Shot Data

UserID	User ID of player
Environment	The environment in which the shot was taken
Control	The control scheme with which the shot was taken
AutoLock	Whether auto-lock was in use for the shot taken
TargetNum	Number of target in test sequence
TestNum	Number of test in test sequence
Volume	System volume level
Hit	Whether or not the target was hit
Locked	Whether or not the reticle was locked on
StartTime	Time at which the target appeared
ShotTime	Time at which the shot was fired
DeltaTime	Seconds between target appearance and firing
TargetX, TargetY	Target position
StartX, StartY	Reticle position when target appeared
StartDPX, StartDPY	Relative position of reticle to target when target appeared, in points
StartDDX, StartDDY	Relative position of reticle to target when target appeared, in meters
StartDAX, StartDAY	Relative position of reticle to target when target appeared, in degrees
ShotX, ShotY	Reticle position when shot fired
ShotDPX, ShotDPY	Relative position of reticle to target when shot fired, in points
ShotDDX, ShotDDY	Relative position of reticle to target when shot fired, in meters
ShotDAX, ShotDAY	Relative position of reticle to target when shot fired, in degrees

This is a large number of variables, but many of them are to observe possible confounds, many of them are simply calculated from other fields for convenience in post-processing, and some of them ended up being unused (including all of the variables beginning with “Start” or containing “DD”). The primary constructs this study hopes to measure using the above data are relatively few:

- **Accuracy**: using shot location relative to target
- **Speed**: using time between target appearance and firing
- **Learning curve**: observing above two constructs over the course of each test

Accuracy and Speed were chosen as natural quantitative measurements of the effectiveness of the interface for the basic task at hand. For the paradigm to be useful at all in any future applications, users will need to be able to find targets with a reasonable level of accuracy, and in a reasonable amount of time. Thus, the variations explored here are qualitatively compared using these constructs. The learning curve is a natural addition to either, since the performance for a particular user is likely to improve over time.

3.2.2 *Survey Data*

Before, during, and after testing, users are asked to complete a total of four surveys:

- Pre-Test Survey
- Environment A Survey
- Environment B Survey
- Environment Comparison Survey

The pre-test survey collects basic demographics, while the other three surveys collect feedback on user opinions of the different configurations tested. Several questions on which user opinions might vary were brainstormed. They were then filtered, modified, and some added, based on feedback from preliminary testers. The resulting questions generally fell into one of three categories:

- Usability
- Enjoyableness
- Immersion

The following sections show the questions asked in each of the four surveys.

3.2.2.1 Pre-Test Survey

The **pre-test survey** collects basic demographics and information on how often the user plays video games. The questions from the pre-test survey are shown below.

What is your age?
- *18 to 24*
- *25 to 34*
- *35 to 44*
- *45 to 54*
- *65 to 74*
- *75 or older*

What is your gender?
- *Male*
- *Female*
- *Other, or prefer not to answer*

How would you describe your vision (with glasses or contacts if applicable)?
- *Not Impaired*
- *Partially Impaired*
- *Significantly Impaired*
- *Other (please specify)*

Do you consider yourself left-handed or right-handed?
- *Left-handed*
- *Right-handed*
- *Ambidextrous*
- *Other (please specify)*

How often do you…

Play video games of any kind?
- *Never – Almost Never – Sometimes – Fairly Often – Often*

Play video games using headphones?
- *Never – Almost Never – Sometimes – Fairly Often – Often*

Play games on a mobile device?
- *Never – Almost Never – Sometimes – Fairly Often – Often*

3.2.2.2 Environment A & B Surveys

The **Environment A** and **Environment B Surveys** are identical. One is given mid-test, and one is given post-test, depending on the order in which the player tested the environments. (See the section on Testing Process for details on the test flow.) There are 10 questions, but for each

question on the survey, users are asked to answer it for all three control schemes, with and without auto-lock. An example of the first question is shown below.

I found the interface _____.

*Using **touch** controls, with auto-lock OFF:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*
*Using **touch** controls, with auto-lock ON:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*
*Using **tilt** controls, with auto-lock OFF:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*
*Using **tilt** controls, with auto-lock ON:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*
*Using **point** controls, with auto-lock OFF:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*
*Using **point** controls, with auto-lock ON:*	*Intuitive*	*1 2 3 4 5 6 7*	*Confusing*

A condensed form of the other questions is shown below, without each configuration:

I found it _____ to control where I was aiming:
Easy 1 2 3 4 5 6 7 Difficult

I found it _____ to know where I was aiming:
Easy 1 2 3 4 5 6 7 Difficult

I found the interface _____ when using it:
Slow 1 2 3 4 5 6 7 Fast

I had enough time to familiarize myself with the interface before the test ended.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree

I was able to imagine where the sounds were coming from.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree

I felt a strong physical connection between the controls and the environment.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree

The experience felt _____.
2-Dimensional 1 2 3 4 5 6 7 3-Dimensional

I found the experience _____.
Not challenging enough 1 2 3 4 5 6 7 Too challenging

The experience was fun.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree

3.2.2.3 Environment Comparison Survey

The **environment comparison survey** asks most of the same questions as the Environment A and B surveys, but instead of answering each question about the controls and auto-lock, the users are asked to compare environments A and B. Therefore, questions relating primarily to the control schemes have been removed, dropping the number of scaled questions from 10 to 7. There is also one additional question asking users about their favorite control scheme in each environment. An example of the first question is shown below.

I found the environment _____.
*In **Environment A**:* *Intuitive 1 2 3 4 5 6 7 Confusing*
*In **Environment B**:* *Intuitive 1 2 3 4 5 6 7 Confusing*

A condensed form of the other questions is shown below, without each environment:

I found it _____ to know where I was aiming:
Easy 1 2 3 4 5 6 7 Difficult
I was able to imagine where the sounds were coming from:
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
I felt a strong physical connection between the controls and the environment.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
The experience felt _____.
2-Dimensional 1 2 3 4 5 6 7 3-Dimensional
I found the experience _____.
Not challenging enough 1 2 3 4 5 6 7 Too challenging
The experience was fun.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
My favorite control schemes for the two environments were:
*In **Environment A**:* *Touch – Tilt – Point*
*In **Environment B**:* *Touch – Tilt – Point*

Lastly, users are given the opportunity to provide general feedback, using the following three prompts:

What did you like about this experience?
What were some challenges you ran into?
If you had one wish to improve any part(s) of this experience, what would it be?

3.3 Implementation Details

This section documents miscellaneous details of the implementation whose variation could affect test performance, but which were kept static for simplification of the study.

3.3.1 *Target Size*

In order to provide hit and miss feedback, and in order to determine the range in which auto-lock locks on, the size of the targets must be decided. This study uses 44x44pt square targets. This is the minimum recommended tap target size according to Apple's own developer documentation ("iOS Human Interface Guidelines," 2015), and, presumably not coincidentally, is also the height

of the default navigation bar on iOS. This represents an aggressive but desirable goal: for non-sighted users to be able to find and select the same size targets that are expected of sighted users.

The environment used will affect the perceived angular size of these targets. In the Curved environment, points on the screen map directly to angles of orientation. Since the real screen height is 768pt, the virtual screen height is 90°, and the target size is 44pt, the effective angular spread of a target can be calculated as follows:

$$Angular\ size\ in\ \boldsymbol{curved}\ environment = \ 44pt \times \frac{90°}{768pt} = \mathbf{5.16°}$$

Equation 3-1

In the Flat environment, the real screen height is 768pt, the virtual screen height is 30 meters, and the target size is still 44 pt. This calculation is a bit more complicated, because the angular size of the target changes as the user moves around the screen (since the screen is flat). However, if we take the case where the reticle is centered on the target, and recall that the screen is 1 meter away, we can use trigonometry to calculate the angle to the edges of the target and double it:

$$Angular\ size\ in\ \boldsymbol{flat}\ environment = 2 \times \arctan\left(\frac{22pt \times \frac{30m}{768pt}}{1m}\right) = \mathbf{81.35°}$$

Equation 3-2

Note the huge discrepancy between these two numbers. This effect is in fact the primary difference this study seeks to demonstrate between the two environments. The Flat environment allows the target to appear much larger once the user nears it, while still providing directional cues when the target is far away. Since the dimensions of the Flat environment are limited only by the attenuation of the sound rather than the wraparound problem of the Curved environment, the screen itself can also be expanded to increase the perceived angular size until the user can just barely hear the sound at the edges of the screen.

3.3.2 *Distance attenuation model*

Since head-related transfer functions (HRTF) only model the direction of the sound, and not its distance, attenuation must be applied on top of HRTF. The attenuation model affects how quickly the sound is attenuated as it moves away from the listener, and is built into AVAudioEngine on iOS. There are three presets available, and for the purposes of this study, the default setting was interpreted to be a recommendation, and was the setting chosen. The default setting uses the Inverse Model, which calculates the gain based on a reference distance and roll-off factor as follows:

$$gain = \frac{referenceDistance}{referenceDistance + rolloffFactor \times (distance - referenceDistance)}$$

Equation 3-3

The default values for referenceDistance and rolloffFactor were also used, and are 1.0m and 1.0, respectively ("AVAudioEnvironmentDistanceAttenuationParameters," 2014).

3.3.3 *Sound sources*

There are five sounds used in this study, as summarized in the table below.

Table 7: Sound Sources

Target	A looped scratching sound used to represent the target
Auto-Lock	An 882Hz beep that speeds up based on target proximity
Hit	A "smack" sound played when a target is hit
Miss	A "whoosh" sound played when a target is missed
Bump	A "bump" sound played when the user hits the screen edge

The **target** sound was recorded using the built-in Voice Memos app and the built-in microphone on an iPhone 6, by scratching a carpet with a piece of cardboard. Recording was performed at the researcher's home under a wool blanket and with all power in the home turned off, in order to minimize background noise and room reflections. The microphone was placed as

close as possible without bumping the source. The resulting sound was equalized using Audacity to boost the bass and treble contents and provide a more broadband sound. This was done based on research as old as 1908 showing that more "complex" sounds are easier for humans to localize (Starch, 1908), presumably due to them being more affected by the directional changes in frequency response imposed by the listener's head-related transfer function.

The **auto-lock** sound was generated using Audacity as a 50ms pulse of a 882Hz sine wave, followed by varying amounts of silence. The 50ms pulse includes a 5ms fade in and a 5ms fade out. Silence is then added in increments of 25ms to generate the different beeping speeds. The fastest speed has 25ms of silence between beeps, while the slowest speed has 200ms of silence between beeps, for a total of 8 different looping samples. The appropriate sample is chosen based on the user's Cartesian distance from the target, which is separated into 8 bands, as shown in the table below.

Table 8: Auto-Lock Bands and their Loop Periods

Distance (pts)	**Beeping Period**
< 75	50ms pulse + 25ms silence = **75ms** loop
75 – 225	50ms pulse + 50ms silence = **100ms** loop
225 – 375	50ms pulse + 75ms silence = **125ms** loop
375 – 525	50ms pulse + 100ms silence = **150ms** loop
525 – 675	50ms pulse + 125ms silence = **175ms** loop
675 – 825	50ms pulse + 150ms silence = **200ms** loop
825 – 975	50ms pulse + 175ms silence = **225ms** loop
> 975	50ms pulse + 200ms silence = **250ms** loop

Each time the user crosses into a new band, the new sample is scheduled to interrupt at the next loop point. If the user crosses the target, the current loop is interrupted immediately and a continuous 882Hz tone is played until a shot is fired.

The **hit** sound was taken from a sound titled "kung fu punch.aif" submitted to freesound.org by user "nextmaking" under the Creative Commons Sampling Plus 1.0 license, and was trimmed to remove surrounding silence (*kung fu punch.aif*, 2009).

The **miss** sound was taken from a sound titled "SWOSH-01 44.1kHz.flac" submitted to freesound.org by user "qubodup" under the Creative Commons CC0 1.0 Universal (Public Domain Dedication) license, and was filtered for noise removal using Audacity (*SWOSH-01 44.1kHz.flac*, 2008).

The **bump** sound was taken from a sound titled "Thuds on Window" submitted to freesound.org by user "zerolagtime" under the Creative Commons Attribution 3.0 Unported license, and was trimmed to include only one thud (*Thuds on Window*, 2014).

3.4 Testing Equipment

3.4.1 *Software*

Testing takes place on an iOS app constructed for the purposes of this study. It is written in Swift, and uses several Apple-provided frameworks, primarily UIKit and SpriteKit for the graphical interface, and AVAudioEngine for the audio interface. AVAudioEngine allows for the placement and playback of sounds in 3D space using HRTF rendering.

The surveys are conducted using Survey Monkey through SFViewController, an Apple framework that allows web content to be shown using the iPad's built-in web browser (Safari) without leaving the app. Unique links for each survey are embedded in the app to drive users to the correct survey at each step.

3.4.2 *Hardware*

The iOS app is deployed on an **iPad** for testing purposes. iPads are generally cheaper than iPhones, and provide all the sensors needed for the controls implemented in this study, in addition to substantially increased screen space. The latter prevents a small screen size from hindering the

'Touch' control scheme, keeping the focus on the resolution limitations of the virtual audio environment, rather than the limitations of the screen itself.

The sound is delivered through **Sony MDR-7506** headphones. These headphones represent a substantial improvement over Apple's earbuds, without being out of reach for the typical consumer. These and similar Sony models have been widely used in industry for decades as studio monitors, due to their relatively flat frequency response for a reasonably fair price (generally less than $100 US). The over-ear design also allows for better comfort and more consistent placement than earbuds, since they don't have to fit a particular individual's ear canal. An over-ear design also provide some sound isolation, and allows them to be easily sanitized between uses with sanitizing wipes.

During testing, users blindfold themselves using a **sleep mask**. This is an inexpensive way of helping users keep their eyes closed during testing. Their low cost allows a sufficient number of masks to be purchased to avoid reuse.

3.5 TESTING ENVIRONMENT

Testing is conducted indoors in a chair at a desk or table, using the aforementioned hardware (see "Testing Equipment"). No visual or auditory barriers are enforced, as users are blindfolded during testing and the headphones provide some aural isolation. Since the quantification of background noise and the task of coupling that data to the testing process with respect to time would significantly increase the complexity of the testing process, no attempt is made to do so, nor are any quantitative requirements imposed on the background noise level of the environment. Qualitatively, all environments were sufficiently quiet to carry on a conversation at a typical volume, as determined by the researcher.

3.6 TESTING PROCESS

After reading and signing the informed consent form, and having had any questions answered, subjects are given a brief verbal introduction to the test that hits the following points:

- Brief overview of the testing process:
  - Pre-test survey
  - Several orientation screens
  - First half of tests (6)
  - Survey on first half
  - Second half of tests (6)
  - Survey on second half
  - Final (4^{th}) survey comparing first and second halves
- Read all in-app instructions carefully and ask any questions
- Be careful not to accidentally bump the volume control
- Try to average ~10 seconds per target if you want to finish in an hour
- Some tests may be difficult, which is okay

The iPads are verified to make sure they are connected to the WiFi (to access Survey Monkey), and have the volume set to a predetermined recommended volume (6 clicks down from maximum). The users then launch the AudioScreen app, which provides detailed instructions and the testing environment in which to test each of the twelve configurations: 2 environments x 3 control schemes x 2 auto-lock states (On/Off). The app links users to online surveys for preliminary demographic information as well as qualitative feedback. A diagram of the overall test flow is shown below, followed by a detailed explanation of each step in the following sections. Note that the order of environment A and B testing will be randomized. The diagram below is an example of A followed by B.

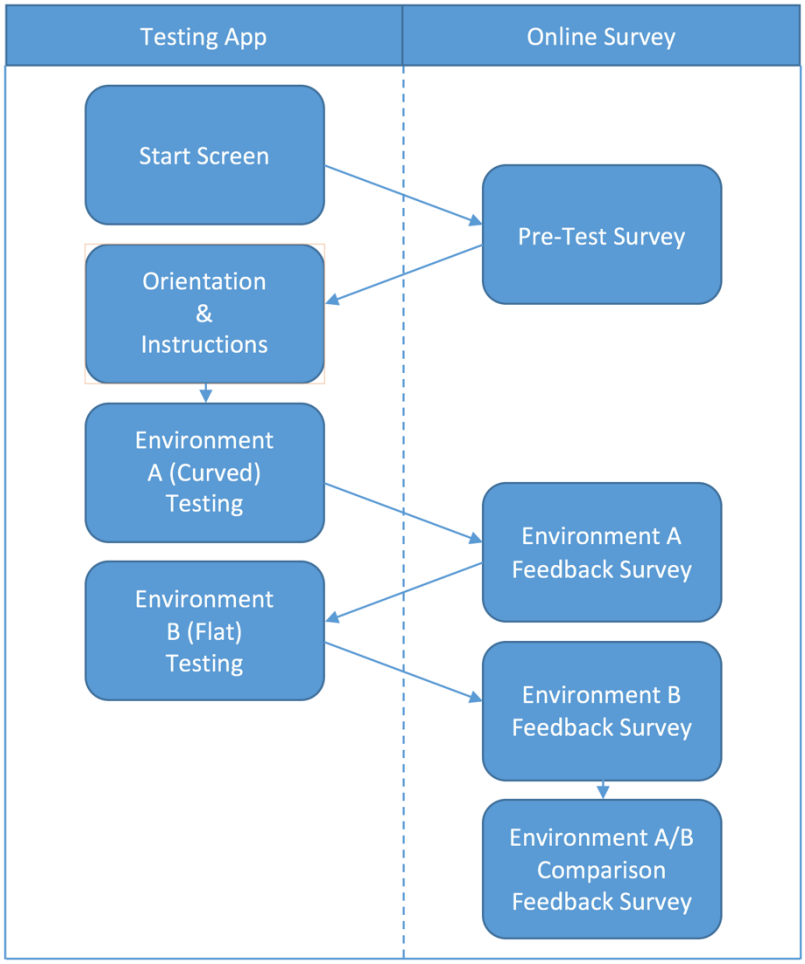

Figure 6: Test Flow

The researcher remains present during the testing process to observe, provide answers to questions, and intervene where necessary if users get stuck or misunderstand the controls. For example, if the user begins a test using one control type, but is moving as though they are testing another, the researcher intervenes to clarify. This was done in order to keep the study focused on the intuitiveness of the controls themselves, rather than the quality of their explanations or the user's ability to read and follow directions.

3.6.1 *Start Screen*

On launching the Audio Screen app, the app creates a new unique User ID for this tester, allowing us to link the online surveys with the app's collected data, without the need for identifying information. Users see the following message:

> *"Welcome to Audio Screen! Tap the Start button in the bottom right corner to get started."*

After pressing the button, the users see the following message:

> *"Before you begin, you'll need to take a quick survey.*
>
> *Tap the button below to open the survey in Safari. When you are finished, return to the Audio Screen app by tapping the 'Done' button on the top left in Safari, and then tap the 'Next' button below."*

The provided button launches the pre-test survey using Safari View Controller.

3.6.2 *Pre-Test Survey*

The pre-test survey collects some basic information about the user's vision, hearing, and possible familiarity with binaural audio. See the Dependent Variables section for survey details. After returning to the app and tapping 'Next', users will move on to the Orientation & Instructions phase.

3.6.3 *Orientation & Instructions – Volume Adjustment*

The next series of screens introduces testers to the Audio Screen concept and the tests they will be performing, but before doing so, it must be verified that all test subjects can hear the sounds they will be attempting to find, and that they're never uncomfortably loud. It is expected that the default volume will be sufficient for most users, but in the event that it is not, this step will allow them to adjust the volume. Users will see a screen with the following instructions:

> *"Put on the headphones. The cord should come from the left ear. You should hear a sound moving back and forth from left to right.*
> *If necessary, use the buttons on the right side of the bottom edge of the iPad to adjust the volume until you can comfortably hear the sound at all times.*

Try not to accidentally adjust the volume after this step, but if you find the sound level even mildly uncomfortable at any point, feel free to change it.

Tap 'Next' when you are done."

The quietest and loudest sounds the user could possibly encounter, respectively, will be when the target is near the edge of the screen in the Flat environment and when it is directly in front of them. The app renders a sound that moves back and forth along the horizontal axis (left to right) between these extremes. Since the system volume is recorded for each shot, the users are allowed to adjust the volume later if needed. Tapping 'Next' will take the user to the next screen.

3.6.4 *Orientation & Instructions - Controls*

Next, the user is introduced to the basic idea and the control schemes:

"You will be testing an audio environment that tries to represent the iPad's 2D screen using 3D audio. Imagine a large virtual screen in front of you, and extending above, below, and to the sides. Targets on the real screen will make a scratching sound coming from the corresponding location on the virtual screen. You can imagine them like little bugs you are trying to find and swat.

You will use three types of controls to move around, and you will test them with and without a feature called 'auto-lock'. Tap on the info buttons below to learn more about auto-lock and the different controls."

Along with the above message, the screen will introduce a table showing the three control schemes and the two states for auto-lock (On/Off). Users can tap on 4 info button to launch pop-ups explaining the different features.

Table 9: Control Scheme Introduction Screen

Control Scheme	**Auto-Lock (i)**
Touch **(i)** Tilt **(i)** Point **(i)**	On Off

Tapping on the info buttons marks will give the following explanations, with corresponding illustrations:

3.6.4.1 Touch

“With touch controls, you touch the iPad's screen to move around. Your finger's location on the screen represents your location on the virtual screen. Listen for the "bump" sound to find the edges of the screen.

Once you think you've found the target, lift your finger, then give the screen a quick tap anywhere to fire.”

See Figure 3 in section 3.1.2 for illustration.

3.6.4.2 Tilt

“Tilt controls are similar to a flight yoke or joystick. Hold the iPad in front of you with the screen facing you. Tilt the screen to the right or left to begin moving right or left. Tilt the top edge away from you to begin moving down, or tilt the top edge toward you to begin moving up. The farther you tilt, the faster you will move. Listen for the "bump" sound to find the edges of the screen.

Give the screen a quick tap anywhere to fire.”

See Figure 4 in section 3.1.2 for illustration.

3.6.4.3 Point

“Point controls use the iPad as a pointing device. Hold the iPad in front of you with the screen facing you, and point the back of the iPad toward the place on the virtual screen you want to be. Turn your head and shoulders with the iPad, like you are pointing a camera around. You may need to turn farther than you think. Listen for the "bump" sound to find the edges of the screen.

Give the screen a quick tap anywhere to fire.”

See Figure 5 in section 3.1.2 for illustration.

3.6.4.4 Auto-Lock

“When auto-lock is on, you will hear a repeating beeping tone. The beeping sound will get faster as you move closer to the target, and slower as you move farther away. When you find the target, the beeping tone will go solid, indicating you are ‘locked’ on the target and can’t miss. If you can’t find the target with auto-lock, get as close as you can and fire without locking.”

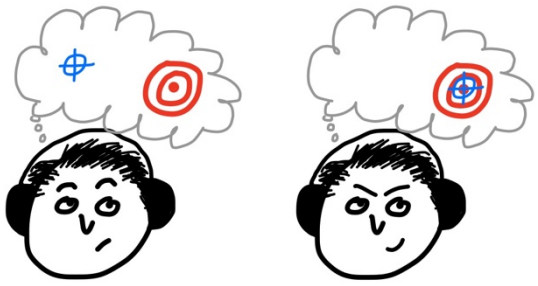

Figure 7: In-App Auto-Lock Illustration

3.6.5 *Orientations & Instructions – Environments*

After users explore the different controls and tap Next, the two environments are added to the table, with the following instructions:

> *"You will also test these different control schemes in two different audio environments. Tap the info buttons to learn more.\n\nWhen you are ready, tap 'Next' to find out what you'll be testing first."*

Table 10: Environment Introduction Screen

Environment (i)	**Control Scheme**	**Auto-Lock (i)**
A **(i)** B **(i)**	Touch **(i)** Tilt **(i)** Point **(i)**	On Off

Note that, from the user perspective, the environments are referred to simply as Environment A and Environment B. The mapping of the screen into the virtual space is a complex concept to explain, and the quality of that explanation may influence the user's experience. Thus, when they tap on the 'Environment' question mark, a more general explanation and illustration is given:

> *"The 'environment' changes how the virtual screen is shaped in the 3D audio space. Rather than explain the difference, we want you to tell us which is better, so simply think of them as 'Environment A' and 'Environment B'. Tap on the info buttons for each environment to get a feel for how they sound."*

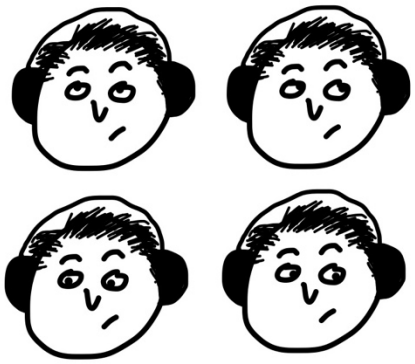

Figure 8: In-App Environments Illustration

When they tap on the question marks for A and B, it will launch a demo for the corresponding environment, wherein a sound starts playing in the center of the screen, and then moves to various other locations as shown in the pattern below.

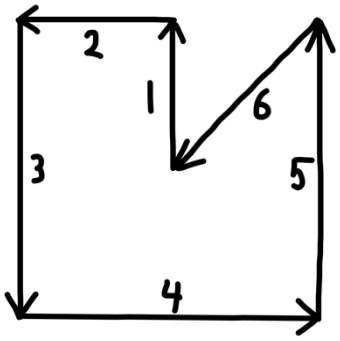

Figure 9: Environment Orientation Sound Path

Since the users will not be blindfolded at this point, they will be able to see the target and the reticle as they move around, familiarizing themselves to the 3D audio environment. The demo will also include the following text at the top of the screen:

> *"This is environment A* [or B, if appropriate]. *You are the blue square. The red square is the target. Listen to how the sound changes as you move up, down, left, and right around the target. Tap anywhere to dismiss."*

Once the user has finished exploring all the info buttons and taps 'Next', they will move onto the first round of testing.

3.6.6 *Environment A Testing*

The testing order of the two environments will be randomized for each user, to control for testing effect. In this example, we are assuming environment A is tested first. The order of the control schemes and whether or not auto-lock is enabled will also be randomized for each user, but that ordering will be consistent for that user for the duration of the test, and the hierarchy of variation will always be the same: environment, followed by control scheme, followed by auto-lock. For example, the three factors for a particular user might be randomized as follows:

Environment: A, B
Control Scheme: Tilt, Touch, Point
Auto-Lock: Off, On

For this user, the testing order would be ordered as follows.

Table 11: Example Configuration Testing Order

Environment	**Controls**	**Auto-Lock**
A	Tilt	On
A	Tilt	Off
A	Touch	On
A	Touch	Off
A	Point	On
A	Point	Off
Break for Environment A Survey		
B	Tilt	On
B	Tilt	Off
B	Touch	On
B	Touch	Off
B	Point	On
B	Point	Off

When the user taps ‘Next’ from the end of the last Orientation & Instructions screen, the table of configurations will remain, and the configuration to be tested first will be highlighted, as shown in the example below.

Table 12: Current Configuration Preview

Environment (i)	Control Scheme	Auto-Lock (i)
A (i) B (i)	Touch (i) Tilt (i) Point (i)	On Off

The user will also see instructional text explaining that testing is about to begin:

> *"You will now test the following highlighted configuration by trying to find and select 10 random targets.*
>
> *"If you hit the target, you will hear a 'smack' sound. If you miss, you will hear a 'swoosh' sound. No matter what, you will move on to the next target. If you run into the walls of the screen at any point, you will hear a 'bump' sound.*
>
> *"Many of these tests will be difficult, so don't worry if you're having trouble. You can tell us what you loved or hated in the surveys. Speed and accuracy are both measured, so if you get stuck trying to find a particular target, just get as close as you can and take your best shot!*
>
> *"Feel free to tap on the info buttons again if you need to remind yourself how something works. When you are ready, tap the 'Next' button to begin testing. Don't worry, there will be another screen before testing begins where you can take your time putting on the blindfold."*

When the user taps 'Next', they will see an intermediate screen reminding them which controls they are testing, how to hold the iPad, and instructing them to put on the blindfold. For Touch controls, the instructions read as follows, accompanied by the image in the figure below:

> *"You are about to test using touch controls. You can hold the iPad however is comfortable, but we recommend laying it flat on the table, as shown below.*
>
> *After the last target, the sound will stop, and you can remove the blindfold to see what the next test will be. When you are ready, put on the blindfold and tap anywhere on the screen to begin."*

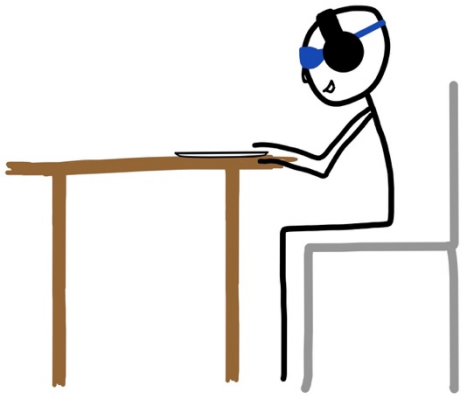

Figure 10: Test Prep Image for Touch Controls

For Tilt controls, the instructions read as follows, accompanied by the image in the figure below:

> *"You are about to test using tilt controls. Start the test with the iPad in front of you, and as close to perfectly vertical as possible, as shown below. It is common to tilt the top of the iPad away from you without knowing, causing you to drift downward without noticing.*
>
> *"After the last target, the sound will stop, and you can remove the blindfold to see what the next test will be. When you are ready, put on the blindfold, hold the iPad in front of you, and tap anywhere on the screen to begin.*

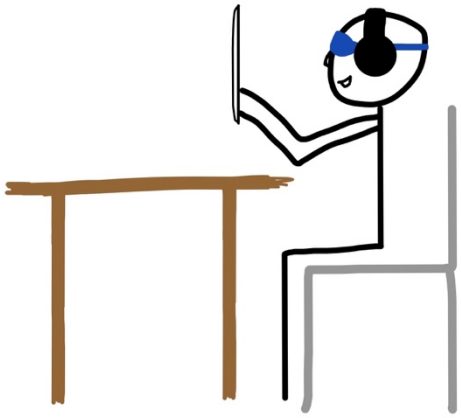

Figure 11: Test Prep Image for Tilt and Point Controls

For Point controls, the instructions read as follows, accompanied by the same image as Tilt controls (Figure 11 above):

> *"You are about to test using point controls. Start the test with the iPad in front of you as shown below.*
>
> *"After the last target, the sound will stop, and you can remove the blindfold to see what the next test will be. When you are ready, put on the blindfold, hold the iPad in front of you, and tap anywhere on the screen to begin."*

After the user puts on the blindfold and taps anywhere on the screen, testing begins for the current configuration. During testing, data about each shot is recorded (see Depended Variables section).

When they are done, they will return to the configuration table with a new configuration highlighted. This process will repeat until they have tested all 6 configurations for environment A. Once they have, the user will be presented with the following text:

> *"You're halfway there!*
>
> *"Before we move on to the other environment, we need you to take a survey about your experiences so far. Tap the button below to open the survey in Safari. When you are finished, return to the Audio Screen app by tapping the 'Done' button on the top left in Safari, and then tap the 'Done' button here."*

The provided button launches the Environment A (or B, depending on which environment was just tested) Feedback Survey in Safari View Controller.

3.6.7 *Environment A Feedback Survey*

The Environment A Feedback Survey asks the user to paste in their User ID, which is automatically copied to the clipboard for them by the app. It then asks several questions about the user's experience, with the intent to measure the perceived usability, immersion, and enjoyableness of each configuration. See the Dependent Variables section for details on the survey.

When the user completes the survey, they are prompted to return to the app and tap 'Next,' which will lead them on to test the other environment.

3.6.8 *Environment B Testing and Environment B Feedback Survey*

Environment B testing proceeds exactly as A, and eventually links to an identical (but separate) survey. When the second round of testing and survey is complete, the user is prompted to return to the app and tap 'Next,' which leads them to the last step.

3.6.9 *Environment A/B Comparison Feedback Survey*

The final step is a comparison survey between environment A & B. The app will give the following prompt:

> *"This is the last survey, we promise.*
>
> *"We'd like you to directly compare the two environments. Tap the button below to open the survey in Safari. When you are finished, return to the Audio Screen app by tapping the 'Done' button on the top left in Safari, and then tap the 'Done' button here."*

Tapping the provided button will launch the environment comparison survey. The environment comparison survey will ask several questions, similar to the environment A and B surveys, but this time asking the user to independently rate the two environments instead of the control schemes. See the Dependent Variables section for details on the survey.

Once the user returns to the app and taps 'Next,' they are greeted by the message, "You're Done! Thank you so much for your help!" and an animated GIF of fireworks, because we care. The animated GIF is sadly but necessarily absent from this paper.

3.7 Potential Threats to Validity

The following two sections provide a brief acknowledgement and analysis of the various internal and external threats to the validity of this study.

3.7.1 *Internal Threats*

The primary internal threats to this study are the potential **testing effects**. Users are likely to gain proficiency with the different levels of each factor over the course of the test, as all of them are encountered repeatedly. This threat is mitigated in two ways: randomization and measurement. As described in detail in the Testing Process section, the order in which the values of each factor are tested is randomized for each participant. Depending on the uniformity of the randomization, this should help control for this effect. The more uniform the randomization, the greater the effects will be controlled. Furthermore, the testing order of each factor is recorded for each participant, so that the effectiveness of the randomization can be measured. Additionally, the test number and target number are recorded for each shot taken, allowing accuracy changes to be analyzed in aggregate across the test progression, independent of the aforementioned randomization. This will allow for observation of the testing effect despite the randomization, to determine how necessary the randomization was.

Another likely factor to contribute to the effectiveness of any binaural 3D audio system is the **instrumentation**. Every point in the signal chain affects the frequency response of the whole system, and since the primary advantage of the HRTF approach is its modeling of the directional frequency filtering of the human head, its ability to do so will be affected by the frequency response of the rest of the system. In part, this is a known limitation of this study, since it already uses generic HRTF (rather than the measured transfer functions of each individual's head), but any *variation* of this effect can be controlled by keeping the instrumentation as consistent as possible. The D/A (digital-to-analog) converter, the headphone amplifier, and the headphone type, quality, and placement can all impact the frequency response. For this reason, identical iPad and headphone models were used (iPad 4 and Sony MDR-7506, respectively). The headphones were purchased

new for the purposes of this study, and the over-ear design allows for more consistent placement than in-ear or on-ear models.

3.7.2 *External Threats*

There are two primary unmitigated external threats to this study, which must be accepted as limitations. First, as mentioned above, it tests a **specific implementation** of **generic HRTF**. The library itself must necessarily be treated as a black box, since Apple offers no detailed information on the implementation behind the API. Did they use an existing set of HRTF measurements to create the library, or measure their own? If they measured their own, how was it done? How far do the measurements extend vertically (many libraries are limited in this dimension)? How many points did they measure? How do they interpolate between the measured points? Is the library tuned specifically for Apple EarPods, or for the best average performance across a variety of headphone models? How will other libraries perform under similar testing? This is a limitation we must accept, but it could be argued that none of these questions are as impactful as the fact that the library is by necessity generic. While it has been shown that humans using individualized HRTF can approach the localization performance of the real world (Bronkhorst, 1995), widespread implementation remains a challenge for obvious reasons, since measuring every user's head is impractical. Rather than utilizing the cutting-edge performance of a technology that is not realistically deployable (individual HRTF), this study seeks to show a unique method of leveraging the performance of its simpler and more widely available counterpart (generic HRTF).

Second, despite its title, this study uses **sighted participants**. The presumption that performance will be identical for blind individuals is a stretch. However, since the parts of the brain in blind people normally used to process vision are adapted to process other senses, it is conceivable that they could achieve similar performance (Balan, Moldoveanu, Moldoveanu, &

Dascalu, 2014). Due to the challenges of both designing and deploying a testing app for truly blind participants, the burden of doing so was deemed too high for this study, but hopefully, the preliminary results found here can be used as a low-cost first step to justify further research.

3.8 Data Post-Processing

This study generated six sources of data, summarized and described in Table 13 below. See the following sections for details on the post-processing of these files, and the Dependent Variables section for details on the exact metrics collected by each source.

Table 13: Data Sources

Name	**Source**	**Format**	**Description**
Player Data	Testing app	CSV	Order of factor values tested, for each tester
Shot Data	Testing app	CSV	Positions and timing for all targets and shots
Pre-Test Survey	SurveyMonkey	Excel	Demographics and video game experience
Environment A Survey	SurveyMonkey	Excel	Usability, enjoyableness, and immersion metrics for all controls, w/ & w/o auto-lock, for Environment A (Flat)
Environment B Survey	SurveyMonkey	Excel	Same as Environment A Survey, but for Environment B (Curved)
Environment Comparison Survey	SurveyMonkey	Excel	Subset of questions from Environment A/B Surveys, but comparing Environments A & B, plus favorite control for each environment and open-ended free-response questions.

3.8.1 *App Data*

Both data files generated by the app are in CSV format, but do not include a header. The headers were pasted in manually at the top of both files to provide variable names for each column. The data was then pulled into Tableau for analysis and visualization, and portions exported back to CSV format for statistical calculations in SPSS.

During analysis, one error was detected in the generated data. The ShotDAX, ShotDAY, ShotDDX, and ShotDDY variables (see Table 6) were actually duplicates of StartDAX, StartDAY, StartDDX, and StartDDY, respectively. The variables containing "DD" were not used in the analysis, but the variables containing "DA" (which show the angular error in the X and Y directions) provide an important illustration of how the Flat environment achieves its accuracy (see Figure 36). Thus, they were recalculated in Tableau from the "DP" variables, using the Tableau calculation language and the known sensitivities of the environments, with the following code snippet (which uses the X-direction variables, as an example):

```
IF [Environment] = "Curved"
THEN [Shot DPX] * [CurvedSensitivity]
ELSEIF [Environment] = "Flat"
THEN DEGREES(ATAN(([Shot DPX] * [FlatSensitivity]) / [ScreenDepth]))
ELSE NULL END
```

The above code uses the following defined parameters:

```
[CurvedSensitivity] = 90/768
[FlatSensitivity] = 30/768
[ScreenDepth] = 1.0
```

The above code yields the angle (horizontally, in this case) between the user's orientation and the actual direction of the target. The parameters were taken from the constants used in the code. The CurvedSensitivity takes the Curved screen's angular height in the audio world (90º) and divides it by its actual height in points (768pt). The Flat Sensitivity takes the screen's physical height in the audio world (30 meters) and divides it by the screen's actual height in points (768pt). The screen depth is in meters.

3.8.2 *Survey Data*

Each of the four surveys were exported to Excel in two ways: once with "actual responses," and again with numbers. The "actual responses" export includes the actual text of the answer

chosen, whereas the "numbers" export reports indexes or scale weights (where appropriate) instead. For example, the actual responses to most questions on the Environment A & B Surveys and Environment Comparison Survey may include text like "(Neutral)" or "Strongly Agree," if the columns were labeled as such, but the numbers export just includes the number for each answer (in this case, a scale from 1 to 7). Both versions for all four surveys were added as tabs into a single Excel spreadsheet, and a third tab was created for each survey, where each column was created by referencing the appropriate export format for the question ("actual responses" or "numbers"). The Environment A & B Surveys were also combined vertically into the same tab, and an "Environment" column added manually, with the appropriate environment label for the two surveys, "Flat" (Environment A) and "Curved" (Environment B).

In the SurveyMonkey export, each response gets its own column, and each respondent gets their own row. Unfortunately, in row-column matrix questions (like those used in the Environment A & B Surveys to ask each question for all three control schemes with and without auto-lock, and in the Environment Comparison Survey to ask each question for Environment A & B), SurveyMonkey exports each row of the question as a separate response. This should have been expected, but in order for this data to be analyzed using Tableau & SPSS, all the responses for each matrix question must be in the same column, and labeled with the appropriate factor values where appropriate (environment, control scheme, and auto-lock). This was done by carefully transforming each individual's responses from row to column, using Excel's transpose function. The results were then verified with the original by meticulously checking several random samples of questions. Then, the appropriate factor values were added as new columns and filled in according to the repeating pattern inherent in the survey design. This pattern was also verified several times.

After discovering that a few users managed to skip parts of questions, it was revealed that the "required question" settings on SurveyMonkey only required a response in one row of the matrix questions, allowing other rows to be skipped without warning if the user didn't notice. Conditional highlighting in Excel was used to identify these blank values, and they were manually replaced with the word "null," in order for them to be properly filtered as such by Tableau. This failure to require a response for each row resulted in 1 null value on the Pre-Test Survey, 8 null values across both Environment A & B Surveys, and 1 null value on the Post-Test Survey. Additionally, one user appears to have closed the Environment A Survey before saving the results, and another did the same for the Environment B Survey. Rather than deleting these incomplete responses, they were included in the export, resulting in two complete surveys full of null values (120 individual row responses). All null values were filtered from any analysis in which they were included.

Also, one user managed to fill out the same Environment B Survey twice, resulting in one full duplicate set of results. An additional column was added, called "RepeatedSurvey," with the labels "First" and "Second." The duplicate survey response that was filled out second (based on the survey start time) was labeled "Second," while all other responses were labeled "First." The figure below visualizes the differences between the two responses from this user. Each line represents a single question, where the two ends of the line represent the answer values for the two different times the user took the survey. Dots imply no difference between the two answers. The questions themselves are not shown, for brevity, but the varying size of the lines gives a general indication of how much a user's opinion could vary when sampled twice. However, it should be kept in mind that it is not clear what the user was thinking when taking the survey the second time. They may have been trying to recreate their original responses, or they may have just been confused.

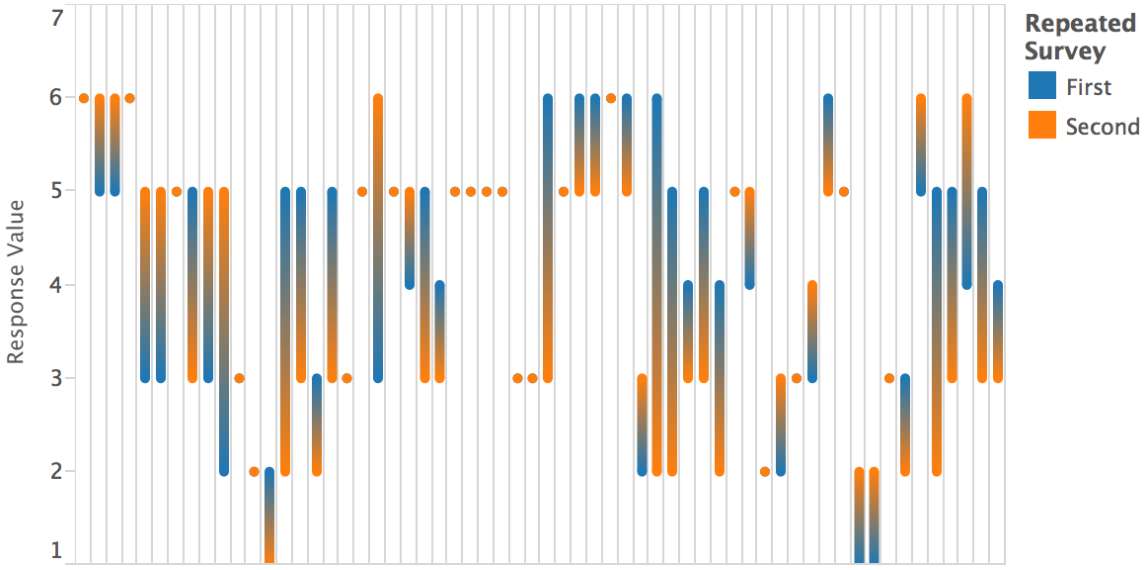

Figure 12: Differences in Duplicate Survey Response

The second response was discarded for all further analysis, under the presumption that their first response would have been most similar to the experience of other users, and not subject to any memory loss effects. As with the Test Data results, the data were imported into Tableau for visualization, from which excerpts were exported to CSV for statistical analysis in SPSS.

3.9 Test Subjects

After submitting an application for Human Subjects Review to the University of Washington Human Subjects Division and receiving a determination for exempt status (Category 7) for the study, test subjects were recruited from the UW Bothell CSS student population as well as acquaintances of the researcher. The only requirements were an age greater than 18, and hearing intact enough to hear the sounds. A breakdown of the participants by age and gender is shown in the figure below.

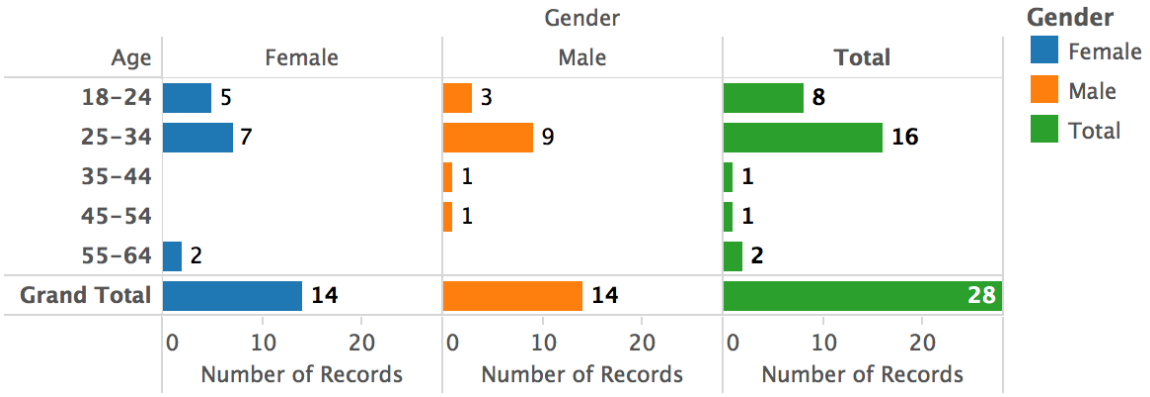

Figure 13: Age & Gender Demographics

A total of 28 subjects were tested, with a 50/50 split between female and male. Since the age survey question was categorical rather than numeric, we don't have a mean age, but the median clearly falls in the 25-34 category. This is not unexpected given the recruitment pools, although all categories up through 55-64 did have at least one representative.

Vision, hearing, and handedness (left or right) had less variation, as shown in the figure below summarizing these basic health demographics. The hearing and vision questions are unfortunately somewhat subjective. The participant who selected "significant" hearing loss did not appear to use hearing assistance of any kind, while the participant who selected "partial" hearing loss wore a hearing aid in one ear. The vision question specifically asks about vision with corrective lenses, so those who indicated "partial" vision loss must either have misunderstood the question, were nearsighted, or only had very slight vision loss, since no participants seemed to have any trouble reading instructions. It appears only one subject was left-handed, although one participant managed to skip this particular question (note the single "null" value).

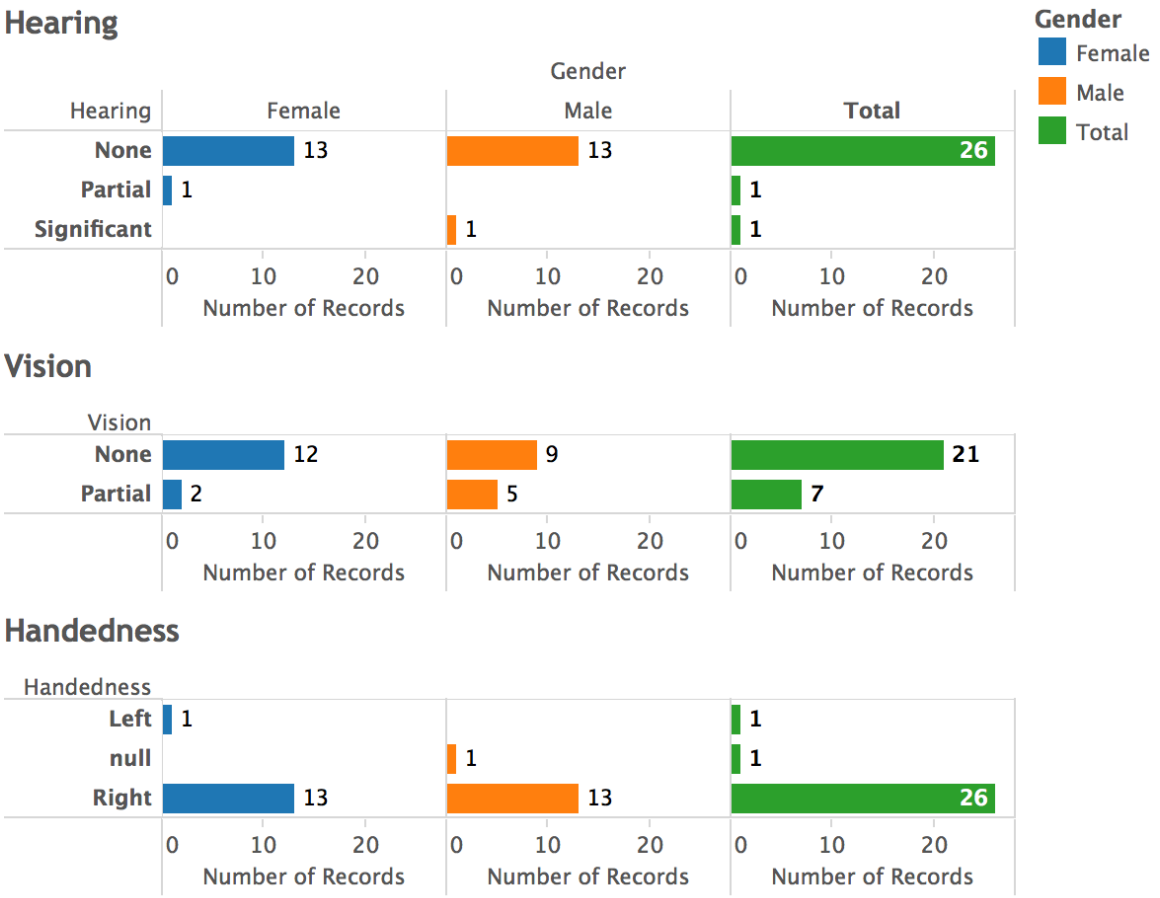

Figure 14: Health Demographics

Overall video game experience varied dramatically, with the median frequency of play clearly falling in the "sometimes" category. Frequency drops as expected when the "with headphones" or "on a mobile device" qualifications are added (since they represent subsets), although there are still representatives left in all categories of experience.

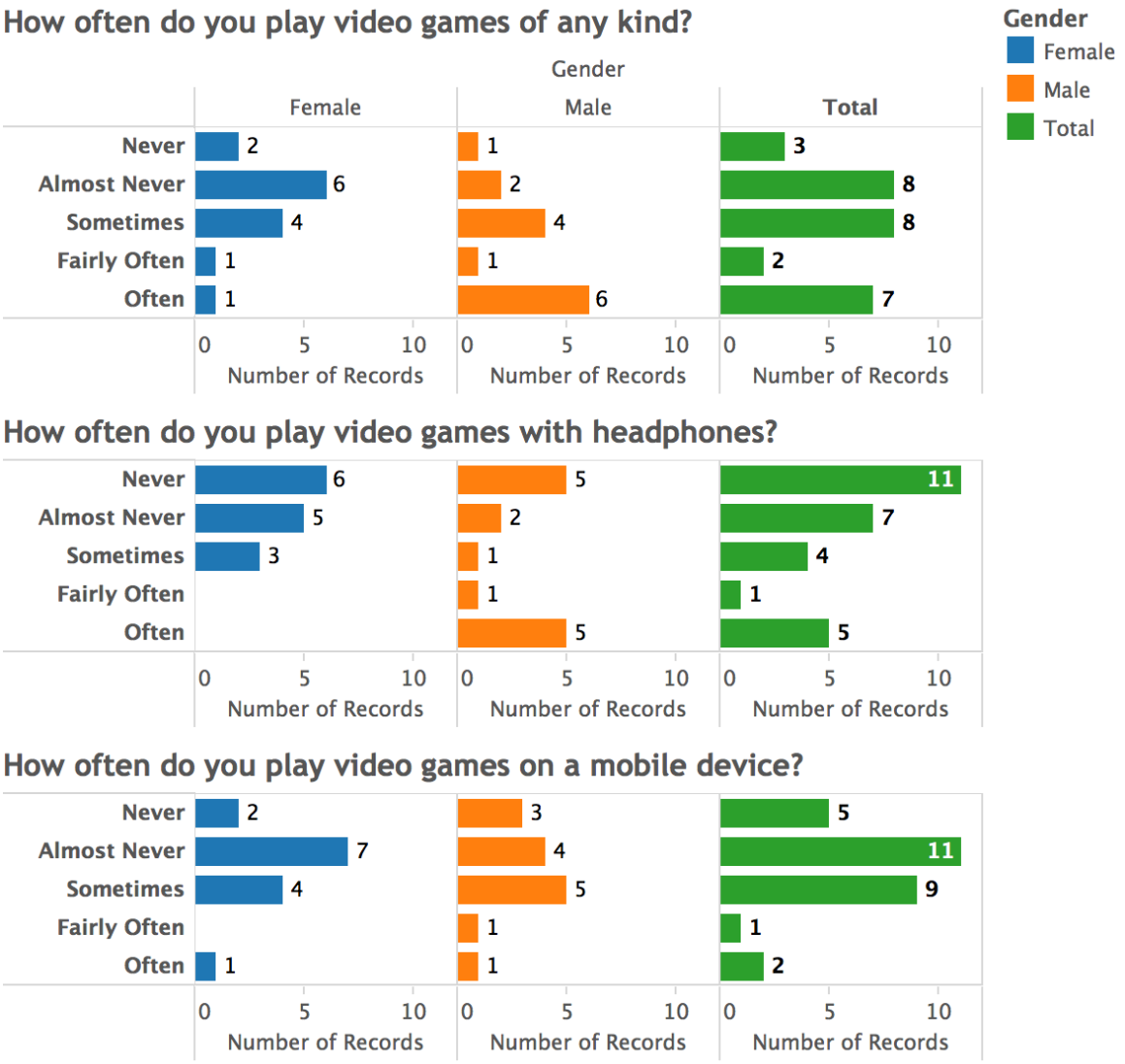

Figure 15: Video Game Experience

3.10 Uniformity of Randomization

This study uses randomization for two things: target placement and testing order. The figure below shows the full distribution of target locations across all 28 test subjects. Each subject performed 12 tests with 10 targets each, for a total of 3,360 targets.

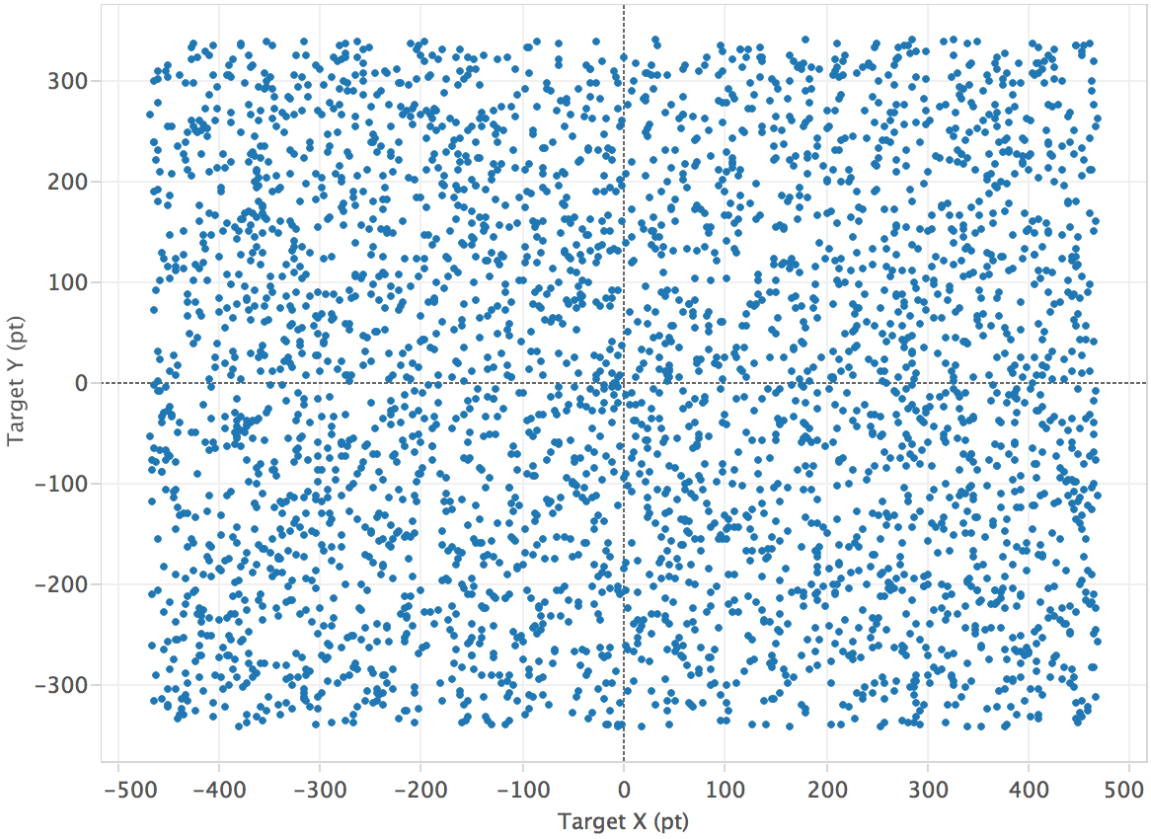

Figure 16: All Target Locations

The randomization function used is **arc4random_uniform**, which provides uniformly distributed pseudorandom numbers over the interval requested ("BSD Library Functions Manual: ARC4RANDOM(3)," 1997). Unfortunately, since the app is restarted between each session, true uniformity cannot be expected due to resetting of the seed, and in the first preliminary test (even without resetting of the seed), the randomization appeared to favor the lower-left quadrant. Nevertheless, as Figure 16 indicates, the larger sample size of the full study yielded a much more uniform distribution, with the 95% confidence intervals of both the mean and median for X and Y containing the central point (0,0).

The second use of randomization was the order in which the values of each factor were tested. This randomization controls for testing effects as participants become more familiar with the study,

the paradigm it uses, and 3D audio in general. Figure 17 summarizes how many participants tested a given value first (or second or third, in the case of control).

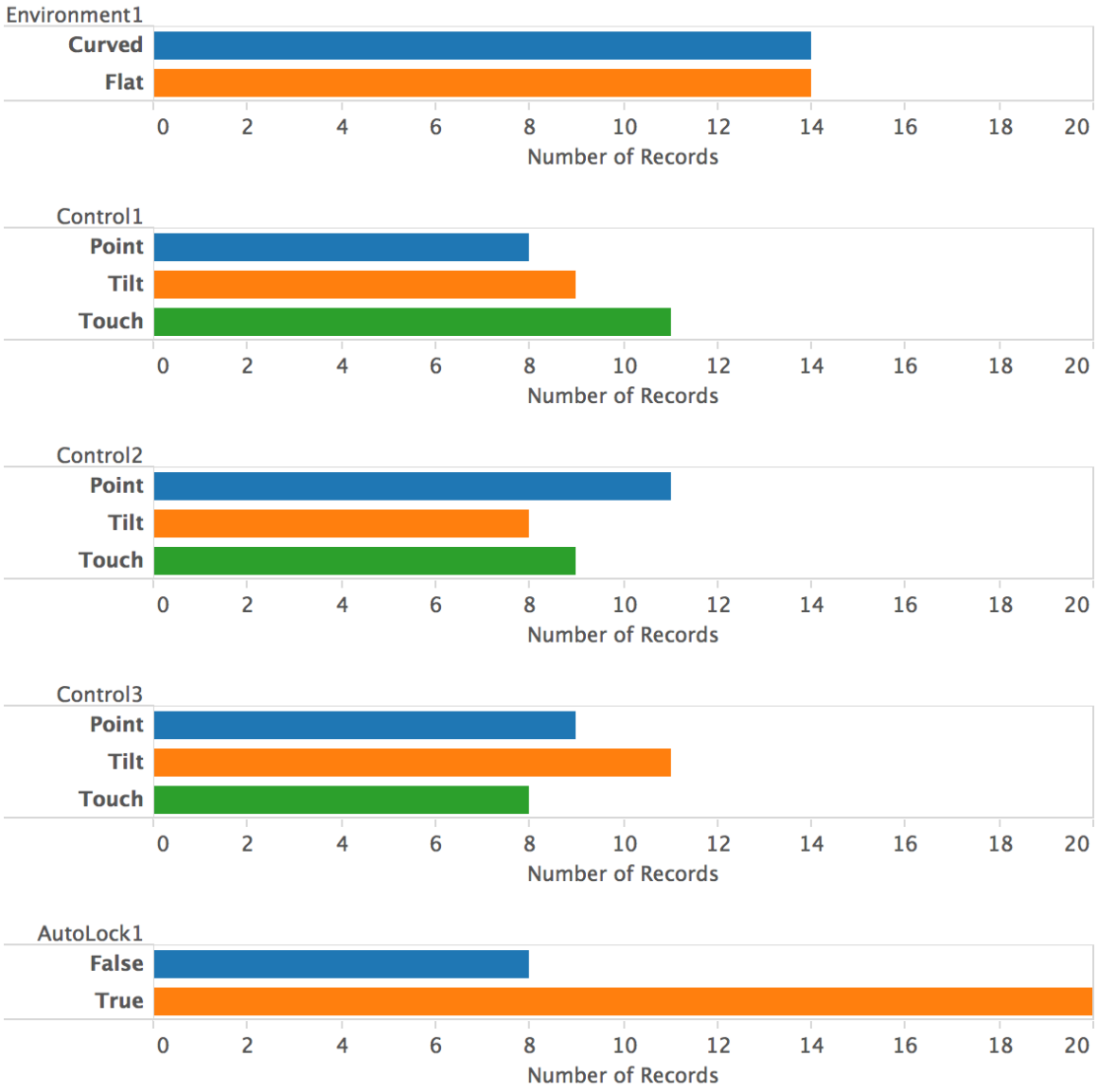

Figure 17: Test Order Randomization

The **Environment1** graph in Figure 17 shows how many participants tested a given environment first, and indicates a perfect 50/50 split between the two environments. This is excellent news, since the environment is the primary factor for this study. The **Control1**, **Control2**, and **Control3** graphs shows how many participants tested a given control scheme first,

second, and third, respectively. The order of this factor was fairly well distributed as well, given the number of values and participants. The **AutoLock1** graph shows how many subjects tested a given state of auto-lock first. This factor didn't fare as well, as the number of people who tested auto-lock *On* first is more than twice the number of people who tested it *Off* first. However, since the hierarchy of randomization is environment first, then control, then auto-lock, the randomization of auto-lock ordering should be the least critical of the three for controlling learning effects over the course of the whole testing process, as each participant will have experienced both states of auto-lock after the first two tests, regardless of the randomization order. Still, it is important to keep this anomaly in mind when considering the results.

Chapter 4. RESULTS & DISCUSSION

4.1 Data Representation

Several figures in the results section will use a box-and-whisker plot with a superimposed grey line representing the average value, as well as a grey band representing the confidence interval of the average. In order to keep the data representation compact, these elements are not explicitly labeled. The figure below serves as an example of this type of visualization, which in this case describes the distance from the target for all shots fired, separated by gender.

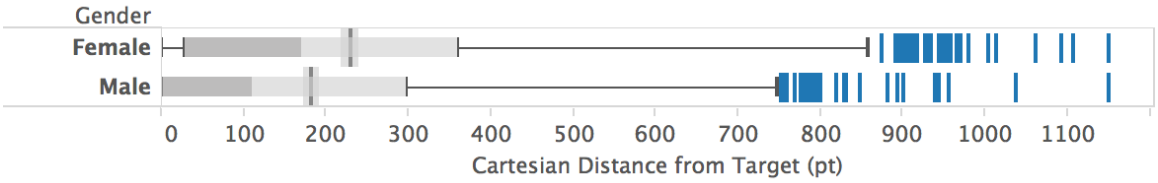

Figure 18: Example Visualization: Distance from Target by Gender

Note the box-and-whisker plot representing the quartiles. The whiskers extend to 1.5 times the inter-quartile range. All data points within the boxplot are hidden, while the blue marks to the right represent the outliers (outside of the whiskers). The vertical grey line shown behind the boxplot for both categories (male and female) represents the average for that category, while the small grey band surrounding it represents the 95% confidence interval for the average. All confidence intervals are 95% and based on the t-test, except where otherwise noted.

4.2 Demographic and Testing Effects

Before looking at the intended effects of this study, it is important to evaluate a few other possible factors, so that they can be kept in mind when considering the rest of the results. This section explores some of those factors and attempts to quantify their significance.

4.2.1 *Experience Level*

The figure below visualizes the statistics for the Cartesian distance from the target for all shots fired, grouped by the experience level of the user, based on the question, "How often do you play video games of any kind?"

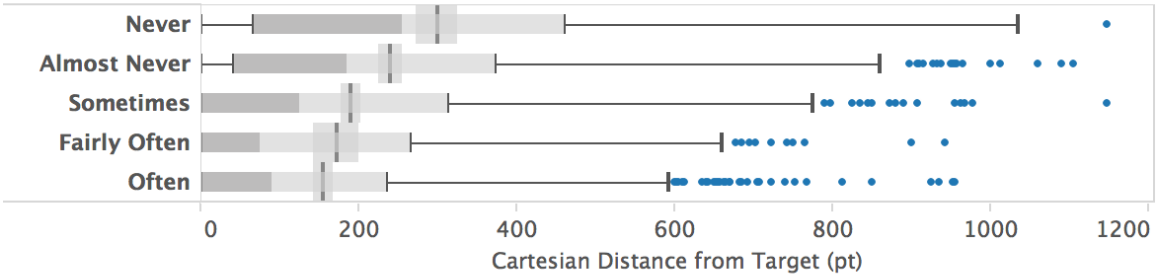

Figure 19: Distance from Target vs. Frequency of Gaming[1]

Note that the average distance from the target appears to decrease with increasing gaming frequency, indicating a possible correlation between the frequency with which the tester normally plays video games and the accuracy achieved in this test. In order to more formally investigate this effect, the game frequency was transformed into a numerical value (1-5), and used to calculate the **Spearman's rank-order coefficient** for the correlation between gaming frequency and distance from the target. SPSS gives a **Spearman's rho** value of **-0.206** for the correlation from gaming frequency to distance, with a significance of **0.000**. This strong statistical significance is a trend seen throughout this study due to the large number of points collected (3,360 targets in total), so it is important to look at the coefficient itself, and not just the statistical significance. In the case of Spearman's rho, a value of +1 or -1 indicates a perfectly monotone relationship between the variables, while a value of zero indicates no correlation. The value in this case, -0.206, shows a

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

relatively weak correlation, indicating that distance from the target does tend to decrease slightly as the user's gaming frequency increases, although not uniformly.

The pre-test survey also asked two other questions on game frequency, adding the qualifiers "on a mobile device" and "with headphones." The box-plots and average values for these are shown below. Based on the averages and their confidence intervals, the pattern seen in overall gaming frequency does not stand out here, so the rank-order coefficients were not investigated.

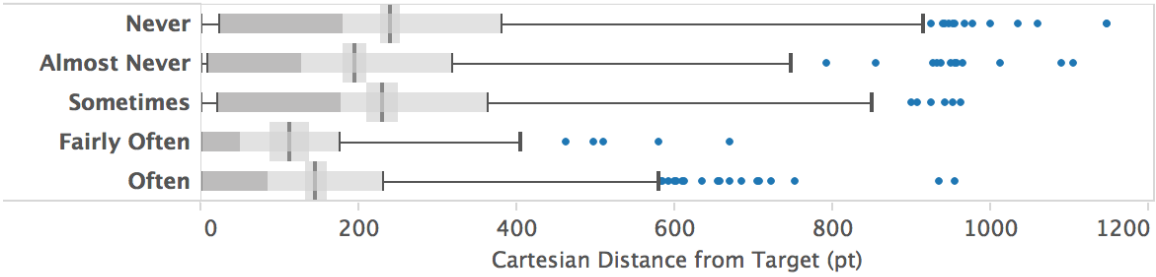

Figure 20: Distance from Target vs. Frequency of Gaming with Headphones[1]

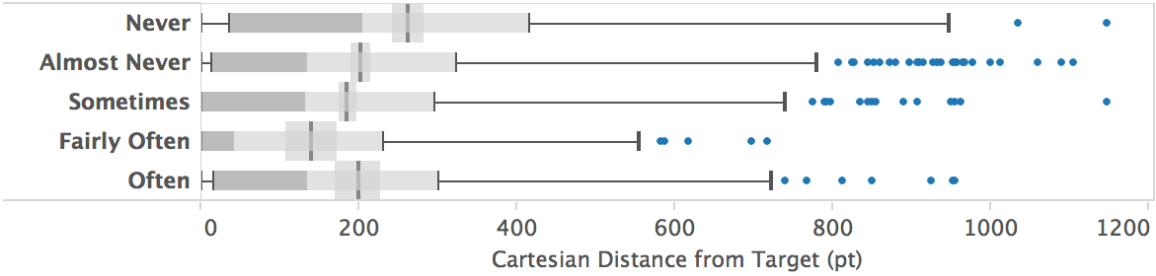

Figure 21: Distance from Target vs. Frequency of Gaming on a Mobile Device[2]

4.2.2 *Testing Effects*

As mentioned in the Methods section, the order in which the difference levels of each factor were tested was randomized for each player to control for testing effects, since it's possible that increasing familiarity with the general framework could cause players to improve over the course of the 12 tests, independent of the current combination of factor values being tested. Although the

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.
[2] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

randomization should help control for this effect, it can still be observed by considering the test number (1-12) irrespective of the configuration being tested. In order to do so, the test number was recorded for every shot fired. The figure below shows the box plots and averages for distance from the target, grouped by the index of the current test within the sequence (1-12).

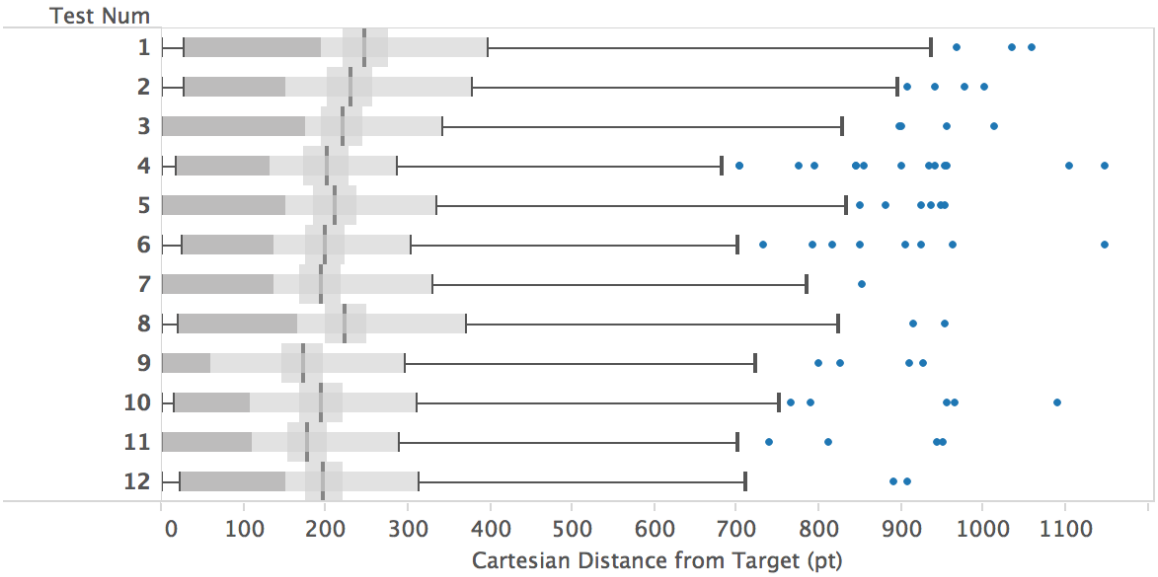

Figure 22: Distance from Target vs. Test Number[1]

The average appears to have a slight trend downward as the test number increases, indicating a possible learning effect, although not a particularly large one compared to the 95% confidence intervals. In order to more closely observe this possible effect, a "global target number" was calculated based on the test number (1-12) and the target number within the test (1-10), to give the target's presentation order within the complete sequence of all targets (1-120). This allows for the creation of the scatterplot shown below.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

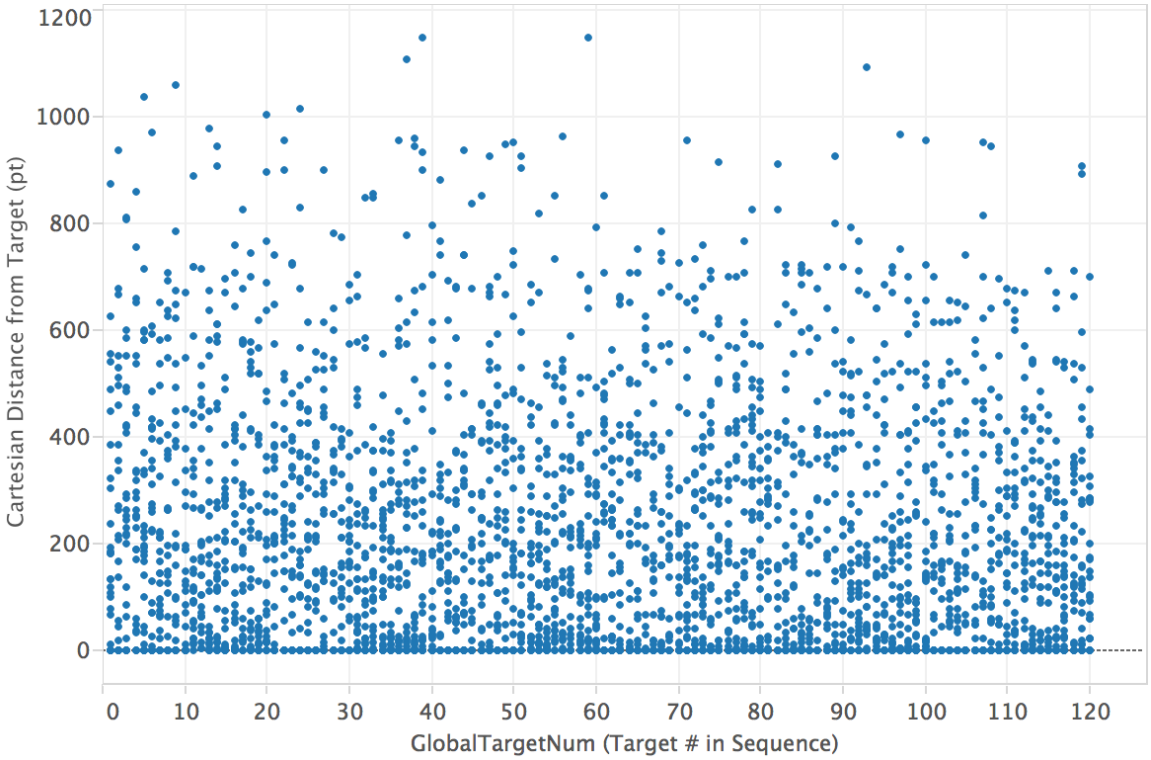

Figure 23: Global Target Number vs. Distance from Target

Using the complete set of all targets, and their global target number, we can quantify the learning effect using **Spearman's rho**, which is **-0.069**, with a 2-tailed significance of **0.000**. Based solely on the sign, this indicates the expected learning effect, namely, that distance goes down over the course of the test, regardless of testing order. Based on the magnitude, this effect appears to be extremely weak. However, the randomization performance must be kept in mind when considering this. Based on Figure 17, the environment and control scheme randomization was fairly uniform, but auto-lock was not. Over twice as many people tested auto-lock On first than tested it Off first. The effects of this can be seen by color-coding Figure 23 based on the number of records at the given distance and target number, resulting in the figure below.

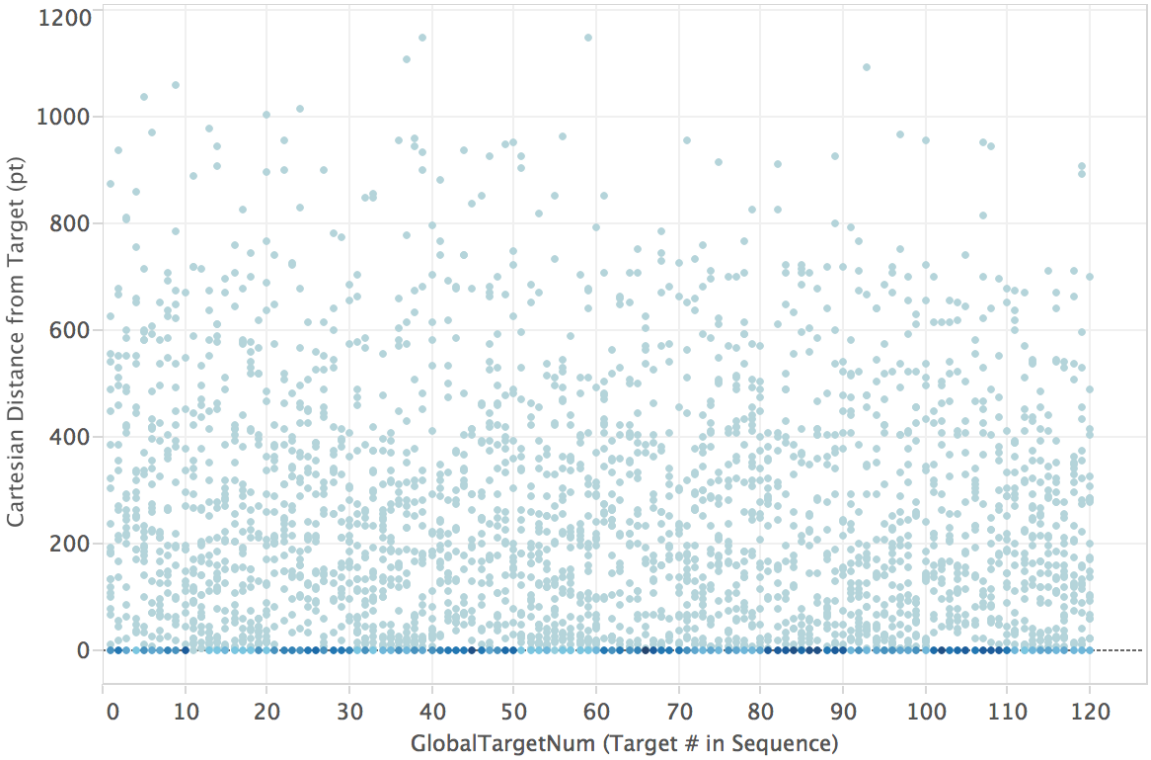

Figure 24: Target Number vs. Distance, Color-Coded by Number of Records

In Figure 24, the darker points indicate a higher number of records at the given distance and target number. Note the alternating pattern every 10 targets along the horizontal axis, and recall that more people tested auto-lock On first than tested it Off first. Auto-lock snaps the reticle directly to (0, 0) when it crosses the target, resulting in these repeated values at a distance of 0 from the target. If auto-lock decreases the average distance when enabled (as we will see in a later section), then this non-uniformity of randomization (which favored testing auto-lock On first) could be reducing the observed learning effect measured above. The amount to which it's doing so is difficult to quantify, but one approach is to separate the data into two groups, based on whether they tested auto-lock On first or Off first, and recalculate the correlation. The Spearman's rho values for these two groups were calculated to be **-0.184** (when auto-lock was **Off first**) and **-0.022** (when auto-lock was **On first**), both with a 2-tailed significance of 0.000. As expected,

testing auto-lock On first decreases the correlation. The true correlation value is likely somewhere between these two values, indicating a very weak learning effect.

4.2.3 *Learning Curve*

One of the secondary effects that this study originally hoped to measure was the effect of the various factor values on the learning curve of the interface. In order for this analysis to be possible, we would have to see a general improvement from the beginning to the end of each test. We can check for this by simply grouping all shots based on the target number (1-10) within the test in which it was presented, and looking for a general improvement. This is illustrated by the figure below.

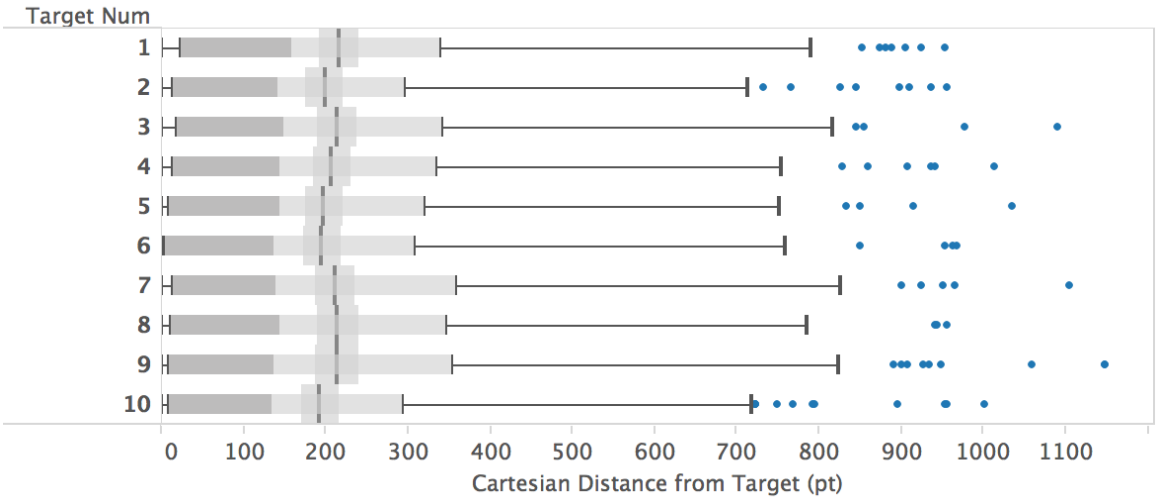

Figure 25: Distance from Target vs. Target Number Within Test (1-10)[1]

Unfortunately, the massive overlap of the averages and their 95% confidence intervals show that there was no clear improvement in distance over the course of the average test. This indicates that the tests were simply not long enough to see significant improvement, and so no attempt to

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

compare variations in the long-term learning curves of each configuration is made here. Instead, this study will have to be limited to measuring users' first experiences.

Ultimately, the number of targets tested (10 in each of the 12 configurations, for a total of 120 per user) was limited by the testing time of the study. Most tests took about an hour, and it is clear from the figure above that the number of targets tested per configuration would have to be significantly increased to observe any learning curve, which, when multiplied by the 12 configurations, would likely have imposed too high of a time burden on the subjects to recruit enough participants.

4.2.4 *Order Effects*

As mentioned in the Methods section, randomization of the testing order of the values of each factor was employed to control for testing effects, and with the exception of auto-lock, the randomization provided a fairly uniform distribution between the testing orders. Also, section 4.2.2 observed the testing effects independent of the testing order and found them to be extremely weak.

Another way to look at the effects of this randomization is to look for differences in performance based on the order of values tested for a given effect. For example, the figure below visualizes the quartiles and average for every shot's distance from the target, separated by the environment in which they were presented, and the order in which the user tested the environments.

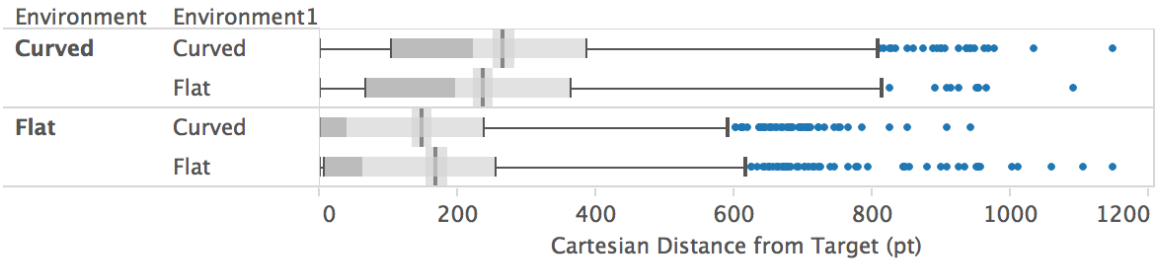

Figure 26: Distance from Target vs. Environment & Environment Order[1]

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

The "Environment" indicates the environment in which the shot was taken, while "Environment1" indicates which environment was tested first by the user. Ignoring the main effects of the environment for now (which will be discussed later), it appears that for each environment, the average distance from the target in that environment was very slightly decreased if they tested the *other* environment first, which is expected based on the weak learning effects illustrated in section 4.2.2. However, although it's difficult to see, the 95% confidence intervals overlap, indicating that neither of these effects are significant.

The figure below shows the order effects for the control type. As with the environment figure, the "Control" indicates the control scheme with which the shot was taken, while "Control1" indicates which control scheme was tested first by the user. In this case, **99%** confidence intervals are used instead of 95% to account for multiple comparisons, since there are three values for this factor.

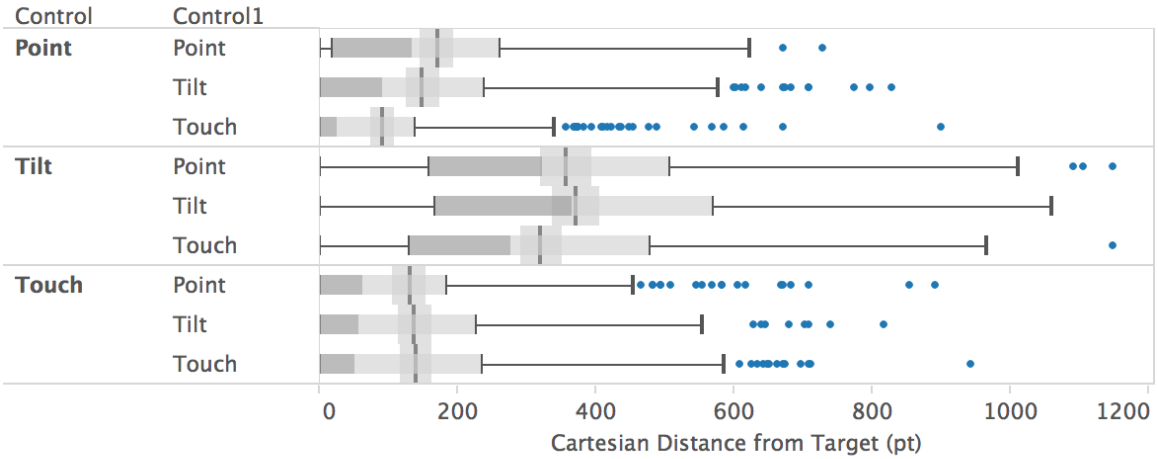

Figure 27: Distance from Target vs. Control & Control Order[1]

Again ignoring the main effects of the control scheme itself for now, the control order does not seem to have any significant impacts, with one notable exception: when testing Point controls,

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

distance is significantly reduced when Touch controls were tested first, as opposed to either of the other two. One obvious confound that could affect this is selection bias of the testing order based on gaming experience. For example, if more experienced players happened to test Touch first, this might make it appear that testing Touch first helps, but the average frequency of video games played for users who tested Touch first lands right in between the other two control types (based on the pre-test survey), as shown in the figure below.

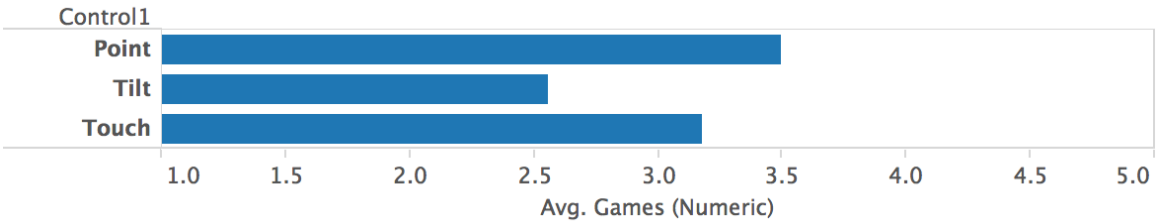

Figure 28: Frequency of Video Game Play vs. Control Tested First

It is thus possible, barring other unobserved confounds, that testing Touch first actually does improve performance in the Point environment. The averages indicate that testing Touch first may have also helped in the Tilt environment, although not enough to be statistically significant. Perhaps the Touch environment, likely being the most familiar of the three, allowed users to become familiar with the general framework using a more familiar control before testing the more advanced controls. More rigorous testing would need to be conducted to confirm this effect, but it is worth noting.

Lastly, performance with and without auto-lock does not appear to have been significantly affected by the order in which it was tested, as shown by the figure below. The average distance in one value of the factor was slightly decreased by testing the other value first, indicating the very slight learning effect observed in section 4.2.2. Nevertheless, as with the environment, this effect is not statistically significant.

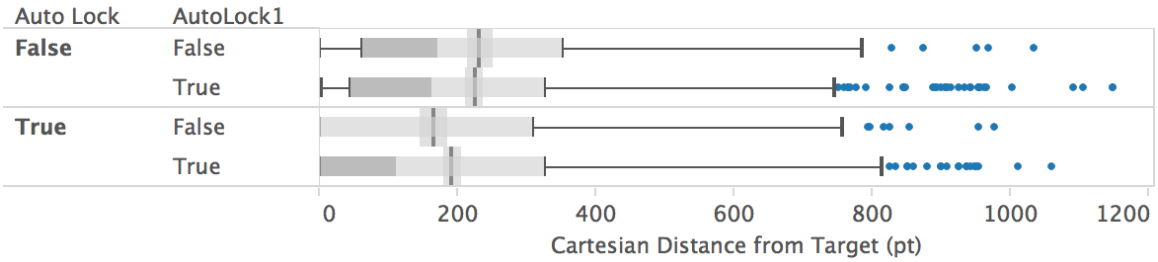

Figure 29: Distance from Target vs. Auto-Lock and Auto-Lock Order[1]

4.3 Accuracy – Preliminary Analysis

One of the most natural ways to illustrate accuracy in simple target shooting is with scatterplots. By subtracting the coordinates of each shot from its corresponding target, we can plot the relative coordinates of each shot on top of each other, as shown in the figure below, and make a few preliminary observations. Note that in this figure, only the "Off" state of auto-lock is represented, resulting in 6 scatterplots instead of 12. This was chosen intentionally due to the fact that the effect of auto-lock on accuracy is not as apparent in a scatterplot, due to repeated points occurring at (0, 0). The effects of auto-lock will be better illustrated in the upcoming section.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

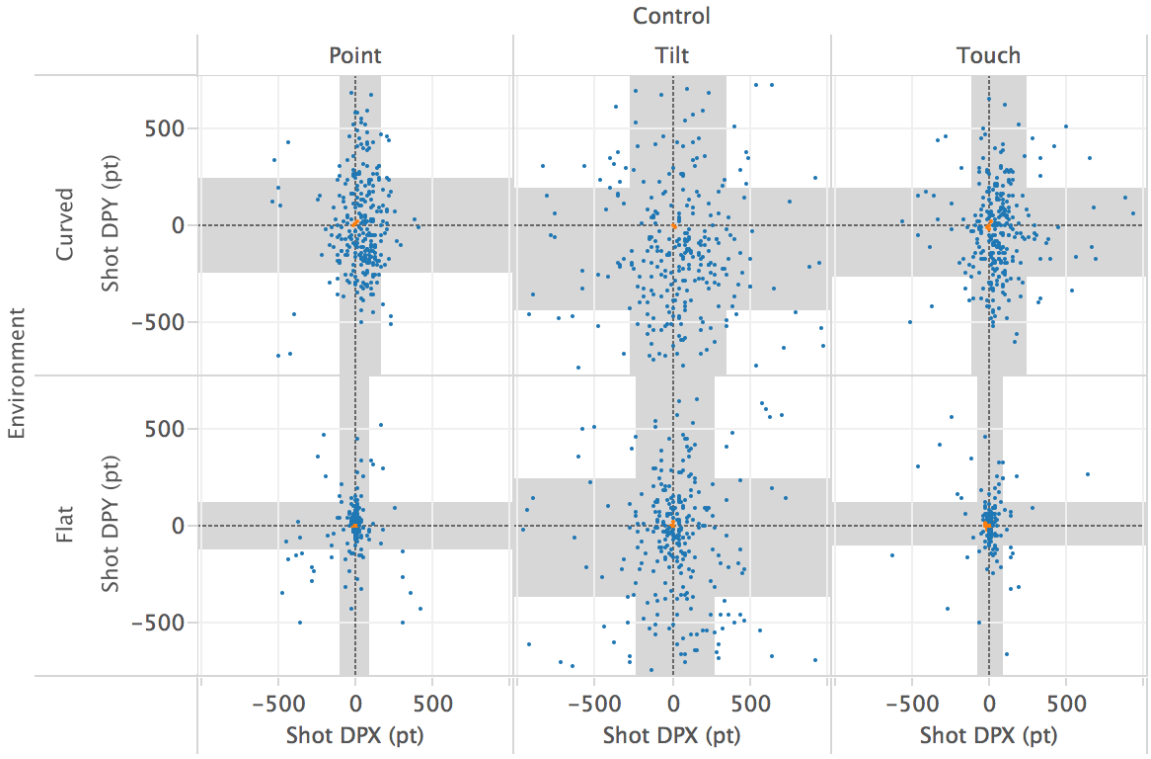

Figure 30: Scatterplot of Relative Shot Coordinates by Environment & Control, with Auto-Lock Off

The vertical and horizontal grey bands represent ±1 standard deviation from the mean for the X and Y coordinates in each subplot. Along with the scatterplot itself, the bands give away two of the main effects already. It is clear that Tilt suffers from lower accuracy than Point and Touch, and that the Flat environment produces higher accuracy than the Curved environment, but keep in mind that this is with auto-lock Off. These effects will be explored more formally in the upcoming section.

However, there are two other effects that are important to point out before looking at the main effects. The first is that the average shot in the Tilt environment is well below the target. Based on observations during both preliminary and actual study testing, this may be due to people's natural tendency to tip the front of the iPad slightly away from them rather than perfectly vertical when

using motion controls. When using Tilt controls, this has a dramatic effect, since the slightest angle can cause the player to move slowly downward, possibly without noticing.

The second and more disturbing effect, which is only slightly visible in Figure 30, is that the average shot in the Curved environment is slightly to the right. In fact, on average across all control schemes, the shot is 45pt to the right in the Curved environment (with auto-lock Off), which is 1pt greater than the size of the whole target, and is statistically significant based on the 95% confidence interval for the mean. This prompted a meticulous re-verification of the Curved environment mapping to ensure that a bug in the code had not caused this discrepancy. In the Curved environment, the 2D Cartesian coordinates of the real screen are translated into the spherical coordinate system of the Curved screen through a simple multiplier, but must then be transformed into a 3D Cartesian coordinate system in order to be placed in virtual space using the 3D audio API. In order to verify that this conversion was done correctly, the 3D coordinates of the target's audio node and the virtual orientation of the listener were observed directly with the debugger while aimed directly at the target, and the "correct" listener orientation was re-derived in reverse and calculated from the target's actual observed 3D Cartesian position, and verified to match the actual observed listener orientation from the debugger. There was no question that the virtual nodes were configured correctly.

There are a number of possible explanations for this, none of which are strictly verifiable within the scope of this study. The HRTF library used by AVAudioEngine could simply not have the resolution to support the accuracy attempted for this study, but in that case, one would expect to see randomly distributed errors, not a consistent deviation to one side. The library itself could have some bug in the listener orientation that causes this slight offset, but this would not be verifiable without significant testing. Anything in the signal path could be introducing a slight

stereo gain mismatch that, while not perceivable under normal listening conditions, showed up under the demanding application of 3D audio delivery. Elements that could have introduced this mismatch include other portions of the software audio stack behind the 3D audio library, the analog amplification components within one or both of the iPads, or the impedance/response characteristics of the left and right drivers in the headphones. Lastly, despite the confidence intervals, we must leave open the possibility that this effect is simply an anomaly.

Lacking the ability to test for any of these possible causes, and given that the difference in effectiveness between the Curved and Flat environments is the primary hypothesis of this study, it was deemed beneficial to estimate this problem's effect on the number of hits achieved in the Curved environment without auto-lock. If this observation is assumed to be caused by something, and if whatever caused it is assumed to be fixable, then by creating a new X variable that is offset by its own average (in the Curved environment, without auto-lock), a new Boolean "hit" variable can be created that simulates whether each shot would have hit this theoretically-corrected target. Based on this variable, the figure below visualizes the difference in the number of targets actually hit during testing vs. the number hit when the above assumptions and corrections are made.

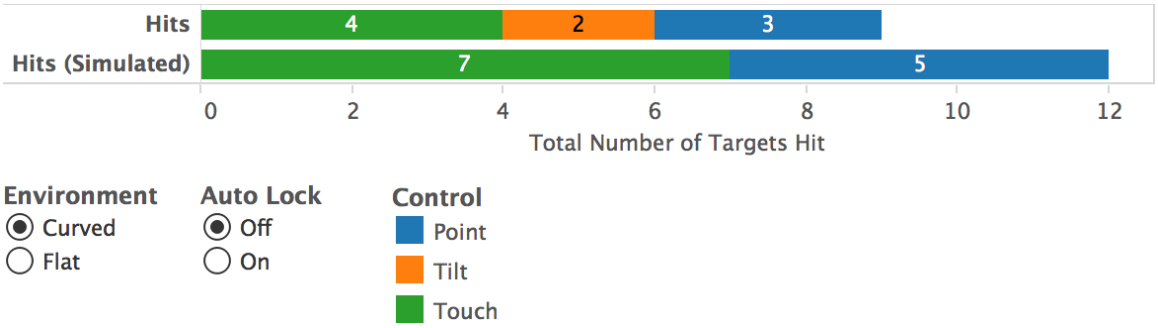

Figure 31: Actual vs. Simulated Hits in Curved Environment with Auto-Lock Off

The simulated metrics increased the total number of targets hit from 9 to 12. This may seem like a significant improvement, but consider the number of targets presented. A total of 3,360 targets were presented across all tests, 840 of which occurred in the scenario simulated above

(Curved environment, auto-lock Off), so the simulation changed the accuracy in this scenario from ~1.07% to ~1.43%. Even if this anomaly has a non-random cause, and even if that cause could be fixed, the Curved environment remains untenable for finding targets of this size, and the improvement shown here is dwarfed by the main effects of the factors present in this study, which are demonstrated in the forthcoming section.

4.4 Accuracy – Main Effects

This section introduces and discusses the main effects of the three factors. Shot DPX and Shot DPY represent the X and Y differences (in points) between the shots taken and their corresponding targets' locations. The descriptive statistics for the absolute error in both the X and Y directions (|Shot DPX| and |Shot DPY|) are summarized in the tables below.

Table 14: Descriptive Statistics for |Shot DPX|

		Environment					
		Curved			Flat		
Control	Auto Lock	Avg. |Shot DPX|	Median |Shot DPX|	Std. dev. of |Shot DPX|	Avg. |Shot DPX|	Median |Shot DPX|	Std. dev. of |Shot DPX|
Point	Off	99.3	81.1	92.3	44.4	11.4	88.6
	On	71.0	0.6	125.9	27.5	0.0	66.5
Tilt	Off	227.7	161.5	210.5	168.2	98.2	188.6
	On	195.4	138.8	201.1	182.6	102.7	213.1
Touch	Off	126.0	82.3	146.1	32.0	8.8	74.5
	On	73.6	5.3	122.4	38.6	0.0	105.3

Table 15: Descriptive Statistics for |Shot DPY|

		Environment					
		Curved			Flat		
Control	Auto Lock	Avg. |Shot DPY|	Median |Shot DPY|	Std. dev. of |Shot DPY|	Avg. |Shot DPY|	Median |Shot DPY|	Std. dev. of |Shot DPY|
Point	Off	193.7	164.6	145.2	66.7	27.5	98.0
	On	87.6	0.4	127.4	71.3	0.0	136.5
Tilt	Off	277.7	235.8	188.7	234.3	161.0	202.0
	On	247.0	184.8	205.5	231.5	166.8	209.2
Touch	Off	180.9	155.5	142.2	60.2	22.5	94.3
	On	121.3	31.3	158.3	48.9	0.0	101.4

While the main effects can be observed in the tables above, the following sections illustrate the differences with histograms of the absolute X and Y coordinates, as well as the box-and-whisker plots with superimposed averages (and confidence intervals) introduced in Figure 18, divided along the three factors.

4.4.1 *Environment Accuracy*

The primary factor of this study is the environment. The following two figures show the effects of the environment on the X and Y distribution of shots taken relative to their corresponding targets.

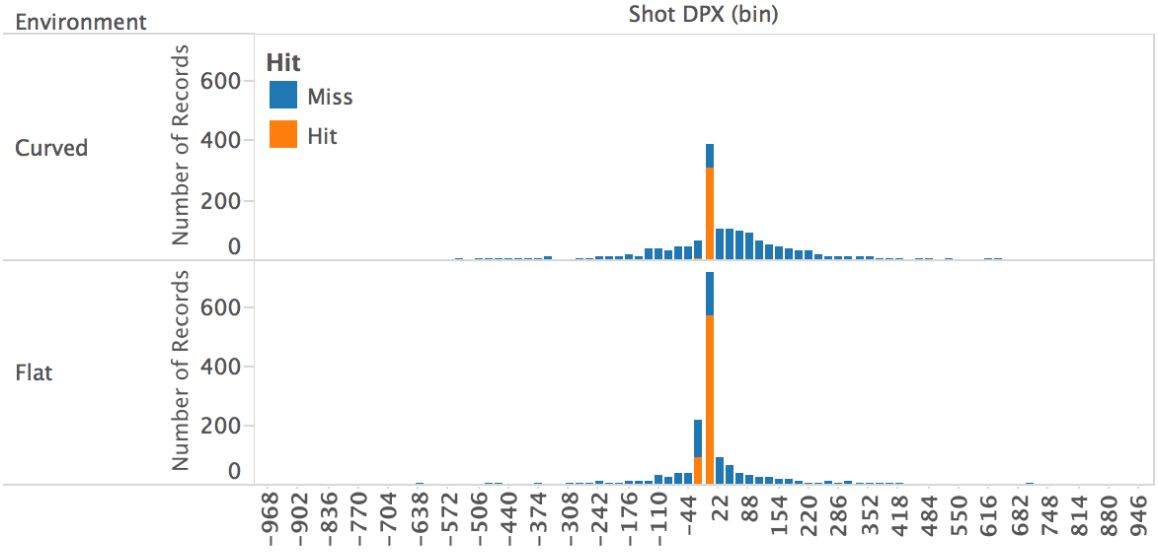

Figure 32: Histogram of Relative X Position of Shots Fired vs. Environment

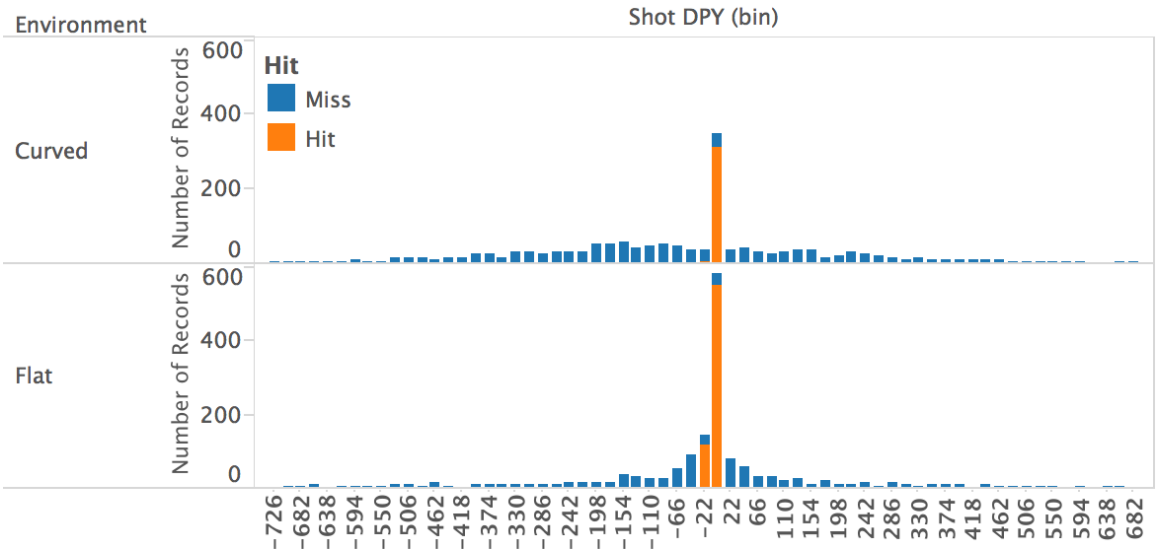

Figure 33: Histogram of Relative Y Position of Shots Fired vs. Environment

The effects of the environment are somewhat masked in the distribution views by the spike in the 0-22pt bin, due to the large number of records at (0, 0) created by the auto-lock function, but since this section looks specifically at the main effects, it is critical to include both states of auto-lock. Still, the effects can be seen in the surrounding bins, which are significantly more compressed around the center of the target in the Flat environment than in the Curved environment. To help quantify this effect, the figure below shows the quartiles, averages, and 95% confidence intervals for the absolute error in the X and Y directions.

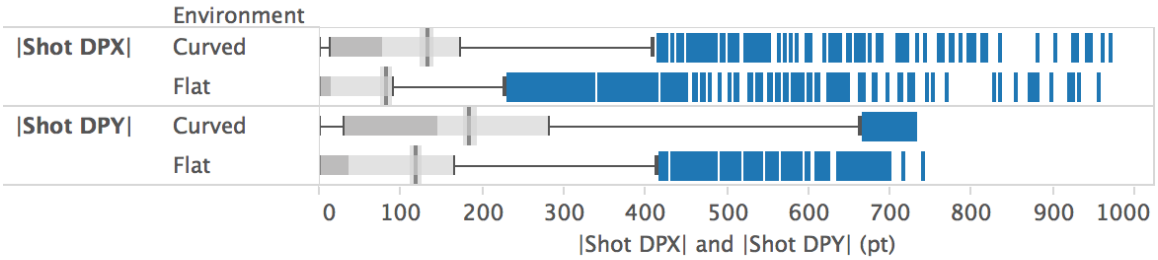

Figure 34: Absolute X & Y Error Statistics for Curved & Flat Environments[1]

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

Due to the number of points, the confidence intervals are so small as to be difficult to see on any reasonable scale for print, but suffice it to say, there is a clear statistically significant decrease in the absolute error for both X and Y directions when using the Flat environment instead of the Curved environment. The exact averages for both environments are shown in the figure below.

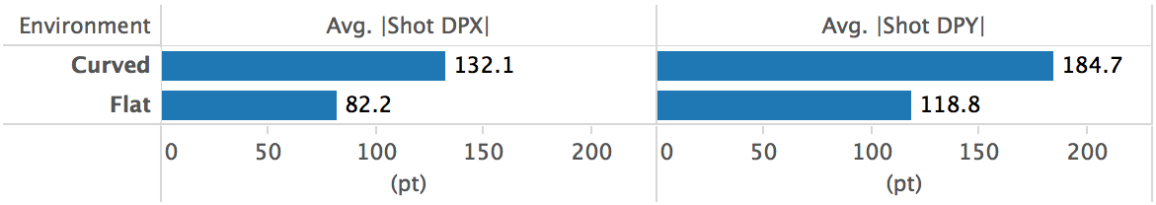

Figure 35: Absolute X & Y Error Averages for Curved & Flat Environments

This is the expected result, and confirms the primary hypothesis of this study: that the Flat environment allows users to be more accurate than the Curved environment. The reasons for this are discussed in section 3.1.1 of the Methods chapter, but another way to look at it is illustrated by the figure below.

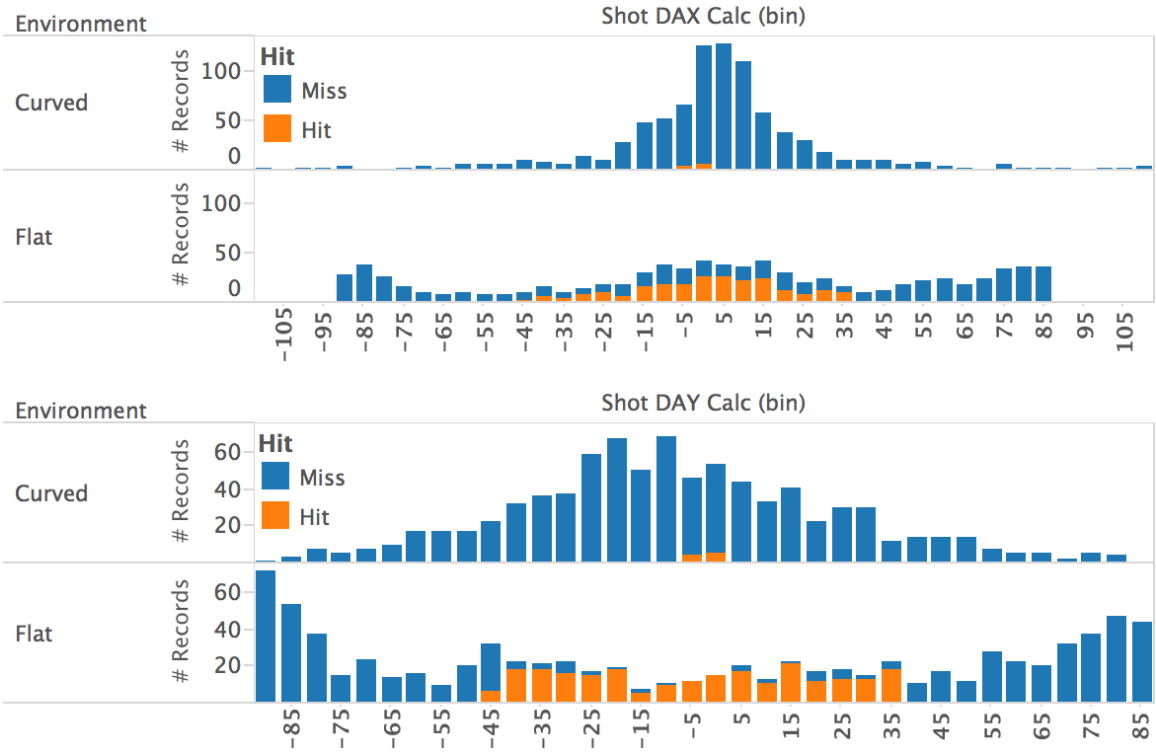

Figure 36: Histogram of Relative X & Y Angle between Shot and Target (Auto-Lock Off)

In the above figure, "Shot DAX Calc" represents the horizontal component of the angle between the user's virtual orientation when they fired and the virtual direction of the target position (in degrees), while "Shot DAY Calc" represents the vertical component of such (also in degrees). Successful hits are shown in orange, while misses are shown in blue. Auto-lock is off in all cases, to make it easier to see the differences. In the Curved environment, as discussed previously, the user's angular orientation is mapped linearly from the points on the screen, so the perceived angular target size, as calculated in Equation 3-1, is only 5.16º. In the Flat environment, the user's angular orientation is fixed straight forward, while they move around in front of the screen. Thus, as shown in Equation 3-2, the perceived angular size (when directly in front of the target) is 81.35º. Due to the nearness, large size, and shape of the screen in the Flat environment, most of the screen area is at an extremely high angle away from the user at any given time. Thus, we expect to see the wider distribution skewed to the outer edges as shown in Figure 36 above. However, because of the increased perceived angular size due to the mapping of the environment, a much wider angular error is tolerated once the general vicinity of the target has been found. This is why we see a wider angular spread of hits in the Flat environment than the Curved environment in Figure 36.

4.4.2 *Control Accuracy*

The following two figures show the effects of the control scheme used on the X and Y distribution of shots taken relative to their targets.

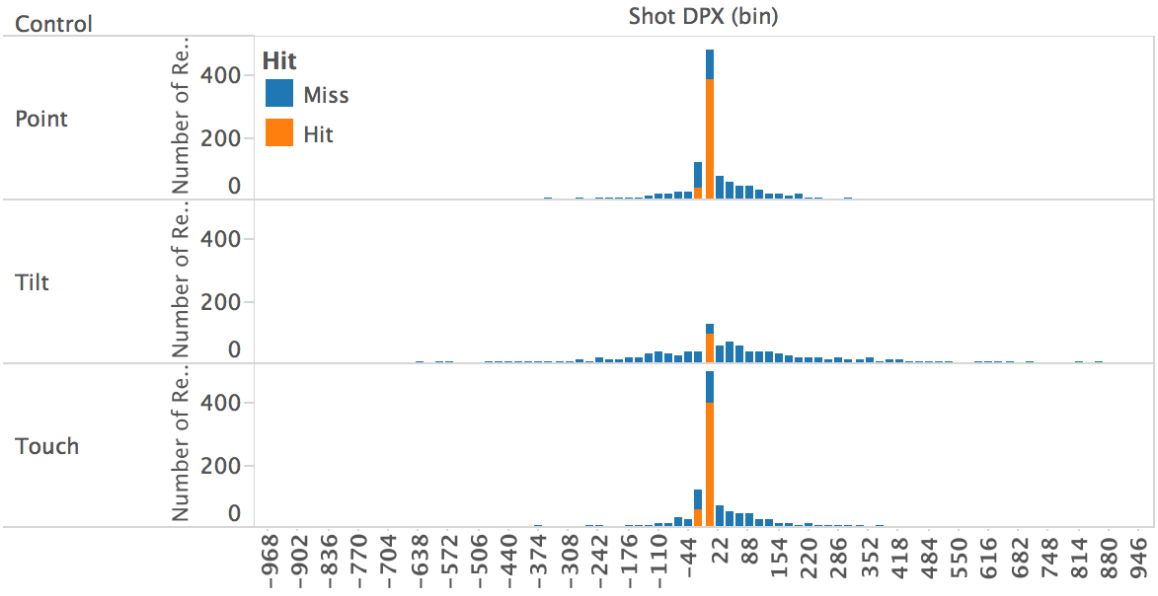

Figure 37: Histogram of Relative X Position of Shots Fired vs. Control Scheme

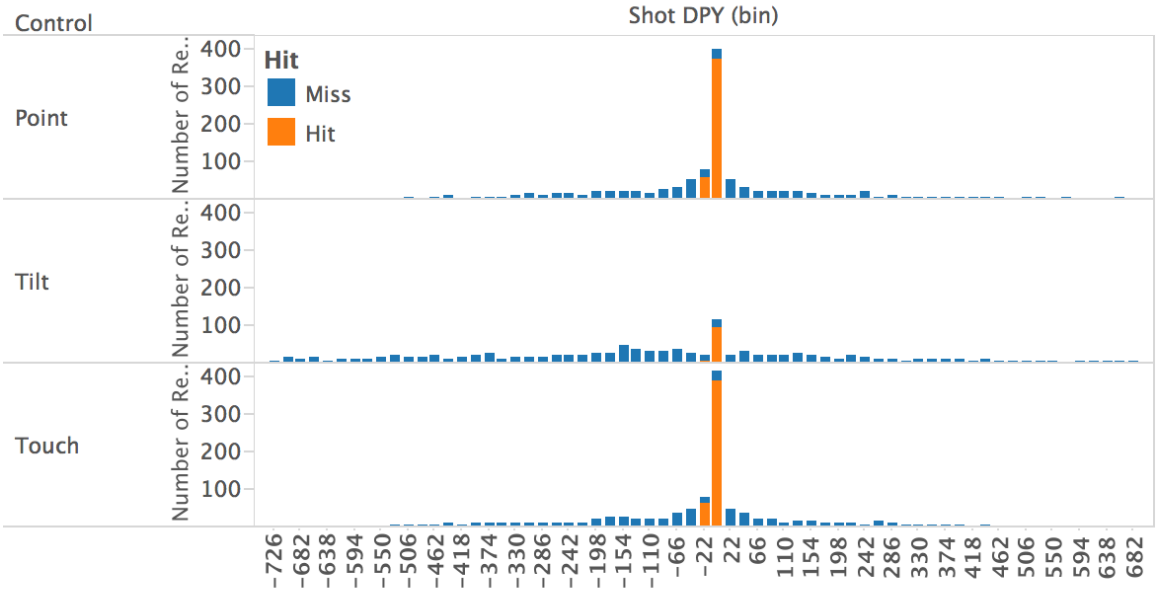

Figure 38: Histogram of Relative Y Position of Shots Fired vs. Control Scheme

As in section 4.4.1, the effects of the control scheme used are somewhat masked by the center spike caused by auto-lock, but a close look can still see the increased accuracy with the Point and Touch controls compared to the Tilt controls, in both the X & Y directions, similar to what we saw in the scatter plots in Figure 30. Again, the descriptive statistics help us quantify this effect, as

visualized in the figure below. Note that in this case, 99% confidence intervals are used to account for loss of significance with multiple comparisons, since there are more than two control schemes.

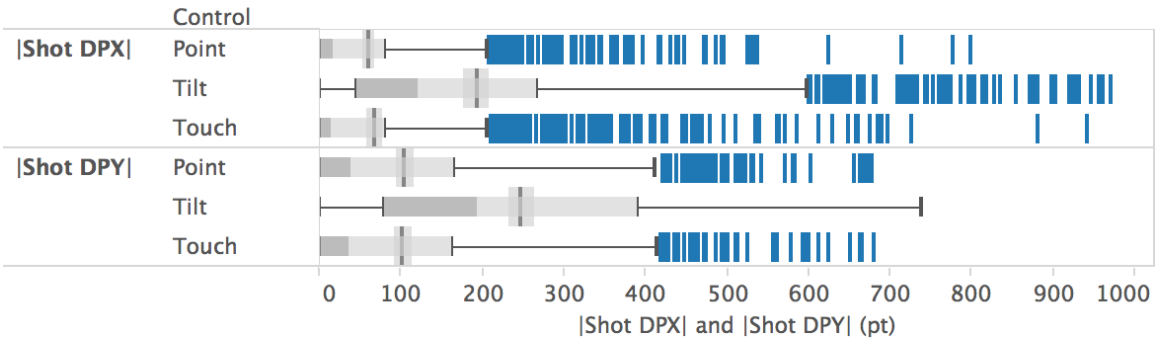

Figure 39: Absolute X & Y Error Statistics for Touch, Tilt, & Point Controls[1]

Even with the smaller number of points per category due to there being three values of this factor (instead of two), the confidence intervals are still quite small, even at 99%. The Tilt controls far underperform both Point and Touch controls, while Point and Touch are statistically insignificant from each other, since their confidence intervals overlap for both X and Y error. The exact quantities for the averages are shown in the figure below.

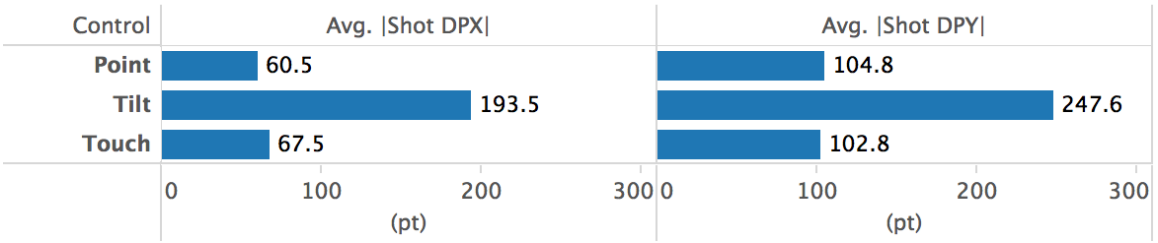

Figure 40: Absolute X & Y Error Averages for Touch, Tilt, & Point Controls

There is no better way to understand the problem with the Tilt controls than by plotting the absolute coordinates of every shot taken and color-coding it by control type, as shown in the figure below.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

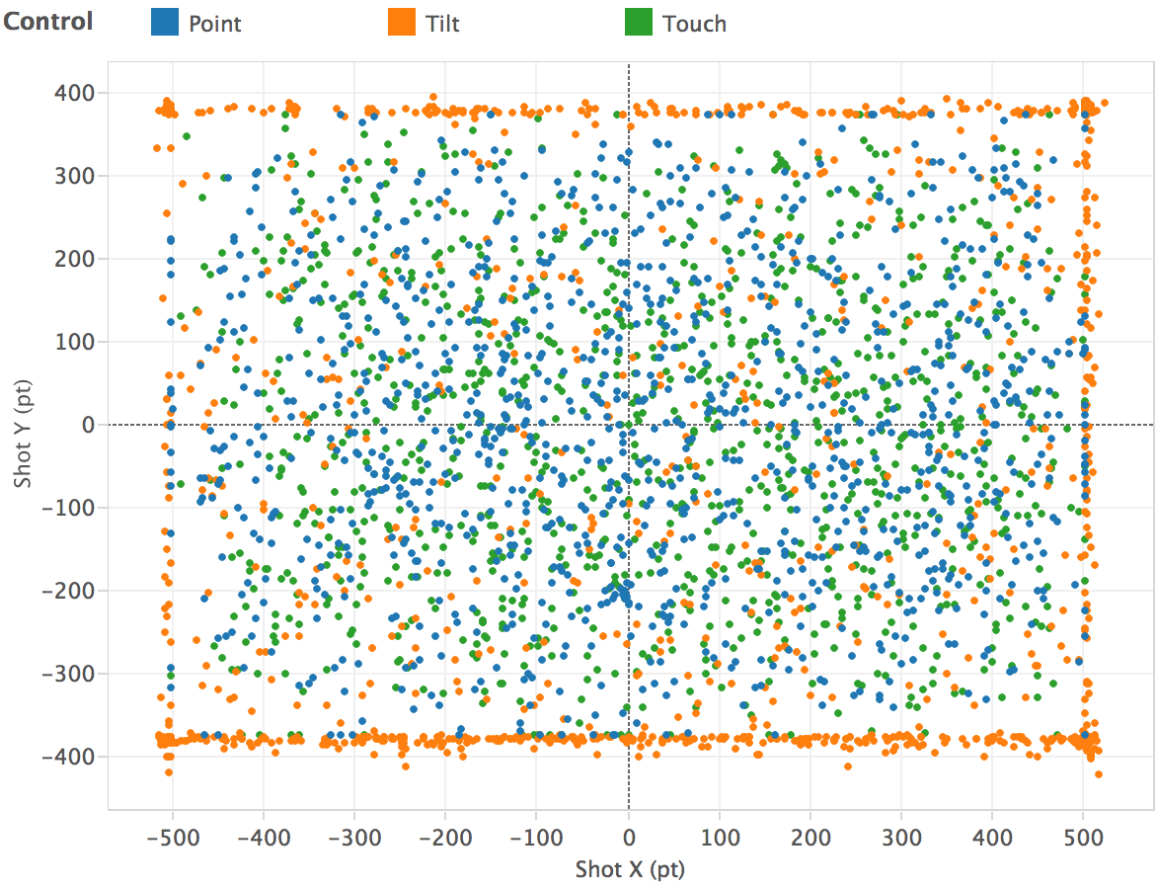

Figure 41: Absolute Location of All Shots Taken

This figure is relatively self-explanatory for anyone who has experienced the test. Shots taken with Tilt controls are shown in orange, while shots taken with Point and Touch controls are shown in blue and green, respectively. The thick orange border illustrates the primary problem with Tilt controls. Users tend to get stuck on the walls, particularly in the corners. The Tilt controls are unique in that they are the only control scheme of the three that does not offer any direct sense of the player's absolute coordinates at a given point in time. With Touch controls, it is possible to feel where one's finger is in relation to the edges of the screen. With Point controls, users have a sense of what direction they are pointing the iPad. With Tilt controls, even if the iPad is held perfectly vertically so as to remain stationary, there is still no way to know one's current location on the screen. Users who fired while tilting aggressively into the wall were even able to stretch

beyond the physics boundary of the world before the collision was detected, as seen primarily in the corners.

4.4.3 *Auto-Lock Accuracy*

The following two figures show the effects of auto-lock on the X and Y distribution of shots taken relative to their targets.

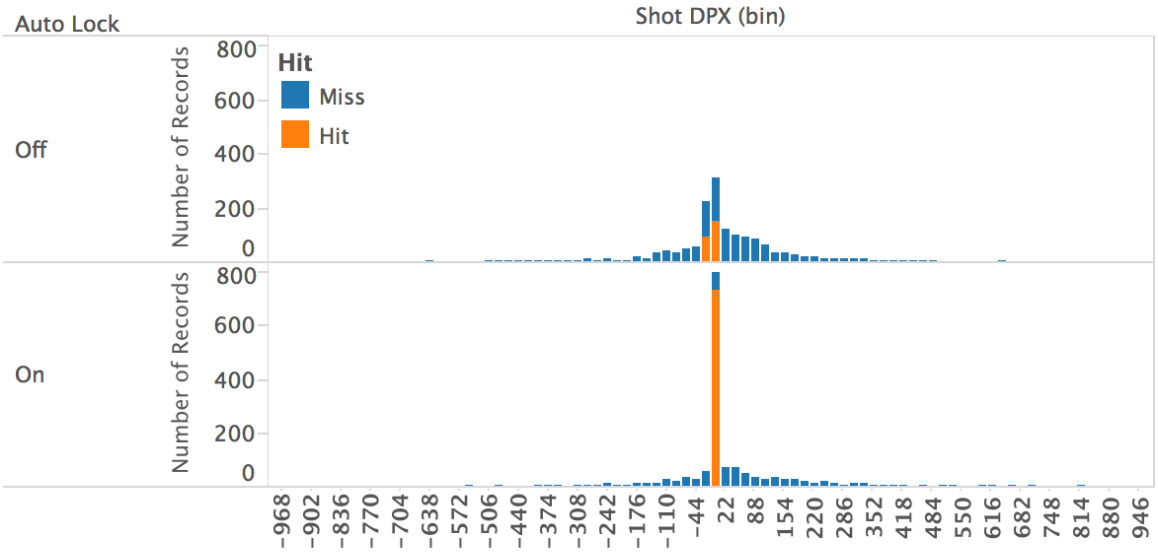

Figure 42: Histogram of Relative X Position of Shots Fired vs. Auto-Lock State

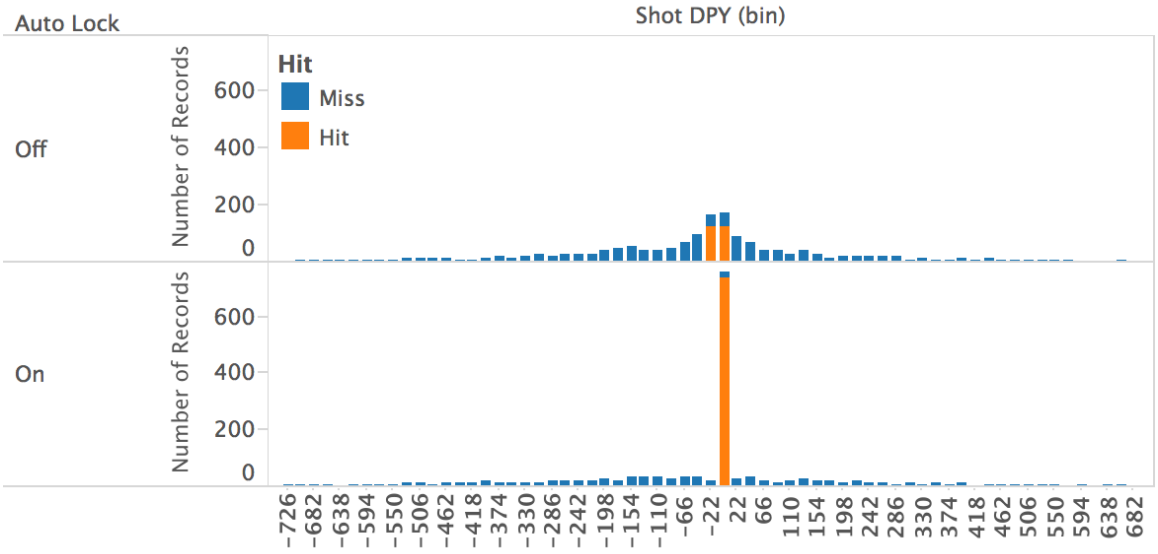

Figure 43: Histogram of Relative Y Position of Shots Fired vs. Auto-Lock State

The above two figures finally reveal where the tall central spike came from in the previous distributions for environment and control. The effects of auto-lock could not be more clear, but the figure below helps to quantify the significance of its impact on distance from the target.

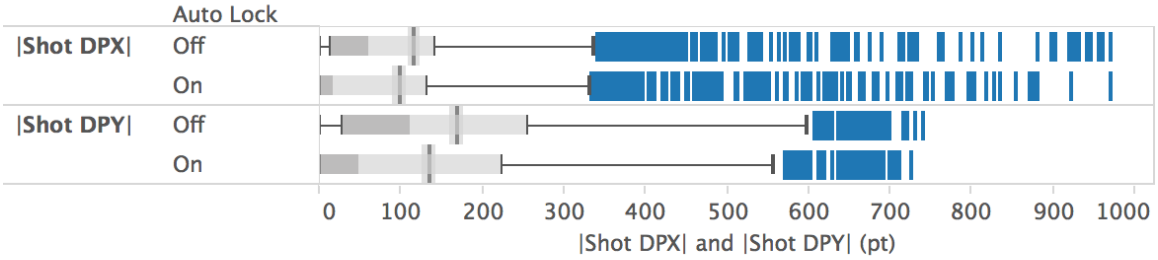

Figure 44: Absolute X & Y Error Statistics for Auto-Lock On & Off[1]

Again, the large number of data points yields a narrow 95% confidence interval, and it is clear that auto-lock decreases the average distance by a statistically significant amount in both the X and Y dimensions. The exact averages with and without auto-lock are shown in the figure below.

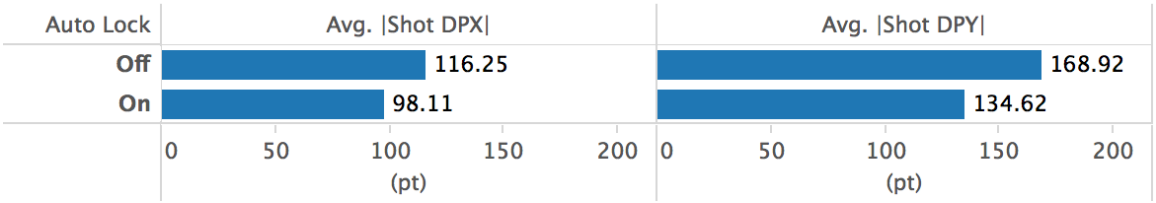

Figure 45: Absolute X & Y Error Averages for Auto-Lock On & Off

The objective advantages of auto-lock is relatively self-explanatory, and thus does not merit significant discussion. More interesting effects of auto-lock will be seen in the next section, which explores interactions between the factors and considers hit percentages in addition to Cartesian distance.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

4.5 Accuracy – Interaction Effects

One of the goals of this study, as with any factorial design, was to explore the interactions between the three factors. Such analysis requires the data to meet the assumptions of the 3-way analysis of variance (ANOVA), including normality and homogeneity of variance. Unfortunately, the data generated here do not meet such criteria. Several transformations were attempted, but none were successful in meeting the assumptions. The primary problem comes from the fact that the raw data (relative X and Y positions) are primarily centered around zero, with several values occurring exactly at zero. Since ANOVA analyzes the difference between means, not the differences between variances, the raw values must be somehow transformed into positive error values, any reasonable approach for which will still result in several values at zero, and a non-normal distribution in general.

Since no non-parametric technique could be found for 3-way ANOVA, only the main effects from the previous section are formally provable. Nevertheless, the interaction effects are still observable, though not statistically verifiable. This section explores the three 2-way interactions using line plots. There are two simple, natural ways of combining the X and Y data into a single number to simplify interaction analysis: average Cartesian distance and percentage of targets hit. The following two figures show the values for these two metrics for all 12 cells of the 2x3x2 factorial study.

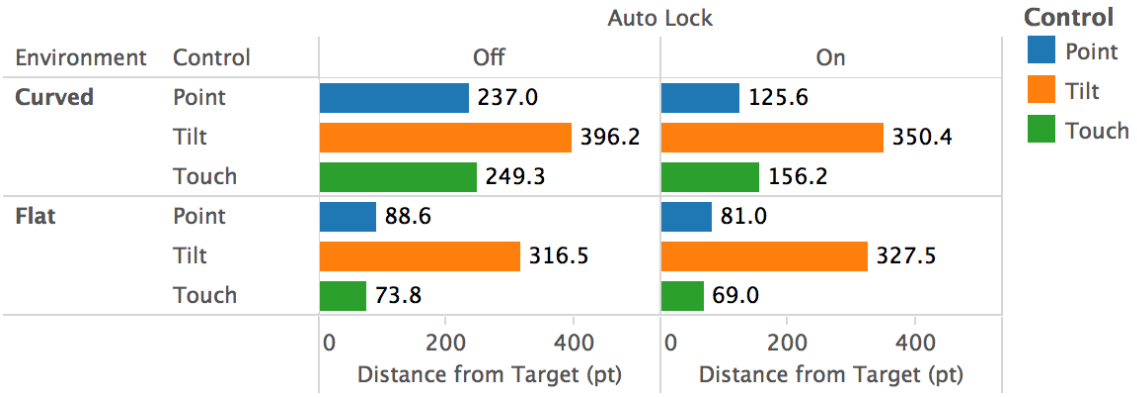

Figure 46: Average Cartesian Distance from Target for All Factor Value Combinations

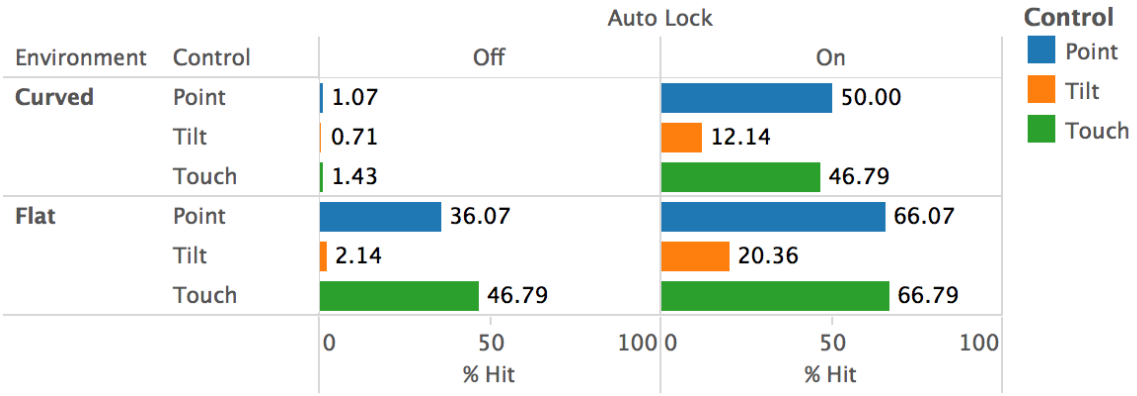

Figure 47: Percentage of Targets Hit for All Factor Value Combinations

Although both of the above figures effectively show the impacts of the three factors, they are difficult to compare alongside each other, since their directions of improvement are opposite. Improvements along a given vector would be indicated by lower average distance, but higher hit percentages. Cartesian distance is not naturally invertible, so in order to make these two metrics more comparable, the hit percentage can be transformed to a miss percentage, so that lower values are desirable for both metrics. The miss percentages for all combinations are shown in the figure below.

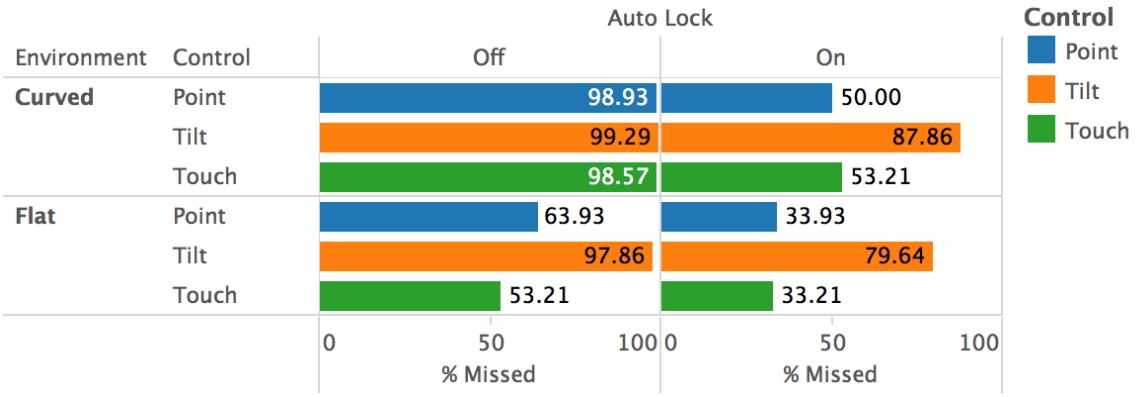

Figure 48: Percentage of Targets Missed for All Factor Value Combinations

Now, using the Cartesian distances from Figure 46 and the miss percentages from Figure 48, there are two comparable metrics with which to evaluate the interaction effects, which are discussed in the following subsections.

4.5.1 *Auto-Lock vs. Control*

The figure below shows the average Cartesian distance and percentage of targets missed for the two states of auto-lock (On and Off), using all three controls.

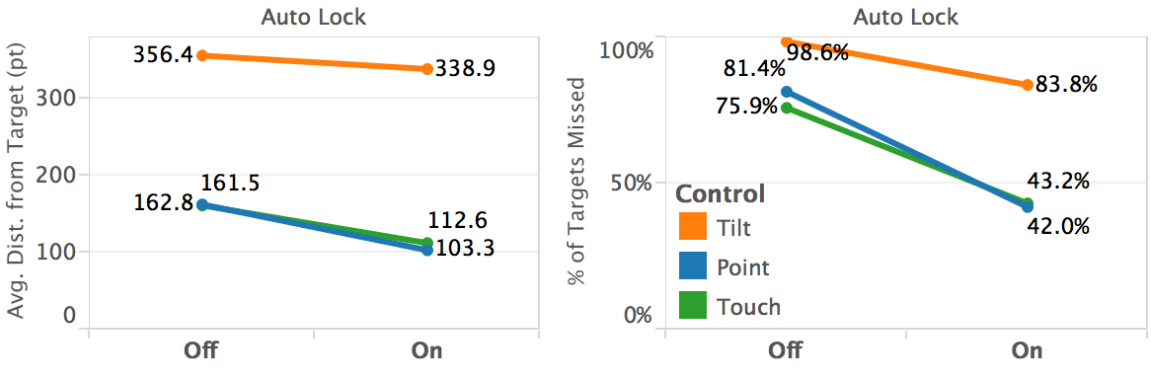

Figure 49: Auto-Lock vs. Control - Average Distance and Miss Percentages

The two main effects are clearly visible, with the auto-lock effects shown by the general downward slope to the right for all lines, and the control effects shown by the differences in vertical line position between Tilt and the other two control schemes. Interaction effects in these graphs

are represented by differences in the slope of the three lines. In general, it appears that auto-lock improved Point the most (since it has the steepest slope in both cases), followed by Touch. Tilt was improved the least by auto-lock. None of these interactions look particularly strong, although they appear to be more accentuated in the miss percentages than the average distance. Keep in mind, however, that there could be a ceiling effect on the Tilt environment that falsely enhances this interaction, since it's not possible to miss more than 100% of the targets.

4.5.2 *Environment vs. Control*

The figure below shows the average Cartesian distance and percentage of targets missed for the two environments, using all three controls.

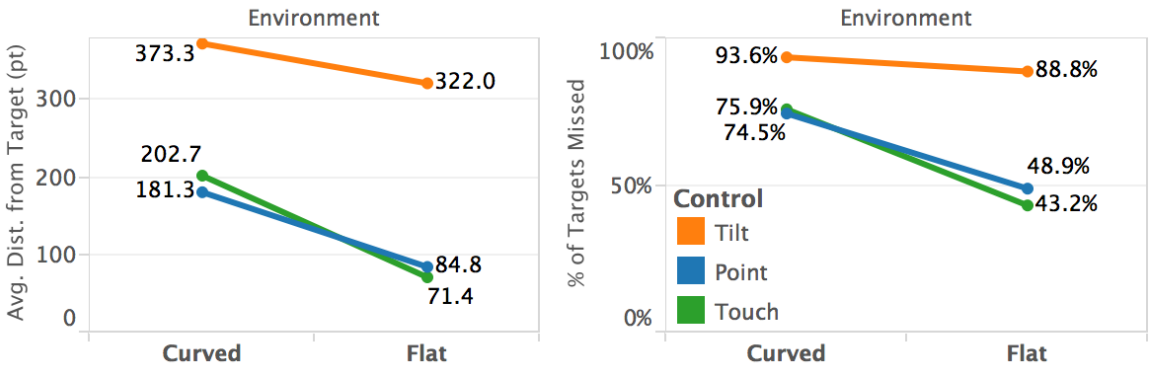

Figure 50: Environment vs. Control - Average Distance and Miss Percentages

As in the previous section, the two main effects are clearly visible, with the environment effects shown by the downward slope to the right for all lines, and the control effects again shown by the difference in vertical line position between Tilt and the other two controls. Interaction effects appear similar to the previous section, but with the roles of Touch and Tilt reversed. The Touch environment seems to be improved the most by the Flat environment, followed by Point. Tilt is improved the least. As before, the interactions appear to be more visible in the miss

percentages than in Cartesian distance, again keeping in mind the possible ceiling effect on the miss percentages.

4.5.3 *Environment vs. Auto-Lock*

The figure below shows the average Cartesian distance and percentage of targets missed for the two environments, with and without auto-lock.

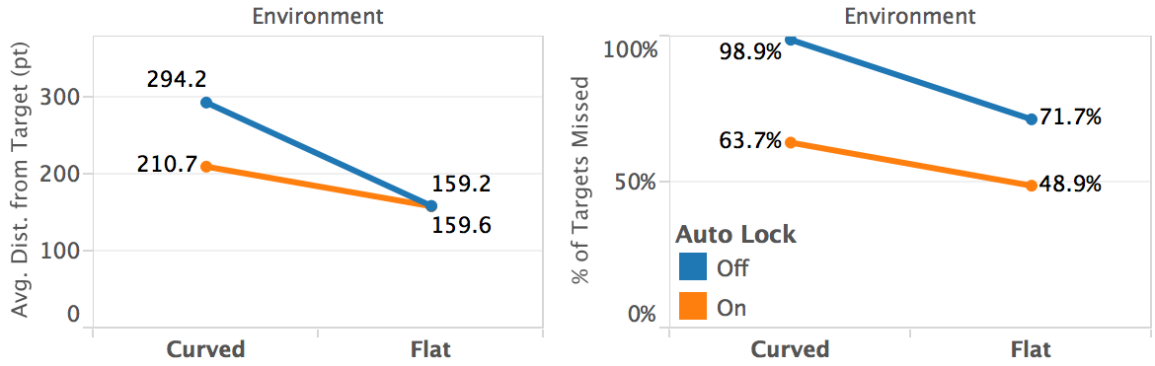

Figure 51: Environment vs. Auto-Lock - Average Distance and Miss Percentages

Again, the two main effects are clearly visible, with the environment effects being represented by the downward slope to the right for all lines, and the auto-lock effects being represented by the vertical separation of the two lines. The distance graph shows perhaps the strongest interaction so far, where auto-lock appears to have had no effect in the Flat environment, despite its strong effect in the Curved environment. It is thus tempting to conclude that the Flat environment is accurate enough that it doesn't benefit from auto-lock. However, while this may be true for average distance, it is not true for miss percentages, where auto-lock *clearly* decreases the miss rate in the Flat environment, thus weakening the interaction effect, although it is still present. In other words, the Flat environment provides a remarkable improvement on average distance (even more than auto-lock on its own), but auto-lock still helps convert near-misses to hits.

4.6 SPEED – MAIN EFFECTS

There are a number of ways that the "speed" of an interface could be measured. In this study, it is represented by the time between a target's appearance and the user's attempted selection. All subjects with the exception of the first participant were told that averaging about 10 seconds per target would allow them to complete the testing in about an hour. The first participant did not receive this instruction until after the first of the 12 tests, resulting in a few extreme data points on the order of minutes. A few shots from other users exceeded the one-minute mark as well. Given that the targets were intended to be sought for a matter of seconds, all analysis performed in this section excludes shots that took more than a minute, of which there are 29. Since there were 3,360 shots taken in total, this results in an exclusion of less than 1% of the total number of records. Furthermore, although all selection attempts are shown in the histograms, hits and misses are highlighted separately, and only successful hits are included when calculating statistics for speed. For reference, the descriptive statistics for successful hits (for shots taken in a minute or less) are shown in the table below.

Table 16: Descriptive Time Statistics for Successful Hits (t <= 60s)

		Environment					
		Curved			**Flat**		
Control	**Auto Lock**	Avg Time	Median Time	Std Dev Time	Avg Time	Median Time	Std Dev Time
Point	Off	9.86	7.18	4.82	11.95	11.02	5.96
	On	13.66	11.92	8.39	9.90	8.76	6.18
Tilt	Off	15.40	15.40	5.32	10.73	11.07	2.15
	On	14.15	10.64	9.20	14.40	11.88	10.33
Touch	Off	15.05	16.47	5.13	12.61	11.06	6.41
	On	16.56	12.77	12.50	12.43	10.30	8.58

4.6.1 *Overall Speed vs. Accuracy Correlation*

Before jumping into the main effects for speed, it is worth exploring whether there is an overall correlation between the time taken searching for the target and the resulting shot's distance from the target. The figure below shows a scatterplot of time vs. Cartesian distance for all shots taken in a minute or less.

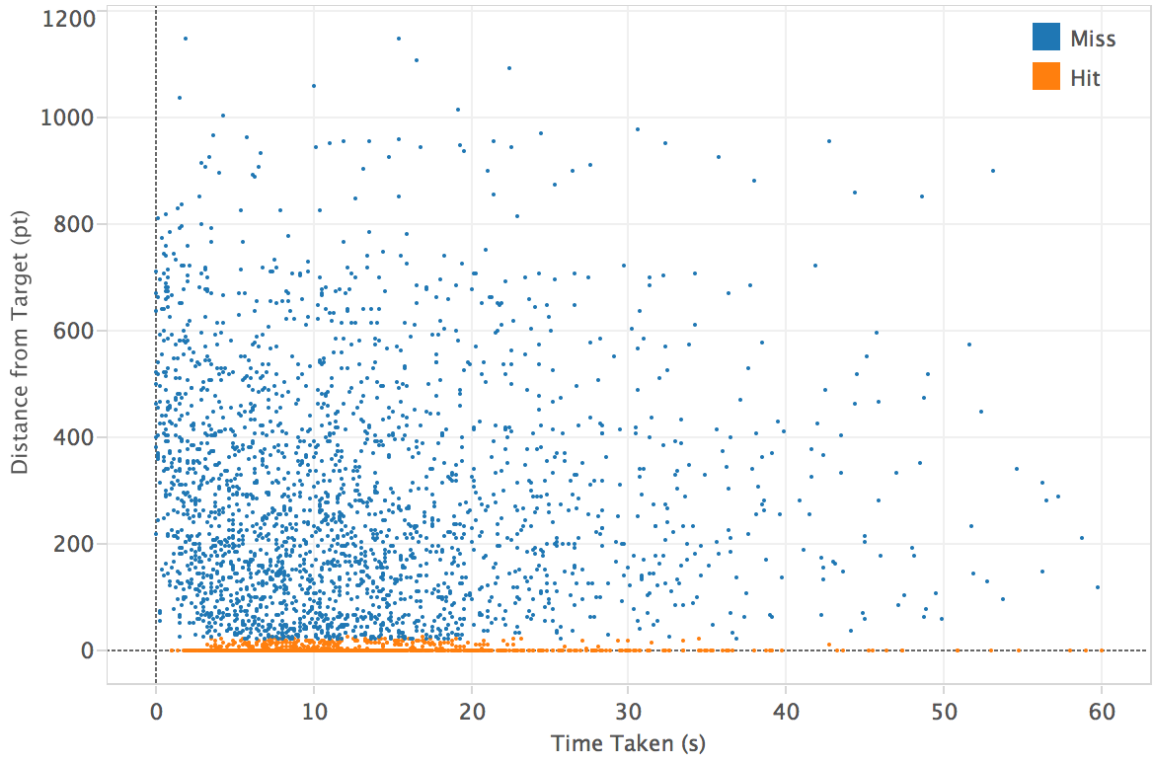

Figure 52: Time vs. Distance from Target (t <= 60s)

While the scatterplot makes it look like there might be a slight correlation, this does not appear to be the case when analyzed formally. The **Pearson correlation** coefficient for the relationship between time and distance was found to be **-0.017** via SPSS, with a **statistical significance** of **0.337**, indicating that there is no clear increase in accuracy for more time taken. The blank space in the lower-left corner of Figure 52 leads us to believe that a correlation might possibly exist if the time was limited to a much smaller value, but given the priorities of this study and the variance shown in the figure above, such exploratory analysis was not performed.

4.6.2 *Environment Speed*

The following two figures show the histograms of the amount of time taken as well as the descriptive statistics for successful hits within the two different environments. Note that, as mentioned at the beginning of this section (4.6), shots that took more than a minute are excluded.

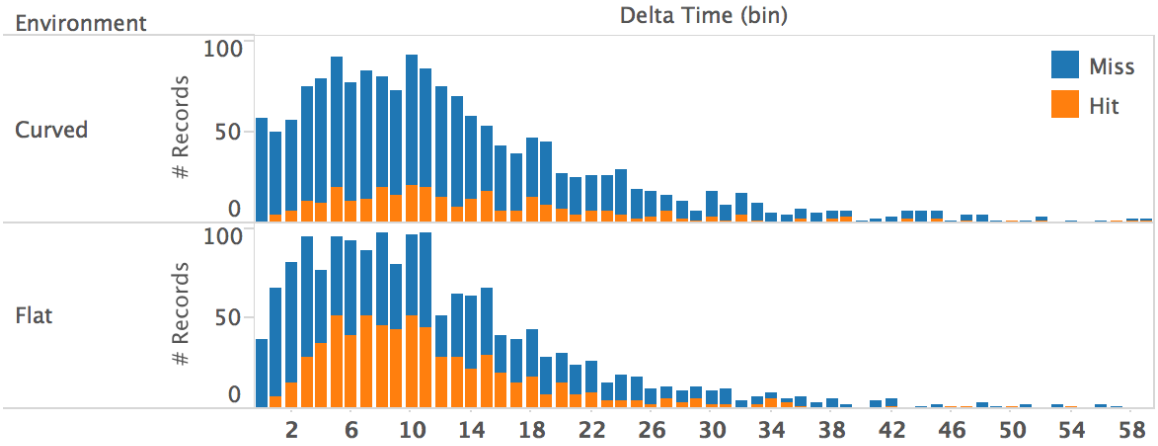

Figure 53: Histograms of Time Taken vs. Environment ($t \leq 60s$)

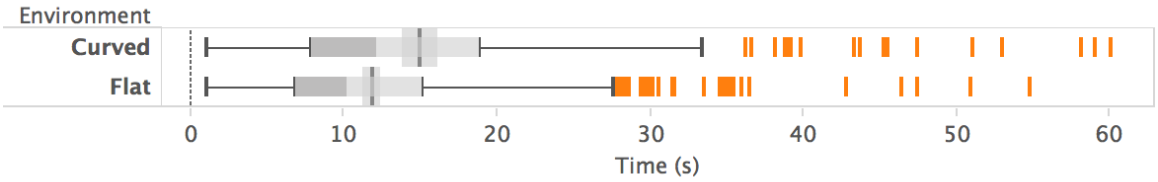

Figure 54: Time Statistics for Successful Hits vs. Environment ($t \leq 60s$)[1]

As with the accuracy statistics in section 4.4, the large number of points yields relatively narrow 95% confidence intervals for the average, showing a significant difference between the Curved and Flat environments in Figure 54, with the Flat environment being faster. The exact averages are shown in the figure below.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

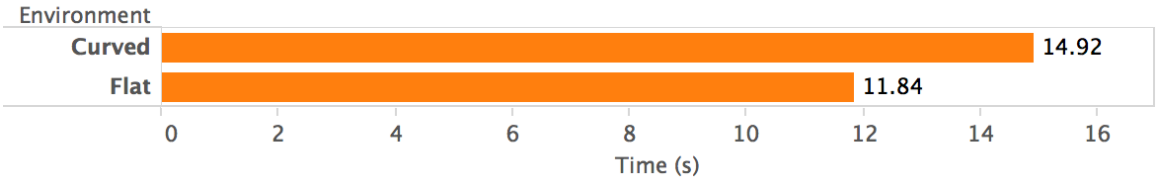

Figure 55: Average Time for Successful Hits vs. Environment (t ≤ 60s)

4.6.3 *Control Speed*

The following two figures show the histograms of the amount of time taken as well as the descriptive statistics for successful hits with the three different control types. Note that points where t > 60 seconds are excluded, and 99% confidence intervals for the average are used instead of 95% confidence intervals to account for multiple comparisons, since there are three control types.

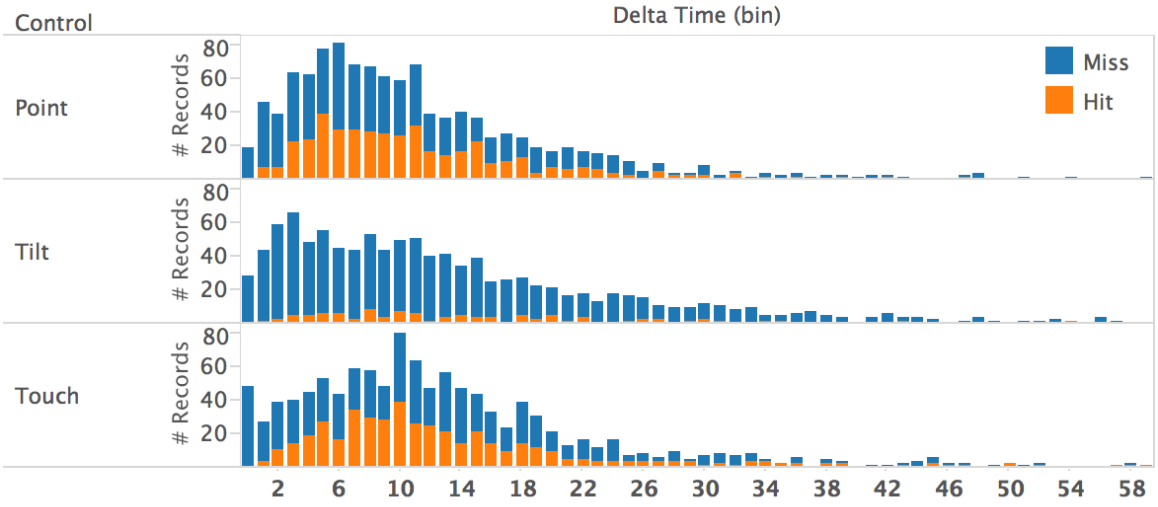

Figure 56: Histograms of Time Taken vs. Control Scheme (t ≤ 60s)

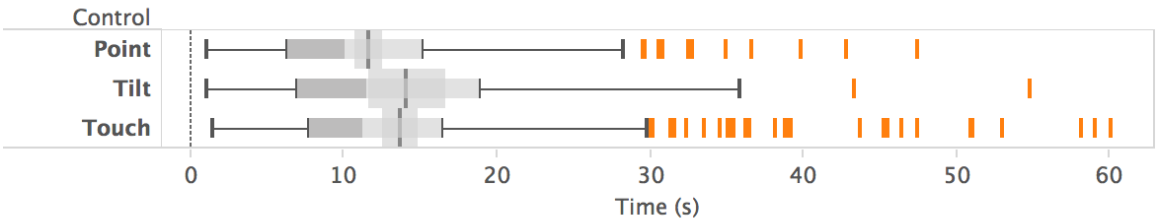

Figure 57: Time Statistics for Successful Hits vs. Control Scheme (t ≤ 60s)[1]

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

The fact that there are three types of controls (as opposed to two types of environment) has a twofold effect on the width of the confidence intervals. First, the increased number of groups causes there to be less samples in each group, which widens the confidence interval. Furthermore, Figure 57 uses 99% confidence intervals around the average instead of 95%, in order to account for multiple comparisons. Furthermore, since so few successful hits were achieved in the Tilt environment overall (as evidenced by Figure 56), its sample number is even lower, resulting in a much wider confidence interval in Figure 57.

Although it's difficult to spot, the confidence intervals for Point and Touch do not overlap, indicating that Point is faster than Touch. Based solely on the average, it looks like Point is likely faster than Tilt as well, but due to the small number of hits in the Tilt environment, its confidence interval is too wide for this difference to be statistically significant. The exact averages for all three control schemes are shown in the figure below.

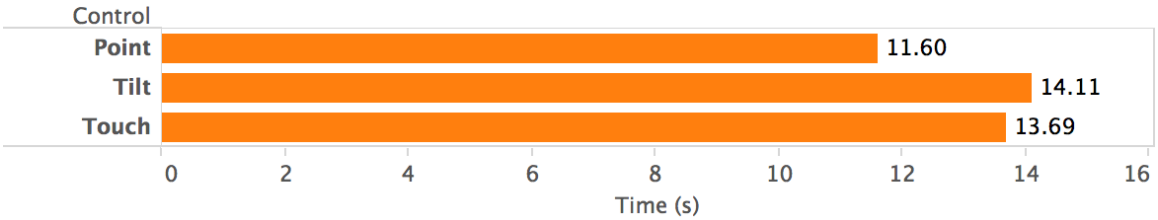

Figure 58: Average Time for Successful Hits vs. Control Scheme ($t \leq 60$s)

4.6.4 *Auto-Lock Speed*

The following two figures show the histograms of the amount of time taken as well as the descriptive statistics for successful hits with the two different states of auto-lock (On and Off). Note that, as in all other cases, points where $t > 60$ seconds are excluded.

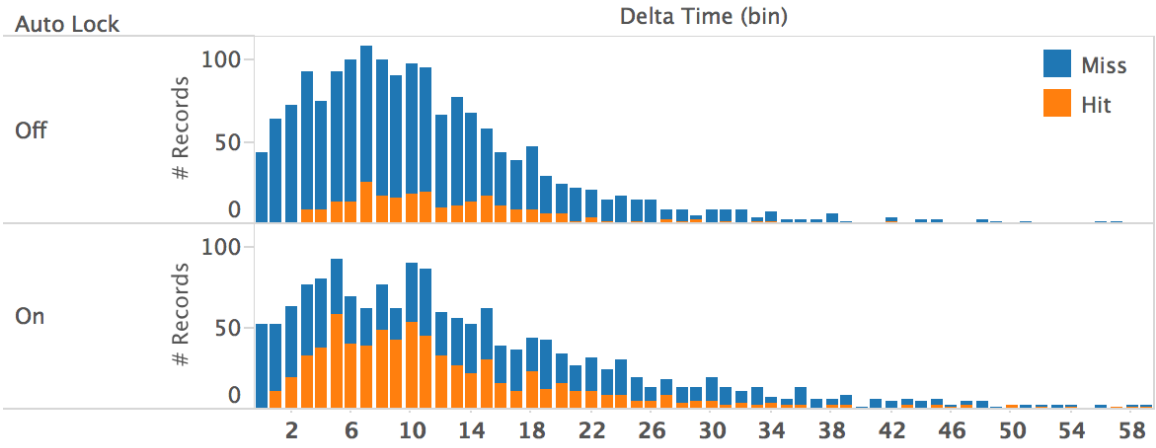

Figure 59: Histograms of Time Taken vs. Auto-Lock State ($t \leq 60s$)

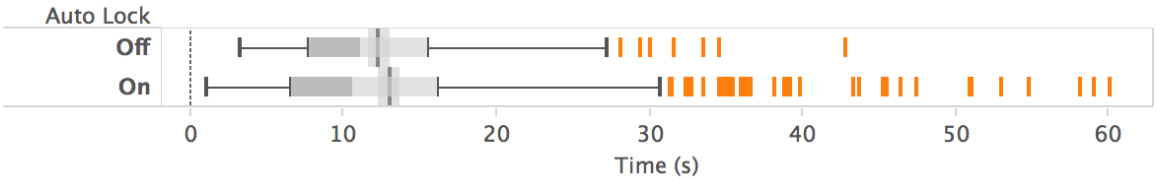

Figure 60: Time Statistics for Successful Hits vs. Auto-Lock State ($t \leq 60s$)[1]

As with the environments, the large number of points yields relatively narrow 95% confidence intervals for the averages in Figure 60, but in this case, they overlap significantly, so there is no statistically significant difference in average speed between the two states of auto-lock. If anything, it appears that the average time taken may have *increased* when auto-lock was On, due to the high number of outliers as shown in the figure above, but it should also be noted that the *median* appears lower with auto-lock On (ignoring confidence intervals for the moment), indicating the average-skewing effect of the aforementioned outliers, so a different choice in the filter cutoff of outliers could potentially yield a different result. No such exploratory analysis was performed, to avoid confounding the results with biased filtering.

[1] Refer to Figure 18 and surrounding discussion for proper interpretation of this figure.

4.7 Survey Results – Pre-Analysis & Reliability

As discussed in the methods section, the surveys ask several questions to collect feedback along three distinct vectors: Enjoyableness, Immersion, and Usability. The Environment A & B Surveys are asked after the first and second halves of the test, depending on the order in which the environments were tested, and each question is answered six times (once for all combinations of the three controls and two states of auto-lock). The full question text along with the general construct it was intended to measure and a shorthand reference for each question (which will be used in the figures in the following section) are shown in the table below. Note that some scales have been reversed from the original question so that high scores are always desirable, with the exception of the "Challenge" question, for which the most desirable answer is in the center of the scale (4), where the challenge is neither too high nor too low.

Table 17: Survey Constructs & Shorthand Guide

Construct	**Metric (Shorthand)**	**Metric (Full Question)**
Enjoyableness	Fun (Disagree-Agree)	The experience was fun. (Disagree-Agree)
Immersion	Know Aiming (Difficult-Easy)	I found it __ to know where I was aiming. (Easy-Difficult)
	Imagine (Disagree-Agree)	I was able to imagine where the sounds were coming from. (Disagree-Agree)
	Physical Connection (Disagree-Agree)	I felt a strong physical connection between the controls and the environment. (Disagree-Agree)
	2D-3D	The experience felt __. (2-Dimensional – 3-Dimensional)
Usability	Challenge (TooLow-TooHigh)	I found the experience __. (Not challenging enough – Too challenging)
	Confusing-Intuitive	I found the interface __. (Intuitive-Confusing)
	Slow-Fast	I found the interface __ when using it.
	Control Aiming (Difficult-Easy)	I found it __ to control where I was aiming. (Easy-Difficult)
	Enough Time (Disagree-Agree)	I had enough time to familiarize myself with the interface before the test ended. (Disagree-Agree)

In order to evaluate the consistency of the metrics within each construct, Chronbach's alpha was computed using SPSS for the Immersion and Usability constructs. Ideally, this value would be at least 0.8 for both of these scales. The Enjoyableness construct consists of only one question, so Chronbach's alpha is not valid in that case. The Immersion questions have a Chronbach's alpha of 0.748, but it increases to 0.832 if the "2D-3D" question is removed. The Usability questions have a Chronbach's alpha of 0.408, which is increased to 0.809 if the "Challenge" question is removed. The inconsistency of the "Challenge" question is not surprising since, as mentioned above, the desirable value is in the middle of the scale rather than at the top. The inconsistency of the "2D-3D" question comes from an unexpected finding of the study, which will be discussed later along with the main effects.

The Environment Comparison Survey asks participants to compare the two environments tested in the first and second halves, using the same questions as the Environment A & B Surveys shown in Table 17, except for the omission of the last three (Slow-Fast, Control Aiming, and Enough Time), which were deemed irrelevant for comparing the two environments. As before, the Enjoyableness construct consists of a single metric. The Immersion questions have a Chronbach's alpha of 0.829, even *without* omitting the 2D-3D question (which would increase it slightly to 0.835). The Usability construct in this survey is reduced to only two questions due to the omissions mentioned above, and has a Chronbach's alpha of -5.922, which according to SPSS, indicates a violation of the assumptions of the Chronbach reliability model. This is not surprising, due to the aforementioned inconsistency with the "Challenge" question scale, but since there are only two questions remaining, the omission of either results in a single-item scale, for which Chronbach's alpha no longer applies. It is also worth mentioning, however, that the entire Environment

Comparison Survey except for the Challenge question, when taken together as a single scale, has a Chronbach's alpha of 0.873 indicating a general consistency of the survey as a whole.

Based on the inconsistencies of the 3D and Challenge questions with their respective constructs, the dearth of Usability questions in the Environment Comparison Survey, and the fact that Enjoyableness is already represented by only one metric (Fun), the forthcoming analysis is done at the individual question level. The relatively small number of questions makes this just barely possible without the graphs being too unwieldy.

4.8 Survey Results – Main Effects

4.8.1 *Control: Survey Results*

The main effects of the control scheme can be seen using the Environment A & B Surveys. By averaging the scores for each question across both surveys, all participants, and both states of auto-lock, the resulting averages illustrate the differences between the control schemes, as shown in the figure below.

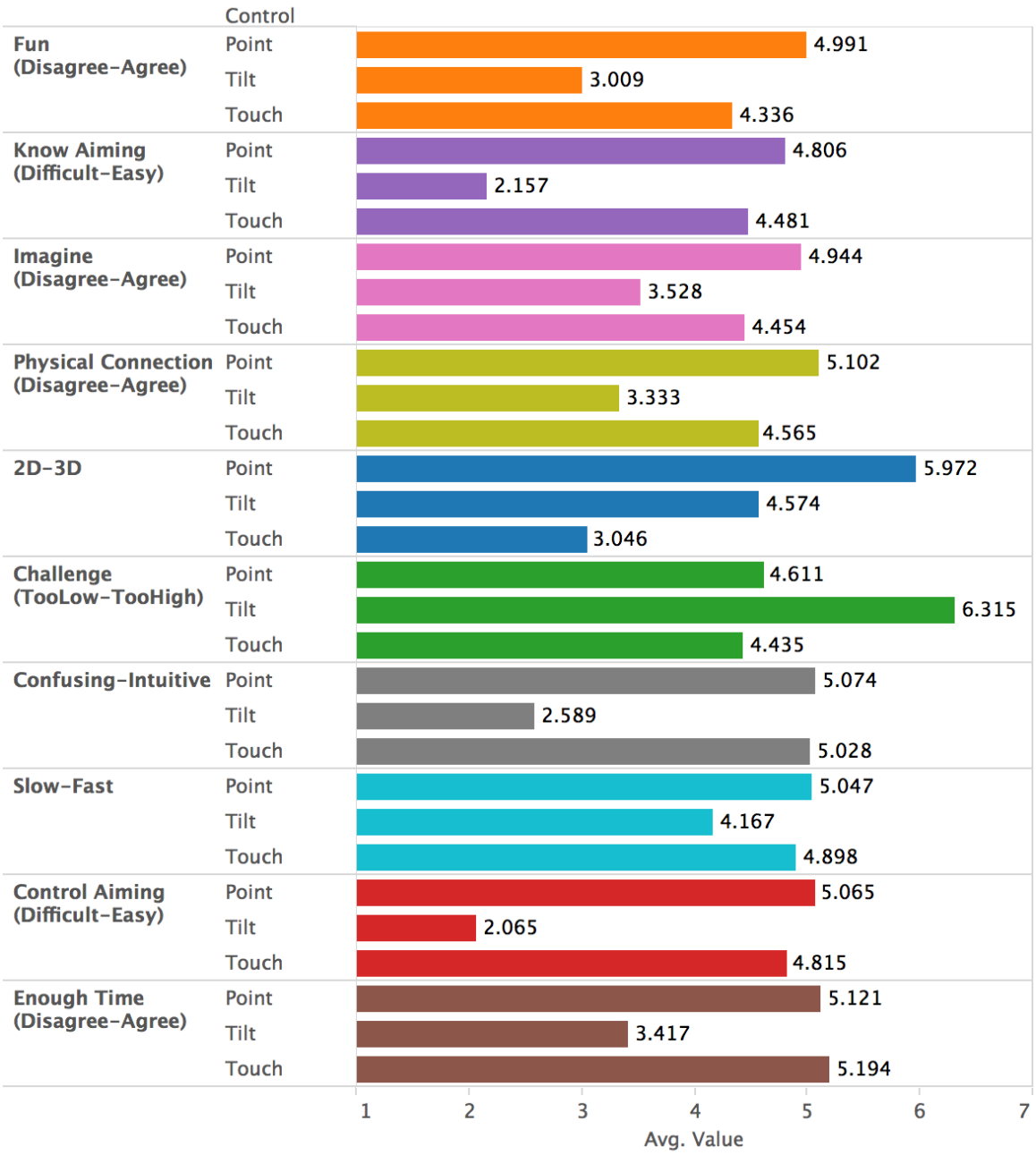

Figure 61: Average Survey Scores vs. Control Scheme

The first thing to notice about the above averages is the general "sideways V" pattern of the results between the different control types within each question. Ignoring the 2D-3D and Challenge questions for now, Point and Touch score higher than Tilt on all metrics. Recalling that the desired value for Challenge is in the center (at 4) rather than all the way to the right, the same is true for

that question as well. The 2D-3D question is a clear anomaly. Users seem to find Tilt less 3-dimensional then Point, but Touch even less so. This discrepancy explains why its removal from the reliability analysis increases Chronbach's alpha, as discussed in section 4.7. The figure below shows the exact same metric averages, but represented as vertical gray lines with their 99% confidence intervals. 99% is used instead of 95% to account for multiple comparisons.

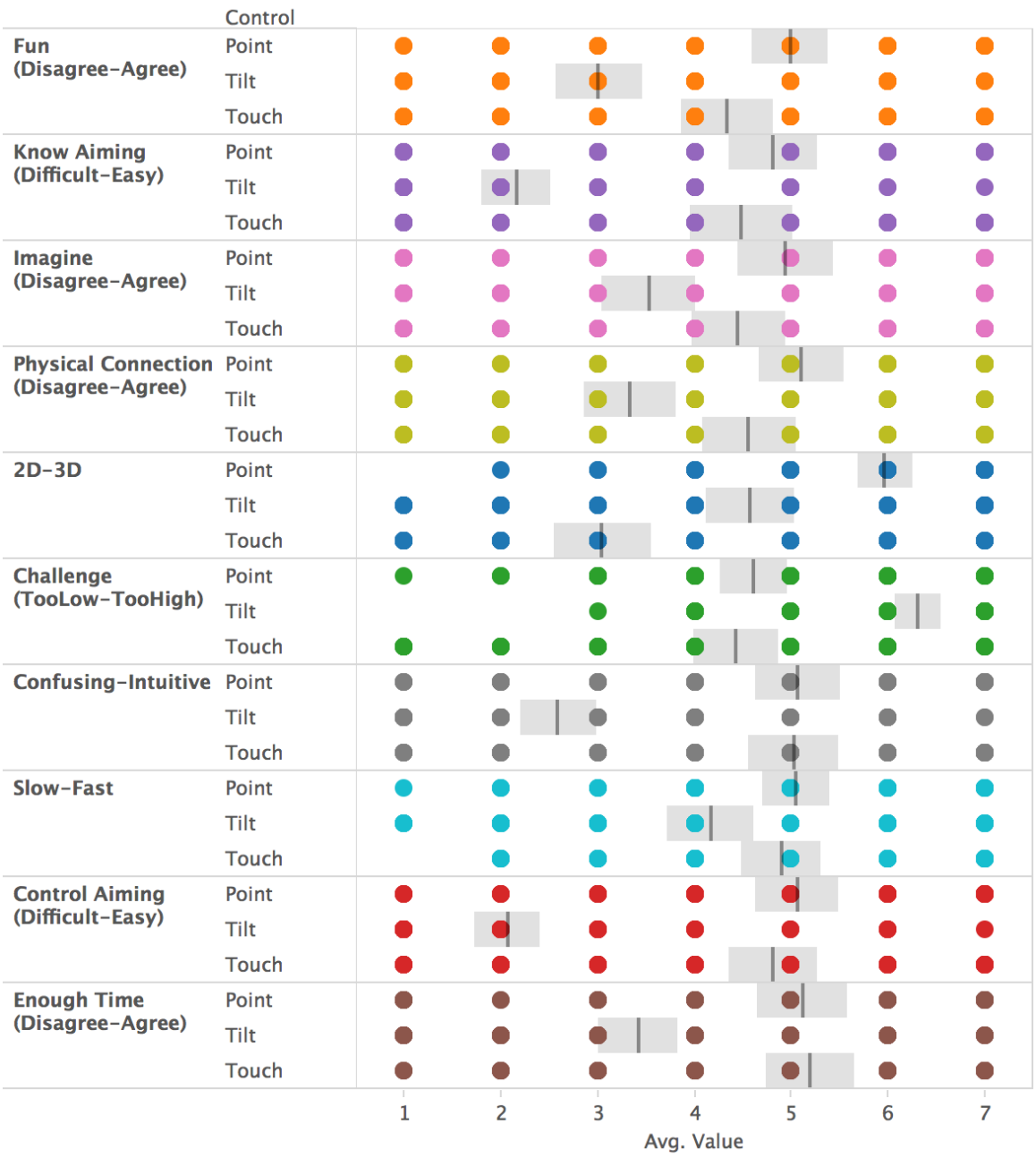

Figure 62: Average Survey Scores vs. Control Scheme (with 99% Confidence Intervals)

Again ignoring 2D-3D, and based on the confidence intervals in Figure 62, the inferiority of Tilt observed in Figure 61 is significant for all questions except Imagine and Slow-Fast, where its interval overlaps slightly with Touch. The 2D-3D anomaly wherein Touch scores lower than both Tilt *and* Point is also statistically significant. Lastly, although Point slightly outperforms Touch on several metrics, none of them are statistically significant.

In general, these results concur with the quantitative data for accuracy discussed in section 4.4.2, which does not bode well for Tilt, but a few caveats are warranted. First, users indicated that they felt less strongly that they had enough time to familiarize themselves with the interface, so it's possible that the Tilt control scheme simply has the steepest learning curve. Second, control schemes could be highly preferential, and it is not the goal of this study to deem one control scheme "better" than another, but to present the average first-brush experience of a group of users, as well as use the variation of this factor as a control to better support the primary hypothesis regarding the environments.

Also of relevance is the figure below, which summarizes users' responses to the question, "My favorite control schemes for the two environments were…" Note that the one null value is from a user who managed to bypass the question.

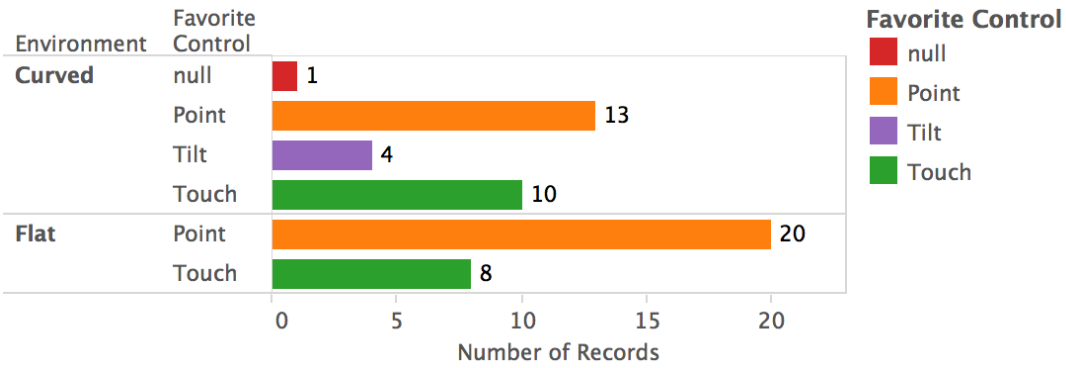

Figure 63: Favorite Control Schemes for Each Environment

While the above figure seems to confirm the general dislike of Tilt, four users still chose it as their favorite in the Curved environment. While it is possible this could have been a mistake, it is also possible that some players simply enjoy the additional challenge it provides, and they may even improve their performance with more time to learn. Lastly, although the differences between Touch and Point were not statistically significant in Figure 62, Figure 63 does seem to indicate a more frequent preference for Point controls over Touch controls in general.

4.8.2 *Auto-Lock: Survey Results*

The main effects of auto-lock can also be seen using the Environment A & B Surveys. By averaging the scores for each question across both surveys, all participants, and all three controls, the resulting averages illustrate the differences between having auto-lock On and Off, as shown in the figure below.

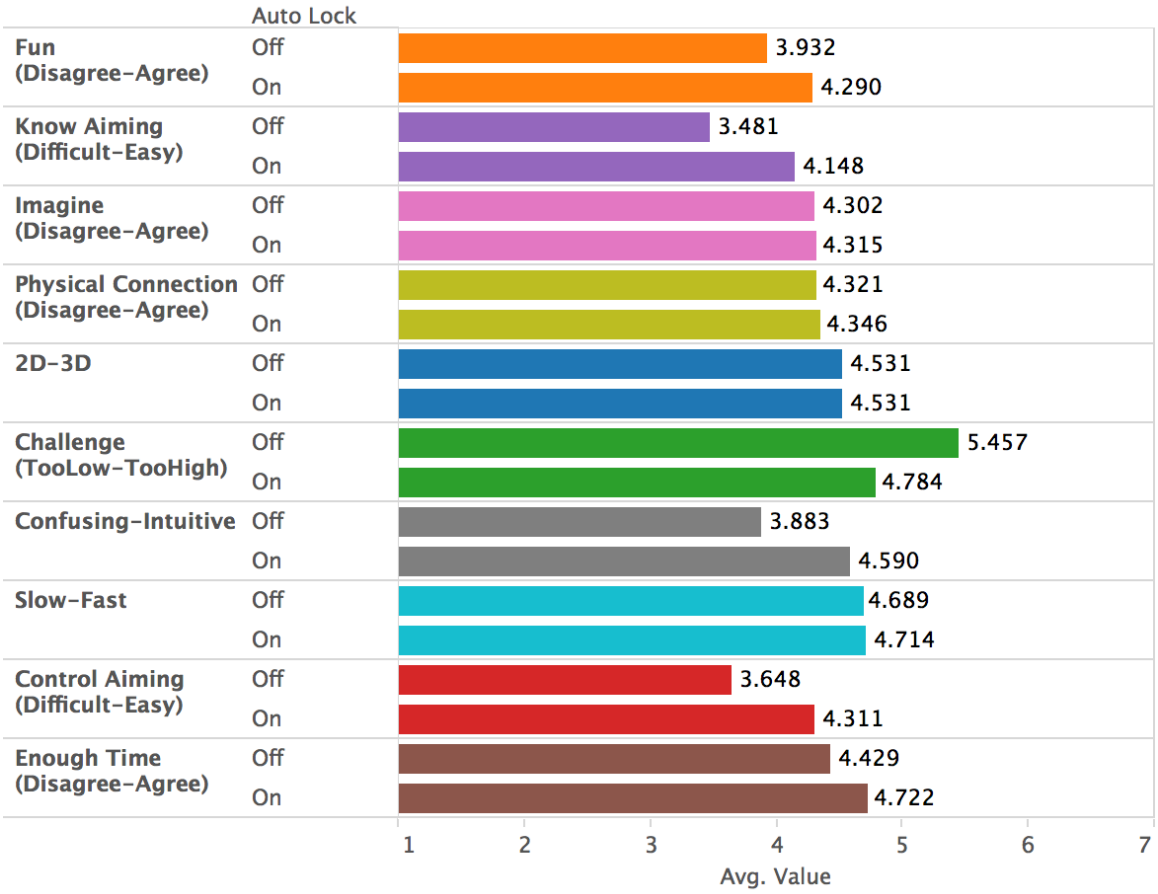

Figure 64: Average Survey Scores vs. Auto-Lock State

Based solely on the averages, it appears that auto-lock scores slightly higher on all metrics (more so on some than others). In order to evaluate whether any of these are statistically significant, the figure below shows the exact same metric averages, but represented as vertical gray lines with their 95% confidence intervals.

Figure 65: Average Survey Scores vs. Auto-Lock State (with 95% Confidence Intervals)

Although it's difficult to see in the figure above on some of the closest cases, auto-lock On showed a significant improvement in four metrics: Know Aiming, Control Aiming, Confusing-Intuitive, and Challenge (recall that the desired value for Challenge is in the middle of the scale rather than to the right). Consequently, we can conclude that auto-lock increased users' feeling that they both knew and could control where they were aiming, increased the intuitiveness of the interface, and reduced the challenge to a more appropriate level. This represents a statistically significant increase in 3 of the 5 Usability metrics without impacting Immersion, except in one case, where it was actually increased (in the "Know Aiming" metric).

4.8.3 *Environment: Survey Results*

The main effects of the environment can be observed in two ways. First, the scores for each question within the Environment A & B Surveys can be averaged (across all three control schemes and both states of auto-lock) and compared. Second, the Environment Comparison Survey asks most of the same questions, but explicitly asks users to compare Environments A & B (Curved and Flat, respectively), so those questions can simply be averaged across all users. The resulting averages for both approaches are shown in the two figures below.

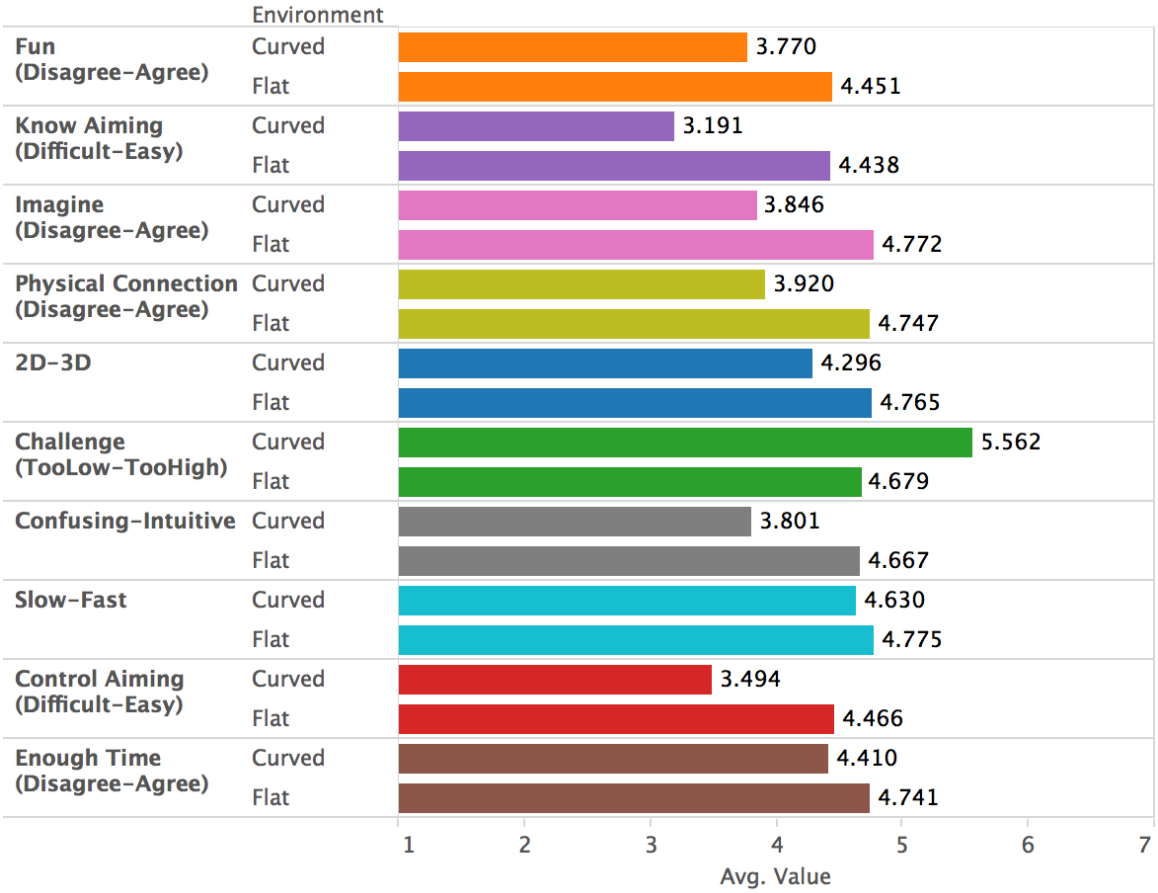

Figure 66: Average Environment A/B Survey Scores vs. Environment

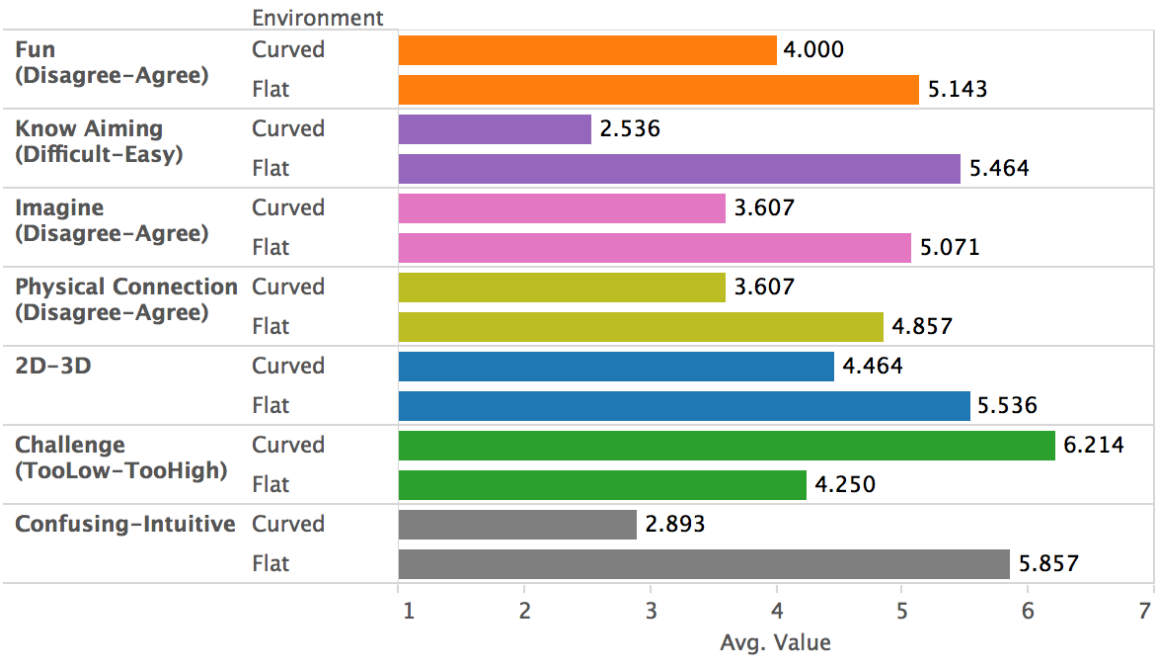

Figure 67: Average Environment Comparison Survey Scores vs. Environment

The first thing to notice is that the Flat environment outperforms the Curved environment for all metrics on both surveys (again, keeping in mind that the desired value for the "Challenge" question is in the middle of the scale rather than to the right). The second thing to notice is that for questions that are common to both/all surveys, the spread between the averages for the two environments is greater on the Comparison Survey than between the Environment A/B Surveys for the questions that are on both. Before jumping to any conclusions, however, the confidence intervals must be considered. The two figures below show the exact same metric averages for the two surveys, but represented as vertical gray lines with their 95% confidence intervals.

Figure 68: Average Environment A/B Survey Scores vs. Environment

(with 95% Confidence Intervals)

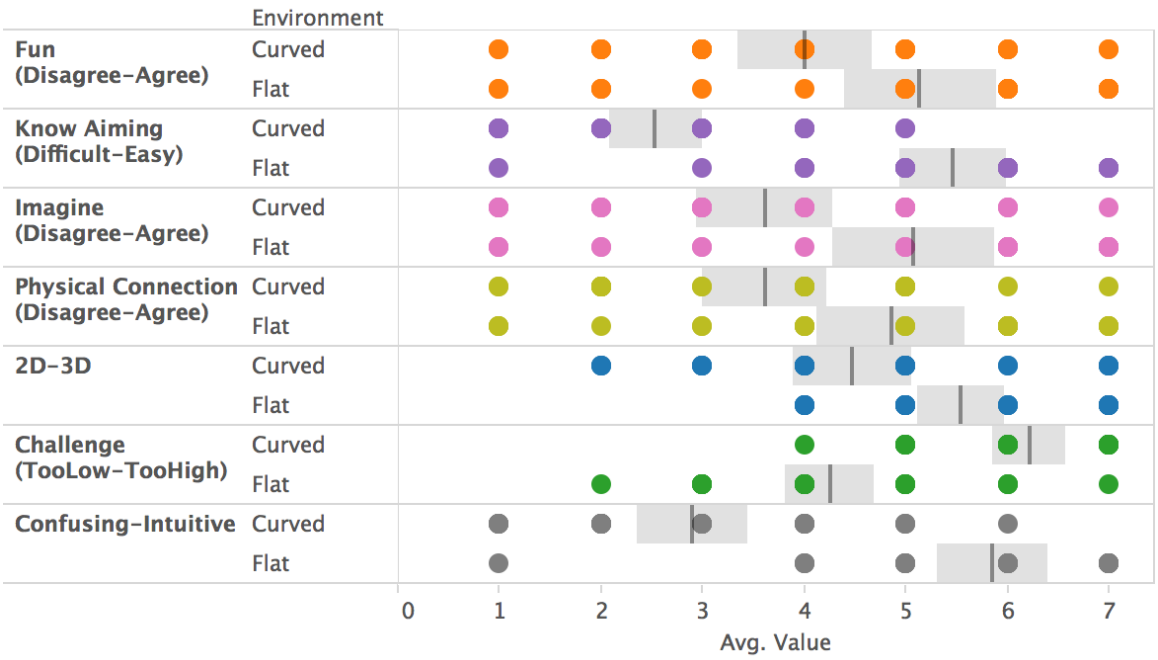

Figure 69: Average Environment Comparison Survey Scores vs. Environment (with 95% Confidence Intervals)

In comparing the confidence intervals of the two figures above, an unfortunate trend is revealed. Although the spread between the averages is greater on the Environment Comparison Survey than the Environment A/B Surveys, the confidence intervals are significantly larger. This is due to the fact that the Environment A/B Surveys have the benefit of 6 data points per question per environment, since they ask about all three control schemes with and without auto-lock, while the Environment Comparison Survey only has one data point per question. Thus, despite the larger difference between averages, fewer metrics show a statistically significant difference on the comparison survey. Rather than step through each metric in both figures to compare the significance results, the table below summarizes which differences are significant using which surveys.

Table 18: Significance of Flat Environment Improvement
using Environment A/B vs. Environment Comparison Surveys

Survey Metric	Environment A/B Surveys	Environment Comparison Survey
Fun	Significant	Not Significant
Know Aiming	Significant	Significant
Imagine	Significant	Not Significant
Physical Connection	Significant	Not Significant
2D-3D	Not Significant	Significant
Challenge	Significant	Significant
Confusing-Intuitive	Significant	Significant
Slow-Fast	Not Significant	N/A
Control Aiming	Significant	N/A
Enough Time	Not Significant	N/A

Note that Table 18 does not indicate anything about which environment showed improvement in the given metric. The Flat environment showed an improvement over the Curved environment on all metrics on all surveys. Rather, the table shows whether this effect is statistically significant using the different surveys. Note also that, although the three questions unique to the Environment A/B Surveys were not expected to show any difference between the two environments (which was the reason for their absence on the comparison survey), the results show that users had a significantly easier time controlling where they were aiming when using the Flat environment. The improvements to the metrics that are common to both surveys (which are the metrics deemed relevant for evaluating the environments), were statistically significant on one or both of those surveys.

Chapter 5. CONCLUSIONS

5.1 SUMMARY OF FINDINGS

5.1.1 *Environment*

The primary hypothesis of this study was that the Flat environment would offer objective performance improvements over the Curved environment, with the knowledge that subjective metrics such as Enjoyableness and Immersion may suffer, as measured by user surveys. This primary hypothesis held true in this experiment, and the latter fears were shown to be unwarranted.

The Flat environment improved accuracy by a significant margin in both the X and Y directions (see section 4.4.1). When using Point or Touch controls, the Flat environment allowed users to successfully hit over a third of the targets without the help of auto-lock (see Figure 47). The Flat environment also offers a significant increase in speed (see section 4.6.2), dropping the average time for a successful hit from 14.92s to 11.84s. Furthermore, all survey metrics showed an average improvement in the Flat environment over the Curved environment (see section 4.8.3). Of the metrics which were common to both the Environment A/B Survey and the Environment Comparison Survey, all improvements were shown to be significant on one or both surveys.

5.1.2 *Control*

Three types of controls are used in this study in order to help account for the variety of options that could potentially be used in actual games. This serves primarily as a control to enhance the external validity of the primary hypothesis regarding the two environments, with the expected secondary effect that performance will vary among the three control types.

Of the three control types, Tilt performed by far the worst, showing a significant reduction in accuracy compared to Point or Touch, while Point and Touch controls showed no significant

difference in accuracy (see section 4.4.2). Tilt was also slower on average for successful hits, but this was deemed statistically insignificant, due to the fact that the low number of hits achieved with Tilt in the first place widened its confidence interval. Interestingly, Point controls were shown to be significantly faster than Touch, dropping the average time for a successful hit from 13.69 seconds to 11.60 seconds (see section 4.6.3).

On the user surveys, with the exception of the "2D-3D" question, Tilt scored lower than Point and Touch on all metrics (see section 4.8.1). Of these, all but two were statistically significant. Point scored slightly higher than Touch on average for most metrics, but none were statistically significant. The 2D-3D question, when grouped by control scheme, remains an anomaly. Tilt still scored lower than Point, but unlike other metrics, Touch scored by far the lowest, indicating it felt the most 2-Dimensional. All of these differences in perceived dimensionality were statistically significant. This association of Touch with 2D could be due to the fact that users interact directly with the coordinates of the real screen, which is of course 2-Dimensional. The fact that Touch scored as well as Point on all other metrics indicates that its perceived 2-Dimensionality may or may not be a bad thing, but it is certainly worth noting.

5.1.3 *Auto-Lock*

Auto-lock was introduced as a mechanic to provide an opportunity for the Curved environment to be improved through additional feedback, and although this was certainly the case, the data show that auto-lock improves accuracy to a statistically significant degree in general (see section 4.4.3). The interaction effects between environment and auto-lock for average distance make it appear that auto-lock had no effect on the Flat environment, but they also show that the Flat environment already achieves a smaller average distance without auto-lock than the Curved environment does *with* auto-lock (see section 4.5.3, Figure 51, left side). However, despite its failure to affect the

average distance in the Flat environment, auto-lock does still improve the hit/miss percentages, presumably due to its ability to turn near-misses into hits (see Figure 51, right side).

No statistically significant difference in speed was found between tests with auto-lock and tests without it (see section 4.6.4). Survey results show that it improved the averages on all survey metrics, but the difference is only significant on 4 out of the 10 questions. Users indicated that auto-lock decreased the challenge (to a more appropriate level), made the interface more intuitive, and helped them to both know and control where they were aiming (see section 4.8.2).

5.2 Recommendations and Future Work

Based on the findings shown here, it appears that the Flat environment is superior than the Curved environment for short-term play with single static targets, as it performs significantly better on all metrics. However, it is important to remember that the learning curve was not observable with the test duration used in this study, so future work should examine this to see if it holds true under longer periods of play. Both the Point and Touch control schemes are viable input methods, although Point looks particularly promising due to the fact that it increases speed and would free up the screen for other actions. Although Tilt appears to suffer significantly, rejecting it outright should be done with extreme care. Although the sensitivity was fixed and the feedback mechanisms kept identical to the other two controls for a fair comparison, customization of the former and modification of the latter could possibly improve its performance. For example, adding a scraping sound when users are moving along a wall, somehow indicating *which* wall they are stuck on, and/or adding an ability to re-center themselves on the screen might help with general orientation and reduce frustration. Keep in mind also that the tests performed here are too short to observe the learning curve of each configuration over time. Most importantly, some advanced gamers may enjoy the additional challenge, perhaps once mastering the easier options. Although

no one chose Tilt as their favorite control in the Flat environment, 4 people chose it as their favorite in the Curved environment, which, barring user error, disproves the unanimous distaste for it altogether.

Auto-lock appears to offer both objective accuracy and subjective usability improvements, although no conclusive effect on speed was shown. Although it would seem there is thus no drawback to including it in a game, at least as an option, it may prove problematic in future works if multiple targets and additional audio feedback are added, due to the inevitable confusion of paying attention to multiple audio cues simultaneously.

The most obvious next step for this paradigm would be to do a similar study with blind participants, rather than blindfolded sighted participants, to confirm the external validity of the superiority of the Flat environment. Performing longer tests to observe the long-term learning curves of the three control types may also be worthwhile. Further development of the application of this paradigm should include moving targets and/or multiple simultaneous targets. The former will further stress the complex interaction capabilities of this model, while the latter will introduce the problem of discerning multiple audio sources while maintaining reasonable localization performance. One possible solution to the multiple-source problem would be to also use multiple target types with unique individualized sounds, with the optional additional mechanic of restricting specific target types to specific areas of the screen. This would allow the sound itself to roughly signify its general area, while the enhanced 3D audio cues of the Flat environment can then hopefully provide the fine-grained accuracy necessary to land an accurate shot.

The most immediate next step for this research is to pursue publication of these results in a more condensed form. Unsighted gaming remains an underexplored area of research that benefits an overlooked demographic, and it is my hope that the paradigm developed here will allow other

researchers to use it in tackling more complex scenarios and eventually, actual games. Collaboration benefits all of us, and hopefully the concepts explored here will also spawn unexpected applications and modifications of this approach by others, to the benefit of this underserved community.

BIBLIOGRAPHY

AGRIP AudioQuake and Level Description Language (LDL). (n.d.). Retrieved March 5, 2015, from https://github.com/matatk/agrip

Allman, T., Dhillon, R. K., Landau, M. A. E., & Kurniawan, S. H. (2009). Rock Vibe: Rock Band; Computer Games for People with No or Limited Vision. In *Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 51–58). New York, NY, USA: ACM. http://doi.org/10.1145/1639642.1639653

Atkinson, M. T., Gucukoglu, S., Machin, C. H. C., & Lawrence, A. E. (2006). Making the Mainstream Accessible: Redefining the Game. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames* (pp. 21–28). New York, NY, USA: ACM. http://doi.org/10.1145/1183316.1183321

AVAudio3DMixing Protocol Reference. (n.d.). Retrieved March 10, 2015, from https://developer.apple.com/library/prerelease/ios/documentation/AVFoundation/Reference/AVAudio3DMixing_Protocol/index.html#//apple_ref/c/tdef/AVAudio3DMixingRenderingAlgorithm

AVAudioEnvironmentDistanceAttenuationParameters. (2014, September 17). Retrieved February 27, 2016, from https://developer.apple.com/library/prerelease/ios/documentation/AVFoundation/Reference/AVAudioEnvironmentDistanceAttenuationParameters_Class/

Balan, O., Moldoveanu, A., Moldoveanu, F., & Dascalu, M.-I. (2014). Navigational 3D audio-based game-training towards rich auditory spatial representation of the environment. In *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference* (pp. 682–687). http://doi.org/10.1109/ICSTCC.2014.6982496

Blauert, J. (1997). *Spatial Hearing : The Psychophysics of Human Sound Localization*. Cambridge, Mass: MIT Press. Retrieved from http://offcampus.lib.washington.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=24381&site=ehost-live

Bronkhorst, A. W. (1995). Localization of real and virtual sound sources. *The Journal of the Acoustical Society of America*, *98*(5), 2542–2553. http://doi.org/10.1121/1.413219

BSD Library Functions Manual: ARC4RANDOM(3). (1997, April 15). Retrieved March 1, 2016, from https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man3/arc4random_uniform.3.html

Cheiran, J. F. P., Nedel, L., & Pimenta, M. S. (2011). Inclusive Games: A Multimodal Experience for Blind Players. In *2011 Brazilian Symposium on Games and Digital Entertainment (SBGAMES)* (pp. 164–172). http://doi.org/10.1109/SBGAMES.2011.24

iOS and Android Audio Latency Test App. (2015, February 12). Retrieved February 26, 2016, from http://superpowered.com/latency

iOS Human Interface Guidelines. (2015, November 5). Retrieved February 27, 2016, from https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/LayoutandAppearance.html#//apple_ref/doc/uid/TP40006556-CH54-SW1

kung fu punch.aif. (2009). Retrieved from http://freesound.org/people/nextmaking/sounds/86019/

Liam Erven. (n.d.-a). LWorks. Retrieved January 24, 2015, from http://www.l-works.net/

Liam Erven. (n.d.-b). LWorks - Zompocalypse. Retrieved March 6, 2015, from http://www.l-works.net/zompocalypse.php

Morelli, T., Foley, J., Columna, L., Lieberman, L., & Folmer, E. (2010). VI-Tennis: A Vibrotactile/Audio Exergame for Players Who Are Visually Impaired. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games* (pp. 147–154). New York, NY, USA: ACM. http://doi.org/10.1145/1822348.1822368

Morelli, T., Foley, J., & Folmer, E. (2010). Vi-bowling: A Tactile Spatial Exergame for Individuals with Visual Impairments. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 179–186). New York, NY, USA: ACM. http://doi.org/10.1145/1878803.1878836

Morelli, T., & Folmer, E. (2011). Real-time Sensory Substitution to Enable Players Who Are Blind to Play Video Games Using Whole Body Gestures. In *Proceedings of the 6th International Conference on Foundations of Digital Games* (pp. 147–153). New York, NY, USA: ACM. http://doi.org/10.1145/2159365.2159385

Östblad, P. A., Engström, H., Brusk, J., Backlund, P., & Wilhelmsson, U. (2014). Inclusive Game Design: Audio Interface in a Graphical Adventure Game. In *Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound* (pp. 29:1–29:8). New York, NY, USA: ACM. http://doi.org/10.1145/2636879.2636909

Starch, D. (1908). *Perimetry of the Localization of Sound* ... State University of Iowa.

Statistical Facts about Blindness in the United States. (2015). National Federation of the Blind. Retrieved from https://nfb.org/blindness-statistics

SWOSH-01 44.1kHz.flac. (2008). Retrieved from http://freesound.org/people/qubodup/sounds/59988/

The AGRIP Project. (n.d.). Retrieved March 5, 2015, from http://agrip.org.uk/

Thuds on Window. (2014). Retrieved from http://freesound.org/people/zerolagtime/sounds/245033/

Understanding Accessibility on iOS. (n.d.). Retrieved February 17, 2015, from https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/iPhoneAccessibility/Accessibility_on_iPhone/Accessibility_on_iPhone.html#//apple_ref/doc/uid/TP40008785-CH100-SW1

Yuan, B., & Folmer, E. (2008). Blind Hero: Enabling Guitar Hero for the Visually Impaired. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 169–176). New York, NY, USA: ACM. http://doi.org/10.1145/1414471.1414503