Harnessing the Capacity of the Human Voice for Fluidly Controlling Computer Interfaces

Susumu Harada

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2010

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Susumu Harada

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

_____
James A. Landay

_____
Jacob O. Wobbrock

Reading Committee:

_____
James A. Landay

_____
Jacob O. Wobbrock

_____
Jeffrey A. Bilmes

Date:_____

Signature _____

Date _____

University of Washington

Abstract

Harnessing the Capacity of the Human Voice for Fluidly Controlling Computer Interfaces

Susumu Harada

Co-Chairs of the Supervisory Committee:

Associate Professor James A. Landay
Computer Science and Engineering

Assistant Professor Jacob O. Wobbrock
Information School

The human voice still remains largely unexploited in the general computing domain as a primary or supplementary input modality despite significant advances in speech recognition technology over many years. While voice input has a number of potential benefits, especially for people with motor impairments, one of the major limitations of current speech-based interaction methods is their inability to provide fluid and continuous input, akin to pointing devices such as the mouse. In this dissertation, I describe work that I have conducted as part of the Vocal Joystick project to harness the non-speech characteristics of human vocalization to enable such fluid hands-free control using the voice.

The Vocal Joystick engine upon which my work is based converts various non-speech vocal features such as volume, pitch, and vowel quality into continuous as well as discrete signals 100 times a second. I conducted a number of user studies to determine the performance characteristics of using such vocal input for various tasks including target selection, steering, and rapid discrete inputs. A key result shows that with only 10 hours of practice, users can approach the level of performance comparable to a manual joystick for pointing, and exceed the performance of an existing speech-based pointer control method.

I also present a number of concrete voice-driven applications that I built and evaluated to situate the study of non-speech vocal input in realistic contexts. VoiceDraw is a hands-free drawing program that enables the user to create free-hand-style drawings with dynamically-controlled stroke thickness and speed using only their voice. VoiceGame Controller rapidly transforms various vocal signals into keyboard and mouse signals, making it possible to play conventional computer games hands-free. VoicePen augments digital stylus input with continuous voice input. Voice Controller seamlessly integrates voice-driven pointer control with conventional speech commands and dictation functionality, creating a powerful and practical hands-free input modality.

The user studies and the applications I have built demonstrate my thesis, that:

> *Non-speech vocal input can be used on its own and in conjunction with other input modalities to enable people—especially those with motor disabilities—to control computer interfaces effectively.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to extend my gratitude to my advisors, James Landay and Jacob Wobbrock, for their invaluable advice and mentorship throughout the course of my dissertation work. I could not have asked for a better pair of advisors. Thank you for believing in me, pushing me, and encouraging me to follow my dreams.

I would also like to thank the other members of my committee for their support and guidance. I would especially like to thank Richard Ladner, whose pursuit of socially impactful research I deeply admire and whose advice and mentorship have fueled my passion towards accessibility research. I also thank Jeff Bilmes for not only his compassionate and patient approach to his students but for having conceived the original Vocal Joystick concept, without which my work would not exist. Axel Roesler helped balance out the committee by bringing his expertise in both engineering as well as design, and providing me with a number of fresh perspectives I would otherwise not have considered.

This work has been supported by a number of sponsors who deserve credit and thanks: the National Science Foundation, NISH, University of Washington Computer Science and Engineering Departmental Fellowship, and the University of Washington Royalty Research Fund.

I could not have accomplished the work in my dissertation without the help, advice, and encouragement from a large number of kind and talented people. In alphabetical order, I would especially like to thank Jeff Bigham, Anna Cavender, Eun Kyoung Choe, Seth Cooper, Kate Everitt, Jon Froehlich, Krzysztof Gajos, Sangyun Hahn, Thomas Isdal, Alex Jaffe, Shani Jayant, John P. John, Tim Kao, Nodira Khoussainova, Travis Kriplean, Michelle Leber, Toiwa Lee, Yongjoon Lee, Jonathan Lester Xiao Li, Jonathan Malkin, Tim Paek, Kayur Patel, Michael Piatek, Maria Rowley, Eva Ringstrom, T. Scott Saponas, Fay Shaw, Michael Toomim, and Deepak Verma. I am truly grateful for your friendship and support.

The tough times were made easier thanks to the opportunities I have had to refresh my mind at karate practices. I extend my heartfelt thanks to the Shotokan Karate club at the University of Washington, as well as all the seniors and members of Shotokan Karate of America for providing that opportunity and teaching me how to persevere and push through any obstacle.

The Tsubaki Shrine of America in Granite Falls, Washington, was my lifeline when everything seemed to be slipping away. Thank you so much Barish sensei, Chika san, and the other Tsubakiko members for helping me recharge when I most needed to.

Finally I would like to thank my family, my parents and my sister, for all they've done to help me get to where I am. I am truly grateful for your love and patience.

DEDICATION

To those who continue to strive
regardless of their abilities or circumstances.

Chapter 1
Introduction

1.1 Motivation

There has been a great amount of research poured into speech recognition technology over the past several decades, with recent significant improvements in recognition accuracy and capability (Karat et al. 2003). Applications of speech-based interaction in areas such as interactive voice response (IVR) systems for automated call centers and telephone services have become ubiquitous. Adoption rates in the medical field, especially among physicians and radiologists who use it for dictating documents into electronic medical record (EMR) systems, have been increasing steadily as the quality of dictation engines continue to improve.[1] With commercial products such as Dragon Naturally Speaking from Nuance Communications, Inc.,[2] and Windows Speech Recognizer, which has shipped as part of the Windows operating systems since Vista,[3] the prospect of using speech interaction in the realm of general computing is continuing to increase. The low cost of the hardware required for speech interaction—a microphone—also makes it an attractive solution for hands-free computer access.

[1] http://www.reuters.com/article/pressRelease/idUS147152+07-Aug-2008+BW20080807
[2] http://www.nuance.com/naturallyspeaking/
[3] http://www.microsoft.com/enable/products/windowsvista/speech.aspx

The exploration of how to enhance the *expressivity* of voice-based interaction is important for a number of reasons. For people with physical impairments that limit the use of their hands for using a mouse or a keyboard, hands-free input methods such as voice input may be their only option for gaining access to the computer. In the United States alone, there are over 700,000 people with disabilities of the spinal cord, 70% of whom are unemployed.[4] For these individuals with limited mobility and motor control, access to a computer may be one of the few options available to them for achieving greater independence, obtaining or retaining employment, staying connected with people and information around them, and expressing themselves creatively (Warren 1997). These issues extend to people with other motor impairments as well, including the 46 million adults in the United States diagnosed with arthritis, the 1 million with Parkinson's disease, and the 50,000 children and adults with muscular dystrophy.[5]

The benefits of enhanced voice-based interaction can also extend to people without motor impairments who find themselves in impairing situations. People may experience situational impairments (Sears & Young 2003), for example while driving or interacting with a wall-sized display, during which a hands-free interaction may be more suitable than traditional manual input devices. On the desktop, voice input can serve to augment the standard keyboard and mouse interaction to provide an additional input modality for greater control, especially in applications that demand multiple dimensions and simultaneous channels of input such as computer aided design tools.

However, speech input, or more generally voice-based input, has yet to become a mainstream computer input modality (Deng & Huang 2004). Some critiques of speech recognition applications have pointed out several barriers to adoption, including inappropriateness in shared environments and the cognitive interference between speech production and problem solving (Shneiderman 2000). These are certainly worthy concerns that need to be considered, but are not insurmountable. There have been various research efforts attempting to bring speech interaction into the desktop and general computing environment (Manaris et al. 2001; Rudnicky et al. 1991; Schmandt et al. 1990a, 1990b). Some of this work has made it into commercial systems, but the adoption rate remains low (Deng & Huang 2004), even among people with physical disabilities who may benefit from such hands-free input technology (Koester 2003).

[4] http://unitedspinal.org/pdf/scd%20fact%20sheet.pdf
[5] http://www.hmc.psu.edu/healthinfo/

One of the major limitations of the current voice-based input modality on personal computers is its inability to specify *continuous* input parameters, akin to manual pointing devices such as the mouse. Traditional speech recognition systems process user's vocal utterances at the word level, resulting in an inherently discrete interaction. For example, the user utters a command phrase, the system recognizes the words, and performs the corresponding action. Similarly, dictation systems generate recognized text output with a delay of several words to several phrases. While these methods work well for executing commands and entering textual information, they are not amenable to, if not incapable of, specifying continuous inputs that are needed in tasks such as drawing, tracing, steering, panning, scrolling, zooming, and so forth.

A key component that is missing in today's speech-based input technology is the analogue to *direct manipulation* that has made the mouse such a successful input device. Direct manipulation—a term introduced by Shneiderman (1983)—describes a style of human-computer interaction that has the following three properties:

1. Continuous representation of the object of interest;
2. Physical actions instead of complex syntax; and
3. Rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.

The third property in particular is what is missing from speech-based input technology, as it is often difficult to rapidly and incrementally execute operations using spoken commands.

For speech to become a fully functional modality for operating the typical personal computer or smartphone today, several key functions need to be supported:

- Text entry: ability to input textual information quickly and accurately.
- Commanding: ability to execute all commands available on a system.
- Direct manipulation: ability to manipulate objects and perform mouse-like operations fluidly.

Much research has been poured into supporting the first two functions through advancements in automated speech recognition (ASR). Commercial products such as Dragon NaturallySpeaking and Windows Speech Recognizer provide cross-application operating system-wide support for text entry and issuing commands on commodity personal computers, although there remains a significant portion of operating system and application functionalities that remain inaccessible

through speech commands. The third functionality of direct manipulation via speech is still virtually unaddressed.

This current state of speech-based input may be analogous to a user being given only a miniature keyboard and no mouse for providing input into a computer. The user may be able to enter text with relatively acceptable accuracy with occasional errors. If the user knows keyboard shortcuts, he may be able to access various menu items and issue commands, and to switch between applications. He may also be able to control the mouse pointer using the arrow keys to move the pointer around in the four cardinal directions, with constant or possibly incrementally variable speed. With such a setup, performing mainly text-entry oriented tasks such as composing email messages or editing documents may be feasible, but other common tasks that typically demand the use of a mouse may be extremely difficult or nearly impossible, such as drawing or creating diagrams, manipulating a scrollable or zoomable interface, and so on.

Given these potential benefits of speech-driven input and their current shortcomings, the need to explore ways to enhance voice-based computer interaction can be translated into the following high-level goals, each with a different emphasis on the problem space it seeks to address.

1. *Make interactions with existing computer applications possible or more accessible for users with motor impairments*.

   The emphasis of this goal is more on the practical needs of users with motor impairments who need to be able to access existing computer operating systems and applications, which have been designed for keyboard and mouse input, using their voice as a primary hands-free modality. As such, it may not provide the ideal solution from the interaction design perspective since the voice input will have to be retrofitted to work with the idiosyncrasies of user interfaces that expect manual pointing devices and keyboards. However, even a non-optimal solution will still benefit the target users more than the alternative of not being able to access the functionality at all.

2. *Make interactions with existing computer applications more effective for general users*.

   Here, the focus is on using voice as an augmentative modality to increase the effectiveness of interacting with traditional applications by supplying an input source parallel to the keyboard and mouse. As prior research on multimodal interactions has

demonstrated the effective integration of speech input with other modalities such as the digital stylus (Oviatt 2003), further improvements may be possible by enhancing the effectiveness of the voice modality.

3. *Design new interfaces and application environments optimized for voice-based control*.

   As an extension to the first two goals, this goal seeks the ideal scenario in which voice input is considered to be a first-class citizen in the input modality space, and in which user interfaces can be made optimal for voice-driven interaction. As Schmandt points out, however, "it is all too easy to be overly optimistic and predict voice as part of every desktop application" (Schmandt 1993 p. 252). Care must be taken to consider both the merits and the limitations of voice input to identify the appropriate design for a voice-centric interface. In light of the principles of Universal Design (Mace et al. 1991), such an interface has the potential to benefit not only people with motor impairments, but general computer users as well if it succeeds in enabling effective hands-free interaction.

I propose that the key to attaining the goals outlined above is to harness the expressive properties of human voice *beyond spoken words*, using such non-speech vocal features to enable fluid and continuous control of the interface previously not possible via voice. Non-speech vocal features refer to acoustic characteristics of human vocalization such as pitch, volume, and vowel quality (i.e., likeness of a given sound to a particular vowel). From the perspective of recognition by a computer, non-speech vocal features can be captured, processed, and classified almost immediately due to the extremely short fundamental "unit" for such features, which can typically be on the order of tens of milliseconds. From the perspective of the user, non-speech vocal features can be produced and varied smoothly and continuously, and may even be modulated in parallel to spoken word utterances. These properties make non-speech vocal features an ideal candidate for bringing to voice-driven interaction the fluidity and continuity essential for realizing the goals enumerated above. In this dissertation, I present the work I have conducted along the three goals listed above, and demonstrate the following thesis:

> *Non-speech vocal input can be used on its own and in conjunction with other input modalities to enable people—especially those with motor disabilities—to control computer interfaces effectively.*

1.2 The Vocal Joystick Concept

At the foundation of my work is the Vocal Joystick engine (Bilmes et al. 2006) which has been developed as part of the Vocal Joystick project at the University of Washington. The Vocal Joystick engine enables audio signal processing to track non-speech vocal features including pitch, volume, and vowel quality in real time. The core Vocal Joystick application enables the user to control the mouse pointer smoothly and continuously by vocalizing various vowel sounds corresponding to the desired direction of movement. Figure 1-1 shows a high-level example of the difference between conventional speech-driven pointer control and Vocal Joystick pointer control. Under conventional speech-driven pointer control, due to the fact that spoken commands are inherently discrete, parameters such as pointer speed and direction can only be modified in discontinuous steps. The rate at which such parameters can be changed is also limited by the speed at which each command phrase can be uttered and recognized by the speech recognizer. Under the Vocal Joystick pointer control, continuously-variable vocal feature such as volume is mapped to pointer speed, enabling real-time and smooth adjustments. Movement direction is specified by vowel sound vocalizations, in which different vowel sounds are mapped to each of the eight cardinal and ordinal compass directions. By varying the vowel sound and the volume continuously, the pointer's movement direction and speed can be smoothly controlled using the Vocal Joystick pointer control. Further details of the Vocal Joystick pointer control and the underlying recognition engine are provided in Chapter 3. My work combines the functionality offered by the Vocal Joystick engine with traditional speech recognition engines to explore ways to extend the capability of voice-based interaction with new user interface technologies.

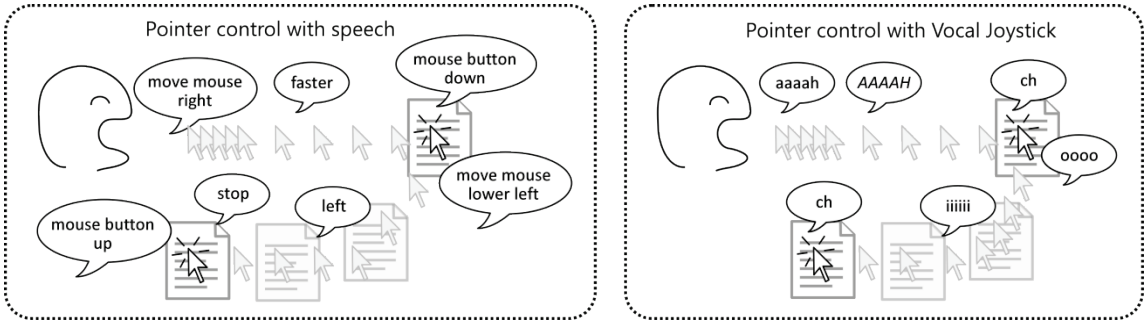


Figure 1-1: Example of the difference between speech-driven pointer control and Vocal Joystick pointer control. Speech-driven pointer control typically results in slow and jerky pointer movements due to the discrete nature of spoken commands. Vocal Joystick pointer control enables more fluid pointer movements by mapping continuously-variable vocal features such as volume and vowel quality to pointer speed and movement direction.

1.3 Research Approach

I have structured the research approach into the following three phases:

1. *Understand* the performance characteristics of non-speech vocalization as an input modality;
2. *Build* concrete applications to investigate the effectiveness of non-speech vocalization as an input modality; and
3. *Generalize* reusable building blocks for creating future voice-driven applications.

Understand the performance characteristics of non-speech vocalization as an input modality

Research into the use of non-speech vocalization for computer interaction is still relatively new. To understand what is possible through the use of non-speech vocalization, we need to first evaluate its basic performance characteristics at the elemental level within the context of fundamental tasks such as pointing and steering. Research questions posed during this phase are:

- How well can people perform basic user interface interactions using various non-speech attributes of their voice?
- What mappings between the acoustic signals and the corresponding effect on the user interface lead to better performance?
- How long does it take a person to learn to perform fundamental tasks such as pointing using various non-speech attributes of the voice?

Build concrete applications to investigate the effectiveness of non-speech vocalization as an input modality

Once we gain insight into the basic capabilities of non-speech vocalization as computer input, we need to investigate the viability of using it for performing real tasks within the context of actual applications. By creating custom applications that leverage voice input as well as by developing methods for interfacing voice input with existing applications, we can design and examine what voice-based interaction techniques beyond the most primitive user interface controls are most effective at facilitating the tasks at hand. The pertinent research questions during this phase are:

- What applications can be made to support hands-free control using just one's voice (both speech and non-speech)?

- What applications can benefit from having voice input (in particular non-speech continuous input) as an augmentative modality?

Generalize reusable building blocks for creating future voice-driven applications

Given the experience gained from building specific applications, we can extract common widgets and generalizable design principles that can be used to develop other voice-driven applications and inform the design of a voice-centric user interface paradigm. This phase raises the following research questions:

- What kinds of custom voice widgets can be designed that take advantage of the expressivity of non-speech vocalization and that can generalize to be used in multiple application domains?
- What would a voice user interface toolbox for supporting rapid development of voice-driven interfaces look like?

1.4 Dissertation Outline

The subsequent chapters in the dissertation are organized as follows:

Chapter 2 provides an overview of the related work in the area of voice-driven input, in particular those that have focused on attempting to enable direct manipulation via voice.

Chapter 3 presents details of the Vocal Joystick engine and the relevant vocal characteristics and control signals it generates.

Chapter 4 describes the results from several user studies in which various performance characteristics of non-speech vocalization as computer input were investigated. In particular, the chapter includes comparative evaluations of Vocal Joystick and the mouse with novice and expert users, a longitudinal study of the Vocal Joystick learning curve involving people with and without motor impairments, and a comparative evaluation of categorical input using voice versus speech.

Chapters 5, 6 and 7 introduce three representative applications that have been built on top of the Vocal Joystick engine. Chapter 5 presents VoiceDraw, a hands-free drawing program that enables fluid free-form drawing using only one's voice. Chapter 6 presents VoiceGames, a controller that translates vocal input into mouse and keyboard input enabling control of existing computer games that were not possible or significantly difficult to control using conventional speech-based input.

Chapter 7 presents VoicePen, a prototype application that augments digital stylus input with non-linguistic vocal input to enable various interaction techniques for simultaneously manipulating multiple parameters.

Chapter 8 presents Voice Controller, a practical real-world deployment of the Vocal Joystick pointer control functionality as a Windows application that integrates with the Windows Speech Recognizer to provide the combined benefits of voice-driven and speech-driven input.

Chapter 9 highlights a number of promising future research directions that have opened up as a result of the work presented in this dissertation.

Chapter 10 concludes with a summary of contributions and some closing remarks.

Chapter 2
Related Work

While there is a large amount of prior work that has investigated the use of speech in human-computer interaction, there is relatively little research that has looked at the use of non-speech vocalizations for computer control. This chapter presents an overview of the work in this area, delving specifically into voice-driven pointer-control methods that have attempted to enable continuous control of the mouse pointer via voice. The first section describes the speech-based pointing functionality available in today's commercial speech-recognition systems. The next section highlights research efforts in academia that have investigated the use of non-speech voice input as an input modality, followed by a section that focuses specifically on voice-driven pointer control methods. The final section provides a look at some of the other hands-free input modalities for continuous control, and compares their properties to those of voice-driven inputs.

2.1 Commercially-available Speech-based Input Methods

The following subsections outline the ways in which current commercial speech recognizers attempt to support mouse-like operations. Two of the most common speech recognizers are highlighted: the Dragon NaturallySpeaking system from Nuance Communications, and Windows Speech Recognizer available as part of the Windows operating system since the Vista version.

2.1.1 Say-What-You-See Method

One way to simulate clicking on a user-interface component using speech input is to employ a method referred to as "say what you see," available in both Dragon NaturallySpeaking and Windows Speech Recognizer (Figure 2-1). This is a method in which the user simply utters the name that is associated with the desired user interface component and the system automatically invokes the default action (usually a single mouse click) on the corresponding component. If an ambiguity is detected—such as when there are two visible menu items labeled "Save" and "Save As" and the user uttered "Save"—the system presents a numbered list of ambiguous targets and requests the user to specify the desired target by uttering the corresponding number. This method works well for explicitly-labeled buttons and links, and its low cognitive demand—owing to its recognition- rather than recall-based interaction—makes it ideal for novice users. However, it fails in cases when the target does not contain any visible text label, or when attempting to point at an arbitrary location on the screen. Designing a user interface that supports this method can also be challenging, as each interactive component will need to be accompanied by a unique text label that can occupy a considerable amount of screen real-estate, or have a visual representation that unequivocally evokes the assigned name. The method also does not support any interaction that requires steering the pointer along some path, such as drawing a curve.

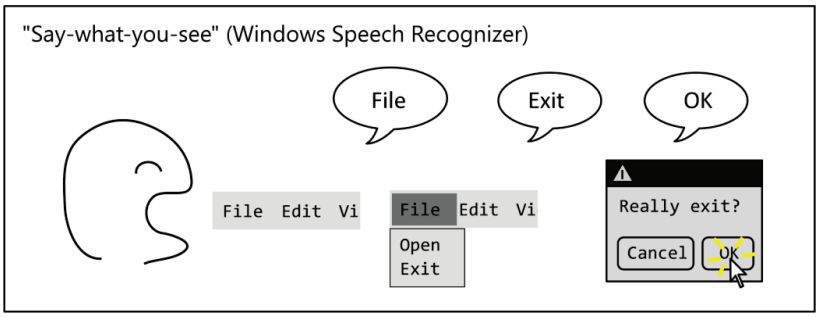


Figure 2-1: Speech-based pointer control using the "say-what-you-see" method in Windows Speech Recognizer. The user simply utters the text label of the user interface component they wish to invoke.

2.1.2 Show-Numbers Method

One solution to the problem of unlabeled targets is the show-numbers method available in Windows Speech Recognizer (Figure 2-2). With this method, the user first utters the command "show numbers," which causes a set of overlays to be displayed above all visible user-interface components that can be invoked. Each overlay is labeled with a number, and the overlays fade in and out in a pulsating manner to let the user see the targets behind them. To invoke the desired

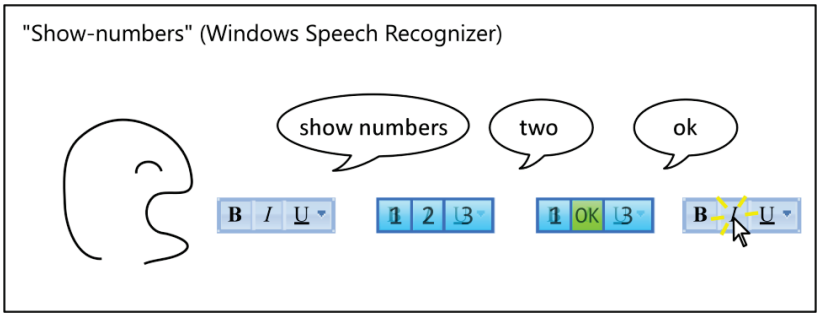


Figure 2-2: Speech-based pointer control using the "show-numbers" method in Windows Speech Recognizer. When the command "show numbers" is uttered, all user interface components that can be invoked are highlighted and overlaid with a unique number. To invoke an item, the user simply utters the corresponding overlay number and confirm with "ok."

target, the user simply utters the number of the overlay on the target, and confirms with an utterance of "ok." This method enables the selection of targets that do not have text labels, and lends itself well to densely-packed user interfaces since the number labels do not occupy as much space as text labels. One downside of this method is that a particular interface component may not always be assigned the same number because the assignment of the numbers depends on what components are available to be invoked at the time when the "show numbers" command was issued. Also, similar to the say-what-you-see method, the show-numbers method does not support steering of the pointer or pointing at an arbitrary location.

2.1.3 MouseGrid

To point at an arbitrary point using speech, Dragon NaturallySpeaking offers a feature called the MouseGrid, in which the screen is overlaid with a 3×3 grid numbered 1 through 9 (Figure 2-3).

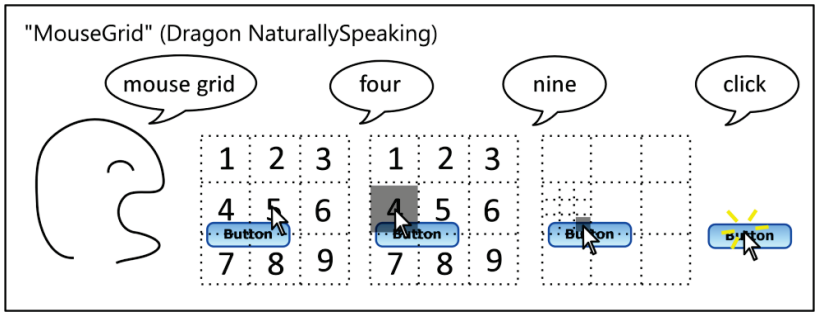


Figure 2-3: Speech-based pointer control using grid-subdivision method of the MouseGrid feature in Dragon NaturallySpeaking. When the command "mouse grid" is uttered, the screen is overlaid by a grid of nine numbered cells. The user recursively utters the number of the cell above the desired item until the center of the grid is directly over the desired item, at which point a mouse action command can be issued.

The user recursively drills down into each grid cell by saying the corresponding numbers until the desired target is under the center of the grid, at which point the user can issue various mouse actions such as left click, right click, start drag, and so on. For a screen resolution of 1,400 pixels by 900 pixels, at most seven subdivisions are required to drill down to a specific pixel when using a 3×3 grid. To drill down to a 20 pixel by 20 pixel region (a typical dimension for the height or width of buttons and links), just four subdivisions are required. While this method makes it possible to move the pointer to an arbitrary pixel using speech, it lacks the ability to smoothly steer the pointer.

2.1.4 Speech Cursor

None of the methods introduced so far support "steering," or the control of the mouse pointer along a trajectory. Windows Speech Recognizer and Dragon NaturallySpeaking both offer a method to perform steering which I will refer to as the "Speech Cursor" (there is no official name) that provides the ability to move the pointer in various directions and speeds using speech commands (Figure 2-4). For example, the user would say "move mouse right" and the cursor would start moving towards the right at a fixed velocity (the default is roughly 4 pixels per second). The user can then issue commands to change the speed (e.g., "much faster"), change direction (e.g., "left"), stop the motion ("stop"), or click the mouse button ("click"). There are three levels of commands for changing the cursor speed ("faster," "very fast," and "much faster," and corresponding ones for decelerating). The cursor movement is discontinuous, updating its position roughly four times a second, thereby skipping over a number of pixels when the velocity is greater than the default. The movement directions are limited to the eight cardinal and ordinal

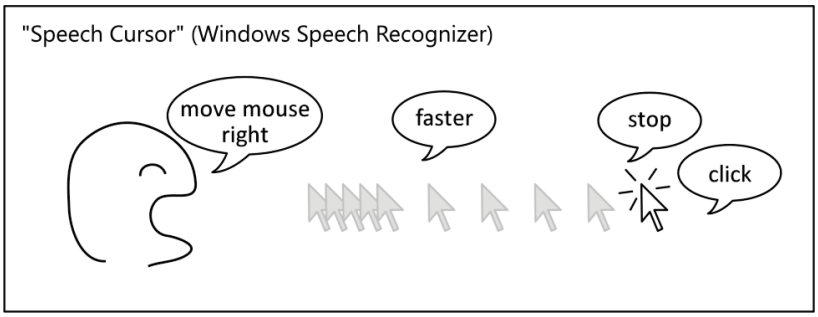


Figure 2-4: Speech-based pointer control using the Speech Cursor feature in Windows Speech Recognizer. The pointer movement is initiated by the command "move mouse *direction*," at which point the pointer starts moving at a fixed speed towards the designated direction. The speed and direction is controlled by uttering commands such as "faster" and "upper left," respectively.

compass directions (up, upper-right, etc.). An advantage of this method is that the command phrases are familiar phrases that map directly to the intended changes in speed and direction.

2.2 Non-speech Voice-driven Pointer Control

There are many other patterns of human vocal production other than spoken words, including acoustic properties such as variations in pitch, volume and duration that contribute to prosody, timbre, and vowel quality. Prosody is a characteristic associated with speech, describing the underlying intonation, stress, and rhythm. Timbre, or tone quality, applies more generally to sounds including those produced by musical instruments and describes the characteristics of the sound that enable a human observer to distinguish one sound from another. Vowel quality is similar to tone quality, except applied specifically to the differentiation of vowel sounds.

Despite the relatively limited investigation into the use of non-speech vocal input for direct interface control, some notable systems have emerged. Several systems have been proposed to specifically enable control of the mouse pointer using voice input, both in academic research as well as in commercial products. Because much of the continuous aspect of user interface input is driven via pointing devices such as the mouse, it would be natural to pick out pointer control as the domain in which to explore the capacity of voice input to deliver continuous control.

Igarashi and Hughes (2001) proposed several techniques for using non-speech voice features such as utterance duration, pitch, and discrete sound frequency to control the rate of change of some interface elements such as panning and zooming of a map. For example, to scroll the map down, the user would say “move down” followed by a continuous vocalization of any sound, during which the map would scroll as long as the sound was present. Scrolling speed was continuously mapped to the magnitude of the difference in pitch relative to the beginning of the vocalization. Lowering the pitch resulted in slower scrolling and raising the pitch resulted in faster scrolling. While pointer control was not demonstrated in their original work, an illustration of how this method can be applied to such a control is shown in Figure 2-5. The advantage of this method over the other pointer control methods introduced so far is the continuous mapping of the magnitude of the change in pitch to the movement speed, enabling fluid manipulation not possible with discrete input. However, the need to utter discrete spoken commands to switch directions still makes it less than ideal for continuous two-dimensional pointer control.

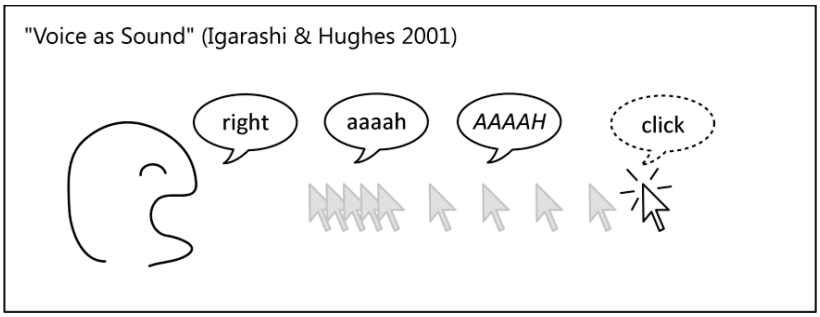


Figure 2-5: Voice-driven motion control proposed by Igarashi and Hughes using spoken commands and pitch changes. While pointer control was not originally demonstrated by the authors, their method can easily be applied in this context. Using their method, the user would first utter the desired direction of movement, followed by continuous vocalization of *any* sound to get the pointer to move at a speed directly proportional to the relative pitch of the vocalization. Increasing the pitch (indicated in the figure by capitalization) results in faster motion and vice versa, and the motion stops when the user stops vocalizing.

Sporka et al. (2004) investigated the use of change in pitch as the method for controlling the mouse cursor by humming or whistling (Figure 2-6). They divided the user's pitch range into high and low, and defined four pitch gestures where each gesture is specified by the starting pitch (high or low) and the direction of the subsequent pitch change (increasing pitch or decreasing pitch). The four gestures are mapped to the four cardinal directions. When the system detects one of the pitch gestures, the pointer starts to move in the corresponding direction until the user stopped vocalizing. A magnitude of the pitch difference from the start of the vocalization mapped to the speed of the pointer. A short sustained high-pitch gesture was used to issue a mouse click. A nice feature of this method is that the basic pitch inflections may be vocalized or hummed even by users who may not be able to fully verbalize spoken words. However, it has yet to be

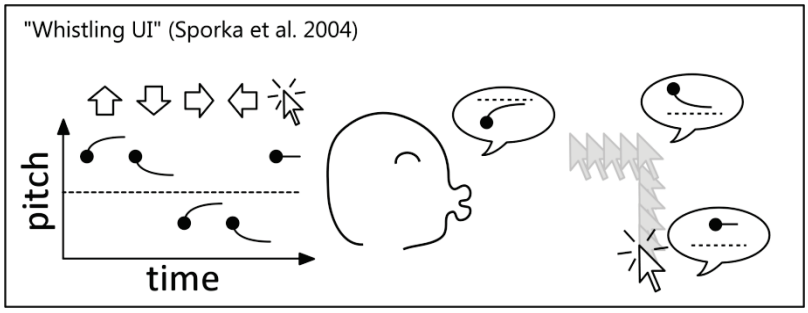


Figure 2-6: Voice-driven pointer control proposed by Sporka et al. (2004) using pitch-inflection "gestures" mapped to different movements and actions. The user's "pitch space" was divided into high and low pitch (separated by a dotted line in the figure), and a pitch gesture was defined as a rising or falling pitch relative to the high or low starting pitch. The pointer moved in the direction corresponding to the detected gesture for the duration of the vocalization. A short sustained pitch vocalization issued a click.

determined how well people will be able to use such a method beyond a pilot study group. Their system is also limited to movement along the cardinal directions, and smooth continuous switching of direction is not possible without first stopping the utterance.

The Migratory Cursor technique (Mihara et al. 2005) also combines discrete verbal command and non-verbal utterance for positioning the cursor. This technique augments the standard cursor with a row or column (depending on the movement direction) of ghost cursors that are numerically labeled. The user hones in on the desired target by using the numeric labels on the line of ghost cursors to specify the coarse-level coordinate (e.g., by uttering "move left, eight") and uses non-speech vocalization to precisely adjust the position (e.g., saying "ahhh" during which the cursor continues to move left at a slow pace). This technique also lacks the ability to fluidly and continuously change the movement direction without having to issue discrete commands.

The SUITEKeys (Manaris & Harkreader 1998; Manaris et al. 2001) system provides a cursor control method similar to SpeechCursor presented above, based on a constant velocity cursor that is initiated by voice commands such as "move mouse down" and "move mouse two o'clock." The cursor continues to move (although it is not specified at what speed and whether it is constant) until the user says "stop" or issues a button press command such as "click left button." The system also provides a way to move the cursor in some direction by a specified amount, or to set its position to a specified coordinate. As with the Migratory Cursor technique, this interaction method also suffers from the discontinuous nature of verbal commands and the inability to control the movement speed efficiently. Karimullah and Sears (2002) proposed an enhancement to this method in which a predictive ghost cursor is displayed ahead of the actual cursor position to allow the user to account for the inherent delay in the recognition and execution of the "stop" command by the system, but they reported no significant improvements in performance.

Building on the work of Kamel and Landay (2002), Dai et al. (2004) compared two different versions of the grid-based cursor control method, similar to the MouseGrid feature described above for Dragon NaturallySpeaking. The two systems differed in whether there was only one active cursor at the center of the middle grid, or nine active cursors at the center of each of the grids. Their results showed that the nine-cursor version resulted in significantly faster task times for acquiring targets of various sizes and distances. Although the grid-based approach can be quite efficient in moving the cursor to a particular point on the screen, it does not allow the user to move the cursor continuously across the screen, as is necessary for tasks such as drawing.

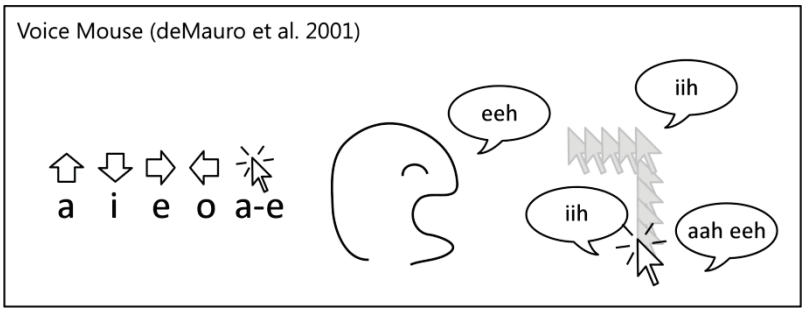


Figure 2-7: Voice-driven pointer control using non-speech vocalizations as proposed by deMauro et al. (2001). To start a pointer moving in a particular direction, the user utters a short vowel sound corresponding to the direction. The pointer's speed increases at constant acceleration, until another short vowel utterance is made. If the uttered vowel sound is different from the previous vowel sound, the pointer changes its movement direction to the new sound's direction. If the uttered sound is the same as the previous sound, the pointer stops moving. A special sequence of two different vowel sounds uttered in quick succession issued commands such as a mouse button click.

The system that comes close to continuous voice control of a pointer is Voice Mouse (de Mauro et al. 2001), which uses different vowel sounds associated with each cursor movement direction. In this system, the user utters a vowel sound corresponding to the desired one of the four directions ("a" for up, "e" for right, "i" for down, and "o" for left), and the cursor starts moving in that direction. Once the cursor starts moving, it is governed by "inertial motion" and the user does not have to continue vocalizing. The cursor speed initially starts out slow and gradually accelerates with time. The cursor is stopped by uttering the same vowel sound again, and a click is performed by uttering a two-vowel command ("a-e"). One major difference between the Voice Mouse and the Vocal Joystick is that with the Voice Mouse, the cursor does not start moving until a vowel sound has been produced for a minimum duration and is recognized by the system (which could take around four seconds from the start of the vocalization). The Vocal Joystick, on the other hand, starts the cursor movement as soon as the user starts producing the vowel sound. The inability to control the cursor speed and to change its direction without stopping it is also a major limitation of the Voice Mouse system remedied by the Vocal Joystick.

2.3 Other Non-speech Voice-based Input

Aside from the pointer control methods presented in the previous section, most of the prior work that attempt to utilize non-speech features from human vocalizations have been focused on applications such as disambiguating parse results in speech recognition (Kahn et al. 2005, 2004; Liu et al. 2004), and music retrieval and production (Goto & Hayamizu 1999; Nwe & Li 2007), and not on using them to directly manipulate the user interface. Olwal and Feiner (2005)

experimented with the use of prosodic features in speech commands such as speech rate, utterance duration, and loudness in a prototype system where the user could rotate and translate an object using verbal commands such as "move left," with the rate of change being controlled by the various prosodic features (e.g., saying "moooove leeeft" slowly resulted in slower translation). No evaluation was conducted of their proposed system. Patel and Abowd (2007) investigated the use of sound produced by blowing at a laptop screen with a built-in microphone as a way to control certain aspects of the user interface such as scrolling, dragging, and selecting icons. These works, however, have not fully explored the potential of non-speech voice input as a generalizable computer input modality.

Igarashi and Hughes (2001) introduced several non-speech voice-based input methods which helped inspire a number of subsequent research on voice-driven pointer control methods discussed in further detail in the following section. One of the simpler methods they presented was the use of "tonguing"—short peaks in the sound signal that can be produced by utterances such as "ta ta ta"—for manipulating discrete parameters. For example, in a diagramming application, the user may nudge a shape into place pixel-by-pixel by uttering "nudge right, ta ta ta," resulting in the shape being shifted to the right by three pixels. The advantages of such short discrete vocalizations are that they are simpler to utter than most words such as "right" and can be repeated very rapidly. Such characteristics are especially useful when the user does not know the exact number of discrete inputs required ahead of time (as in the shape nudging example in which the user relies on the continuous visual feedback to decide when it is complete), or when multiple discrete input needs to be supplied in rapid succession (as is typical in action-oriented games). The major limitation of the implementation demonstrated by Igarashi and Hughes was that the system treated any instantaneous spike in the input volume as a discrete input, thereby treating any short utterance or even a clap of the hands as the same, requiring the utterance of a spoken command such as "nudge right" to set the context for how the signal should be interpreted. Also, the fact that each instance of an utterance is mapped to only one input event makes it impractical and inefficient for specifying a large number of discrete events.

Goto et al. (1999) also developed several unique methods for combining non-speech vocalizations with speech utterances to enhance speech-driven input. They introduced a novel speech interface functionality called *speech completion* (Goto et al. 2002) that can detect *filled pauses*—extended vocalization of a voiced sound such as "uhh" that typically signifies hesitation

or contemplation—and uses it to augment various speech inputs. Their demonstration system provides the ability for a user to issue search queries by voice into a music retrieval system, even when the user does not remember the exact name of the song or artist. When the user wishes to search for songs by an artist with the first name of Michael but whose last name cannot be recalled, the user can utter the word "Michael" followed by a filled pause, sustaining the trailing vocalization at the end of the word. The system detects the filled pause, and presents the user with a numbered list of most likely completion candidates based on the words recognized up to that point, akin to the auto-completion feature in most text-based search engines. The user can complete the phrase by reading out the rest of the desired name, or simply uttering the number corresponding to the desired list item. The use of filled pauses in this type of scenario is particularly appropriate since it closely resembles the natural vocalizations that one makes when faced with uncertainty or when contemplating the invocation of some command.

Goto et al. (2004) also developed a system called the Speech Spotter that uses the average pitch of an utterance as a way to distinguish between utterances intended for the speech recognition system and those intended for another human listener with whom the user happens to be conversing. When the user's utterance is at the average speaking pitch level, the system assumes the user is speaking to another person and ignores the utterance. When the user makes an utterance at a higher pitch than their average pitch, the system processes it as a command intended for the system. The method can also be used to disambiguate between dictation mode and commanding mode when composing text using speech, in which the system needs to determine whether the phrase uttered by the user such as "new paragraph" was meant as a command or as text to be entered into the document literally. Most commercial speech recognition systems address this ambiguity by requiring the user to pause slightly before uttering a command phrase when in dictation mode. In the Speech Spotter system, the system can be set to interpret higher pitch utterances as commands and regular pitch utterances as dictation text. They also incorporated their filled pause detector into the Speech Spotter system, making the system less susceptible to unintended recognitions by requiring a filled pause to be uttered before the high pitch utterance for it to be processed as a command phrase. Both the filled pause detection and pitch-based Speech Spotter functionality offer a simple mechanism for quickly switching operational modes during voice-based input.

2.4 Other Hands-free Continuous Control Modalities

A natural question to ask may be, why not use other hands-free devices such as eye trackers or head trackers, especially within the context of accessibility? There are indeed a number of such devices that can offer continuous control, each with their strengths and drawbacks.

Mouth-operated joysticks such as the Integra Mouse from Tash Inc.[1] and Quad-Joy from SEMCO[2] provide pointer control by gripping the joystick tip with the mouth and using the lips and sometimes the neck to control movement. Clicking is enabled through a sip and puff switch typically integrated into the joystick. The need to keep the joystick in the mouth and fatigue are often cited by their users as major issues with such devices. Their cost can range from several hundred to several thousand dollars, with additional expense incurred for the mounting units.

Eye trackers provide an alternative input modality that requires only the ability to move one's eyes. Although these devices provide an attractive option for individuals with very limited motor capacity, they require specialized hardware with high calibration requirements, and are several orders of magnitude more expensive than the computer itself. They also incur extra steps in the interaction process to work around the 'Midas touch' problem (Jacob 1990), in which the user may inadvertently click on a target which they only intended to look at without acting upon it.

Head operated devices such as the HeadMaster Plus from Prentke Romich Company[3] offer relatively accurate pointer control using a tracking object (usually an ultrasonic sensor or an infrared reflective sticker) attached to the user's head. This requires that a separate tracking device be located near the computer screen. Clicking can be performed either through dwelling (fixating on a point for longer than a fixed duration) or through the use of an additional device such as the sip and puff switch. Although generally less expensive than an eye tracker, a head tracker can still cost orders of magnitude more than a conventional mouse.

Other solutions such as tongue-operated pointing devices (Kencana & Heng 2007; Salem & Zhai 1997) and EEG-based methods using the electrical activity in the brain to control the movement of the pointer (Zhu et al. 2007) have been proposed, but the state of these technologies is still too premature to deduce their efficacy as practical input systems.

---

[1] http://www.tashinc.com/
[2] http://www.quadjoy.com/
[3] http://www.prentrom.com/

2.5 Summary

My proposition is not that voice-based interaction is superior to all of these alternative hands-free input technologies. In fact, voice can be a great supplement to many of these devices in compensating for their various shortfalls. However, my research places primary focus on voice for several reasons. First, voice interaction does not require any elaborate hardware setup, only a microphone whose cost is comparable to a mouse. Second, the human voice is produced by muscle groups that are very highly developed, second only to the hand as illustrated in the motor homunculus representation in Figure 2-8. The complexity of the motor control required to produce human speech (Kent 2000) and the significant role that these cerebral regions play in the production of speech (Murphy et al. 1997) explain the highly expressive nature of the human voice (Oller 2000), much of which remains unexploited in human-computer interaction. Third, especially when combined with spoken word input, voice can deliver much greater informational bandwidth than most other hands-free input methods.

It goes without saying that voice input is not without its limitations. As was mentioned in the introduction, it is important to acknowledge such limitations and guide the design decisions based

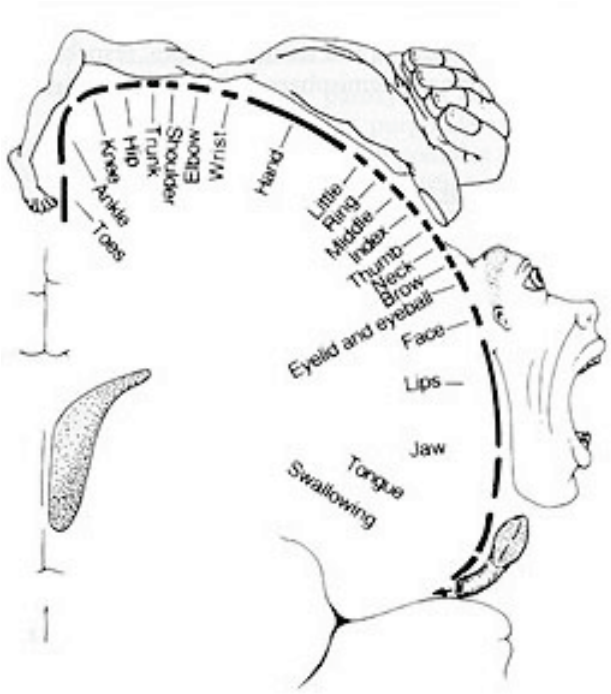


Figure 2-8: The motor homunculus representation of the primary motor cortex and regions associated with the execution of movement in the corresponding body parts, adapted from Penfield & Rasmussen (1952).

upon them. Susceptibility to noise is a major concern with speech user interfaces, although improvements in microphone technology are continuing to mitigate this factor. There are social constraints as well, where vocalizing (especially non-speech sounds) may not be socially appropriate. This may be addressed to a degree by better microphones that can reliably detect user's utterances even at low volume levels, or by workspace rearrangements. Vocal fatigue can also be an issue, thus systems need to ensure that they do not demand from the user any more vocalization than necessary. Despite these possible constraints, voice input has the potential to greatly enhance the state of hands-free input, especially for people with motor disabilities.

Chapter 3
The Vocal Joystick Engine

This chapter provides a high level description of the Vocal Joystick engine (Bilmes et al. 2006), a system that lies at the heart of the projects described in the rest of this dissertation. Development of the Vocal Joystick engine was started at the University of Washington under a cross-departmental initiative headed by Dr. Jeff Bilmes from the department of Electrical Engineering, along with students and faculty from the departments of Speech and Hearing Sciences, Linguistics, and Computer Science and Engineering. The initial application of the Vocal Joystick engine was a mouse pointer controller that enabled the user to smoothly control the mouse pointer using continuous non-linguistic vowel-like vocalizations. During the course of my dissertation work, the Vocal Joystick engine has evolved into a cross-platform, modular library, and a number of additional applications have been built upon it.

At the highest level, the Vocal Joystick engine is functionally similar to a traditional automatic speech recognition (ASR) engine. Figure 3-1 shows the processing stages common to both types of engines. A typical process through the system is as follows: The sound wave resulting from the user's vocalization reaches the microphone, where its amplitude is sampled at a fixed rate by the audio-extractor module and converted into its digital representation. The digitized audio samples

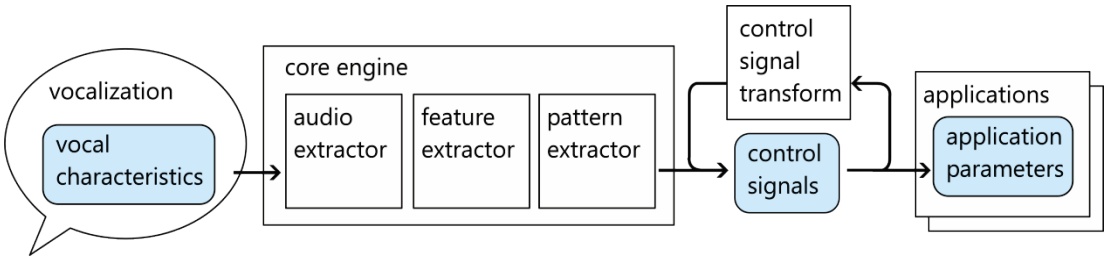


Figure 3-1: High-level and simplified functional flowchart describing the processing stages common to both the Vocal Joystick engine and traditional automatic speech recognition (ASR) engines.

are then processed by the feature-extractor module, which calculates various basic features across a block of audio samples, such as the average power, number of zero crossings, and various spectral features including Mel-frequency cepstral coefficients (MFCCs) (Davis & Mermelstein 1990). The extracted features are further processed by the pattern-extractor module, which attempts to decode and classify the features into a higher-level pattern that is being recognized by the system. In the case of an ASR engine, this higher-level pattern belongs to a set of words and phrases; in the case of the Vocal Joystick engine, they belong to a set of vowel sounds and discrete sounds (which are defined in the following section). Finally, the classified patterns are interpreted by some external application to cause some change in its state. In the case of an ASR engine, the recognized words may be processed by the operating system to invoke a corresponding command, or by a word processing application to enter the uttered text into the document.

Beyond the high-level similarities between the Vocal Joystick engine and ASR engines, the two systems differ in a number of ways. First, the Vocal Joystick engine continuously tracks and outputs features of *continuous non-linguistic vocalizations*, such as pitch, power, and vowel quality. This is in contrast to typical ASR systems that output textual representation of speech utterances as they are recognized at the word or phrase level, resulting in a series of intermittent output. Second, the Vocal Joystick engine can also recognize *discrete non-linguistic vocalizations* (e.g., sounds such as “ck” and “ch”), which are similar to spoken words but can be much shorter and thus faster to utter and be recognized. Finally, due to the elemental nature of the vocal features tracked by the Vocal Joystick engine, if the user wishes to adapt the engine’s acoustic model to his or her voice, the amount of sample utterance required is significantly less—on the order of several seconds—compared to ASR systems, which typically require several paragraphs of sample text to be read.

From the user interaction perspective, the two parts of the functional flowchart shown in Figure 3-1 that are of particular interest are the *vocal characteristics*—the aspects of the user's input the system can recognize—and the *control signals*—the types of signals the system can output and provide as input to the intended applications. The following sections present the nature of the vocal characteristics processed by the Vocal Joystick engine and the control signals it outputs, followed by a technical overview of the Vocal Joystick engine architecture and its evolution.

3.1 Vocal Characteristics Processed by the Vocal Joystick Engine

Figure 3-2 shows the various categories of vocalizations that are processed by traditional ASR engines and the Vocal Joystick engine, along with examples of each type of vocalizations. Unlike traditional ASR engines that recognize sequences of spoken words, the Vocal Joystick engine processes the user's *non-linguistic* (or *non-speech*) *vocalizations*.[1] By *non-linguistic vocalization* I mean utterances that do not have any specific correspondence to words or phrases in a language,

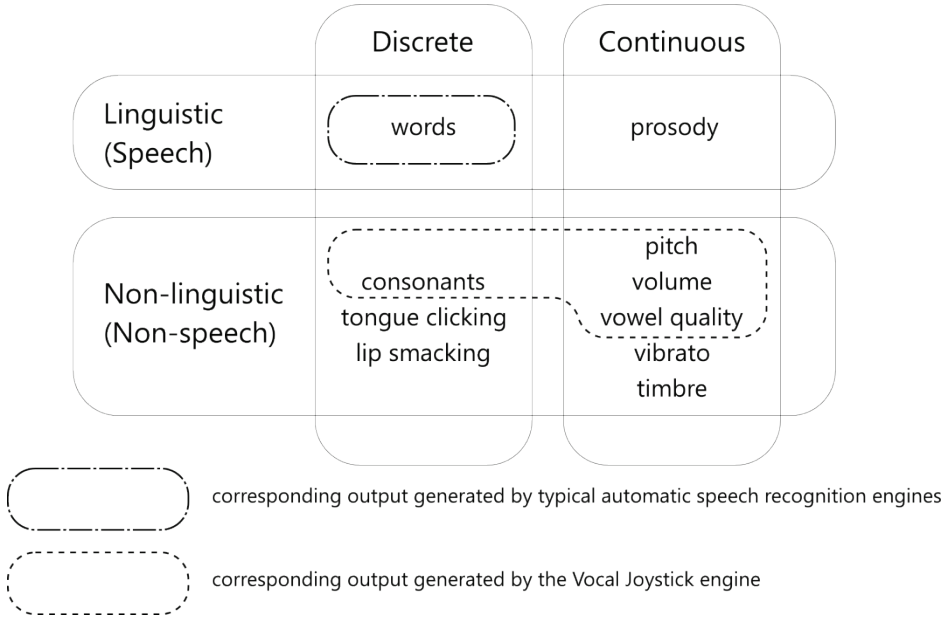


Figure 3-2: Examples of vocal characteristics whose corresponding outputs are generated by traditional automatic speech recognition (ASR) engines and the Vocal Joystick engine. The vocal characteristics are categorized along two dimensions. The vertical dimension distinguishes between linguistic and non-linguistic vocalizations. The horizontal dimension distinguishes between discrete and continuous vocalizations. Examples of vocal characteristics for each combination of categories are also provided. As can be seen, typical automatic speech recognition (ASR) engines do not process any continuous vocalizations, while the Vocal Joystick engine does.

[1] For the purposes of my dissertation, I consider the terms "non-linguistic" and "non-speech" to be interchangeable, as well as the terms "linguistic" and "speech."

such as the sounds "aaaah," "zzzzz," or even humming sounds. Non-linguistic vocalizations can be categorized as being either *continuous* or *discrete*.

Continuous non-linguistic vocalizations, or *continuous sounds* for short, include vocal features that can be vocalized continuously, and possibly be modulated smoothly, over some duration of time, such as pitch, power, and vowel quality. One notable property of these continuous sound features is that it is possible to modulate multiple features simultaneously, such as increasing both the pitch and power while changing the vowel quality.

Discrete non-linguistic vocalizations, or *discrete sounds* for short, represent vocalizations that are very short in duration, typically on the order of several hundred milliseconds or less, and are usually consonant-like. Examples of discrete sounds include isolated consonant sounds such as [*k*] or [*t*], as well as clicking or tonguing sounds. The short duration of these discrete sounds enables the recognizer to recognize and generate an output much quicker than with spoken words.

The following sections describe in further detail each type of non-linguistic vocalizations that are processed by the Vocal Joystick engine.

3.1.1 Pitch

When a sound is produced through the vibration of the speaker's vocal folds (often referred to as vocal cords), the resulting sound is called a *voiced sound*. When the Vocal Joystick engine detects the presence of a voiced sound, the engine estimates its fundamental frequency, or the rate of closing and opening of the vocal folds, and reports the resulting value as the observed *pitch* value, in hertz (Hz). The Vocal Joystick engine tracks pitch values in the range of 50Hz to 500Hz.

3.1.2 Power

The *power* represents the intensity of the vocalized sound as measured by the microphone. The Vocal Joystick engine reports every 10 milliseconds the power value averaged over a 40-millisecond window. One important distinction to be made is that the power value reported by the Vocal Joystick engine represents the *measured* sound intensity, and not the *perceived loudness,* which is a human perceptual quantity that is non-linearly related to the measured sound intensity and is dependent on various factors such as the underlying frequency of the sound or duration. Power is calculated from the audio samples within each 40 millisecond window as follows:

$$P = \frac{1}{n}\sum_{i=1}^{n} {s_i}^2 \tag{1}$$

where n is the number of samples in a 40-millisecond window (640 in the case of 16 kHz sampling frequency) and s_i represents the raw signal amplitude observed in the ith sample. The Vocal Joystick engine also provides an alternate representation of the power value on the logarithmic scale relative to a reference value, referred to as *volume*. Volume is measured in decibels (dB), or more specifically, decibels relative to full-scale digital (dBFSD), and is calculated as:

$$V = 10\log_{10}\left(\frac{P}{P_{\text{FSD}}}\right) \tag{2}$$

where P is the mean measured power value, and P_{FSD} is the maximum possible value of P (in the case of signed 16-bit samples, this value would be $(2^{15})^2 = 2^{30}$). Since the human perception of loudness varies roughly linearly with the cubed root of the sound power (Stevens 1957), the volume may be a more useful representation than power, especially in the context of user interface components such as a meter that provides visual feedback for the audio signal level referred to as the VU (Volume Unit) meter.

3.1.3 Vowel Quality

Vowel quality is a description of a particular *vowel sound* characterized by a set of *articulatory features*. *Vowel sounds* are sounds that are produced when the vocal cord vibrates with the vocal tract open, and *articulatory features* describe the state of the physiological structures involved in vocalization, such as the shape of the lips, position of the tongue, and the openness of the vocal tract. From the perspective of *acoustic features*, or features that describe the nature of the produced sound, vowel sounds are typically characterized by the amplitude of their first two *formants*—a set of resonant frequencies determined by the fundamental frequency of the vocal cords and the spatial arrangement of the vocal tract.

Figure 3-3a shows the International Phonetic Alphabet (IPA) vowel chart, a chart of vowel sounds as defined and mapped by the International Phonetic Association in which each sound is represented by the corresponding IPA symbol. The chart shows that different vowel sounds can be vocalized by varying three primary articulatory features. The first two features—vowel height

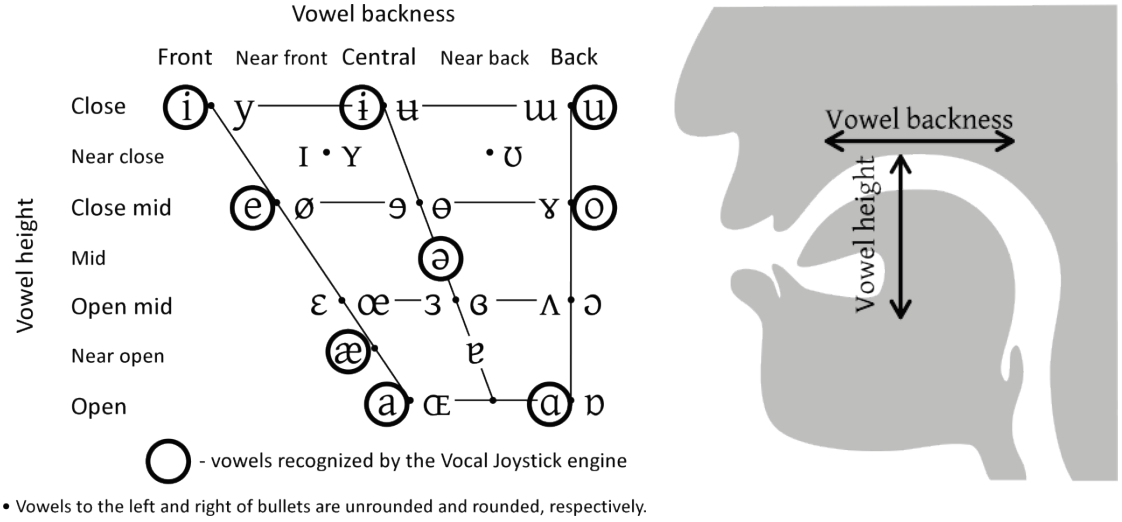


Figure 3-3: (a) The International Phonetic Alphabet vowel chart showing various vowel-sounds as they relate to their two primary articulatory features (vowel height and vowel backness). The circles indicate sounds that are recognized by the Vocal Joystick engine. (b) A midsagittal-plane view of the mouth region showing how the position of the tongue relates to the definition of vowel-height and vowel-backness.

and vowel backness−are characterized by the position of the tongue along the midsagittal plane[2], as illustrated in Figure 3-3b. The third feature, vowel roundedness, is characterized by the shape of the lips during vocalization, and is represented in Figure 3-3a by pairs of vowel sounds, with rounded vowels appearing to the right of their corresponding unrounded vowels wherever such a pairing exists. An important point to observe is that this *vowel space*—the space of sounds defined by these three features—is continuous, even though Figure 3-3a shows a discrete set of vowel sounds. In other words, it is possible for a speaker to smoothly vary any of the three articulatory features to continuously transition from vocalizing one sound to another within this vowel space. This is the primary characteristic of vowel sounds that make them an ideal candidate for significantly expanding the scope of control input types that can be specified using the voice.

The Vocal Joystick engine contains a vowel recognizer that can recognize up to nine vowels, indicated in Figure 3-3a as circled vowels. For every frame of audio data, the vowel recognizer calculates the probability distribution across all the vowels, representing the likelihood that the audio frame was generated as a result of the corresponding vowel being vocalized. The vowel

[2] More precisely, vowel height and vowel backness are typically characterized by acoustic features (first and second formants, respectively) of a vowel sound, but for illustrative purposes we describe them here in terms of the position of the tongue, which roughly corresponds to their respective acoustic features.

sound with the highest probability above a preset threshold is considered to be the "recognized" vowel for that frame, although for most applications including the pointer control application, the vowel probabilities are aggregated or interpolated to provide a more continuous output. Further discussion of the vowel recognizer is provided in Section 3.3.2.

The particular set of nine vowels was chosen by linguists, phoneticians, and electrical engineers on the Vocal Joystick team based on several considerations. First, a total of nine vowel sounds—eight along the perimeter of the IPA vowel chart and one at the center—were chosen so that the natural two-dimensional mapping of those sounds can be exploited by the Vocal Joystick engine (the details are provided in the subsequent section on the Vocal Joystick engine architecture). Second, sounds that are acoustically distant from each other were chosen to maximize the recognizer's ability to distinguish among them.[3] Third, sounds that are more prevalent across the world's major languages were favored over those that are less prevalent.

3.1.4 Discrete Sounds

The Vocal Joystick engine also has the ability to recognize an arbitrary sequence of phonemes that start with a consonant sound, including words as well as isolated consonants. Under the default configuration, the Vocal Joystick engine recognizes two discrete sounds, the consonants [k] (as in the ending of the word "click") and [tʃ] (as in the start of the word "change"), henceforth indicated for clarity as "ck" and "ch", respectively. Further details of the discrete sound recognizer are provided in Section 3.3.2.

3.1.5 Summary of the Vocal Characteristics Tracked by the Vocal Joystick Engine

The above set of vocal characteristics tracked by the Vocal Joystick engine is summarized in Figure 3-4. Pitch and power are represented by vertical arrows as continuous scalar values. The two discrete sounds that the Vocal Joystick engine can recognize by default are represented as two rounded corner shapes. The nine vowel sounds that are classified by the Vocal Joystick engine are represented in a radial "vowel compass," in which the relative position between vowels corresponds to that of the IPA vowel chart, with the eight peripheral vowel sounds mapped to each of the cardinal and ordinal compass directions and the mid-central vowel sound [ə] mapped to the directionally-neutral central position. This radial arrangement provides a

[3] The *acoustic distance* between two vowel sounds does not necessarily correspond to their Euclidian distance in the spatial representation of the IPA vowel chart shown in Figure 3-3a, since the IPA vowel chart is a representation based on articulatory features rather than acoustic features.

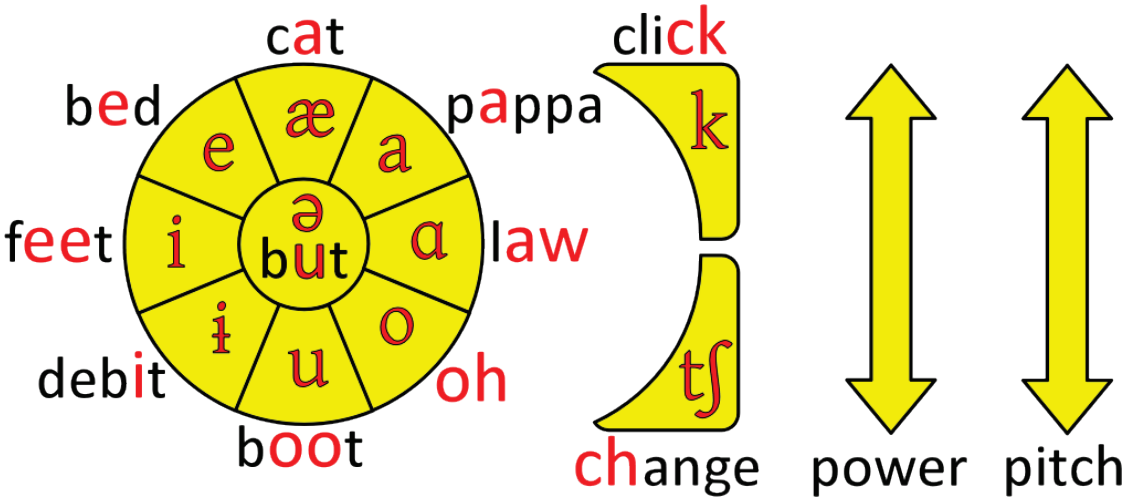


Figure 3-4: The "vowel compass" on the left shows the sounds mapped to each radial direction in the Vocal Joystick. The red vowels in each word approximate the sound corresponding to that direction. The middle vowel sound (represented by the IPA symbol [ə]) can be used to modulate power or pitch without any directional change. The Vocal Joystick also tracks power and pitch, as well as discrete (non-vowel) sounds such as "ck" and "ch."

concrete mapping between the vowel sounds and the well-defined, equally-distributed two-dimensional directions. Such a concrete mapping makes it easily generalizable to a variety of two-dimensional controls. It also aids the user in learning the mapping, as they can associate a vowel sound or a pair of vowel sounds to one of the two cardinal- or two ordinal-axes, as opposed to the less concrete spatial association in the original IPA vowel chart.

The Vocal Joystick can operate under any one of 4-, 5-, 8- or 9-vowel modes. Under the 4-vowel mode, only the four main cardinal-direction vowel sounds ([æ], [a], [u], and [i]) are recognized, and the 5-vowel mode adds the ability to recognize the central vowel sound ([ə]) as well. While these modes provide coarser granularity for directional control, they can serve as a good starting point for novice users since they are more tolerant to slight deviations in the vocalizations from the intended sounds. Similarly, the 8-vowel mode includes the eight vowel sounds along the periphery of the vowel compass, with the 9-vowel mode including the central vowel sound.

One may observe that the arrangement of the vowel sounds shown in Figure 3-4 is not exactly the same as that shown in the original IPA vowel chart in Figure 3-3a. While the relative arrangement between vowels is the same, the Vocal Joystick engine's vowel compass has transformed the original IPA vowel chart by rotating it clockwise by about 135 degrees then reflecting across the vertical axis. This is an artifact from the first version of the Vocal Joystick engine, whose

directional mapping was not based on the two dimensional vowel arrangements. In that version, vowel quality was used to determine the horizontal direction, and pitch change was used to determine the vertical direction. For the horizontal direction, vocalizations whose difference between the first two formants was greater than a certain threshold were mapped to one direction, and those whose difference was less than a certain threshold were mapped to the other direction. To maximize the discriminability of these sounds, two vowel-sounds, one with the largest and the other with the smallest difference between the first two formants ([i] and [ɑ], respectively), were chosen to represent each of the two horizontal directions. These vowel sounds define a nice central axis through the IPA vowel chart, and thus their mapping to the horizontal directions carried over to subsequent versions of the Vocal Joystick engine as the engine gained the ability to classify arbitrary vowel sounds. The initial decision to assign [æ] as opposed to [u] as the upward direction was loosely based on an informal survey of several people asking them which sound they felt best represented each direction. Therefore, the particular orientation of the current vowel compass mapping holds little semantic significance, and thus may be arbitrarily rotated or reflected without affecting the functionality of the system.

Despite the arbitrary nature of the orientation of the vowel compass mapping, an important consideration that should be taken into account is that the relative arrangement of the vowel sounds to each other is significant. This is so that as the user varies the vocalization from one vowel-sound to an adjacent sound in the vowel compass, the radial direction output can also transition smoothly. Suppose that the user wishes to smoothly transition the output from a leftward direction towards an upper-left direction. Under the current mapping, the user would start vocalizing [i], and then gradually change the structure of the vocal tract towards vocalizing [e]. If for instance the mapping of the vowel sounds [e] and [æ] were swapped such that [æ] maps to the upper-left direction and [e] maps to the up direction, as the user attempts to vary the vocalization from [i] to [æ], there may be a moment when the output for the up direction is unintentionally generated. This is because the sound [e] is likely to be vocalized during a smooth transition from [i] to [æ], as can be seen in the IPA vowel chart. If the angular granularity offered by the eight directions is not required, however, and the four cardinal directions are sufficient, then it is possible to reassign a particular cardinal direction to a different vowel sound that belongs to one of the adjacent ordinal directions, at the expense of a possible decrease in precision in the transition to and from such direction.

3.2 Control Signals Provided by the Vocal Joystick Engine

The previous sections presented the core set of control signals that the Vocal Joystick engine can output and provide as input to client applications. This set of control signals includes continuous signals representing pitch, power, and vowel probabilities, as well as instantaneous signals representing the utterance of discrete sounds. This section focuses on the *application parameters*—the types of input signals that client applications expect from input devices—as well as the *control-signal transforms*—the process by which the control signals generated by input devices are translated into those application parameters.

The power of the Vocal Joystick engine lies not just in the rich set of core control signals it provides, but also in the variety of ways in which they can be combined and composed into other types of control signals, such as two-dimensional displacements and multiple keystrokes. In fact, this ability to transform control signals of one type to another is especially important in assistive technologies, in which devices of various forms and capabilities—often more limited than keyboards and mice—need to emulate the keyboard and mouse for which most applications have been designed.

To illustrate the richness of the Vocal Joystick control signals and the ways in which they can be transformed into various application parameters, I present in this section a diagrammatic representation of control signal types and their corresponding control transforms (Figure 3-5). The representation is not meant to serve as a complete model or taxonomy of input devices and control signals, but simply as a visual language with which we can inspect and compare the possible mappings between input devices and applications. While there are many input taxonomies that focus on categorizing input devices and signals (Buxton 1983; Foley et al. 1984; Lipscomb & Pique 1993; MacKenzie 1995; Myers 1990), my representation focuses on categorizing the *transformations* that can be applied to the input signals, highlighting the compositional aspect of control signals and facilitating the conceptualization of new ways to construct such compositions. The representation can be seen as a graphical depiction of a more formal model of input devices and *composition operators* presented by Card and Mackinlay (Card et al. 1991; Mackinlay et al. 1990), viewed in light of the dimensionality of the control signals as presented by Buxton (1983). The diagrammatic representation was inspired by the Input Configurator Toolkit (Dragicevic & Fekete 2004), which provided an interactive tool for

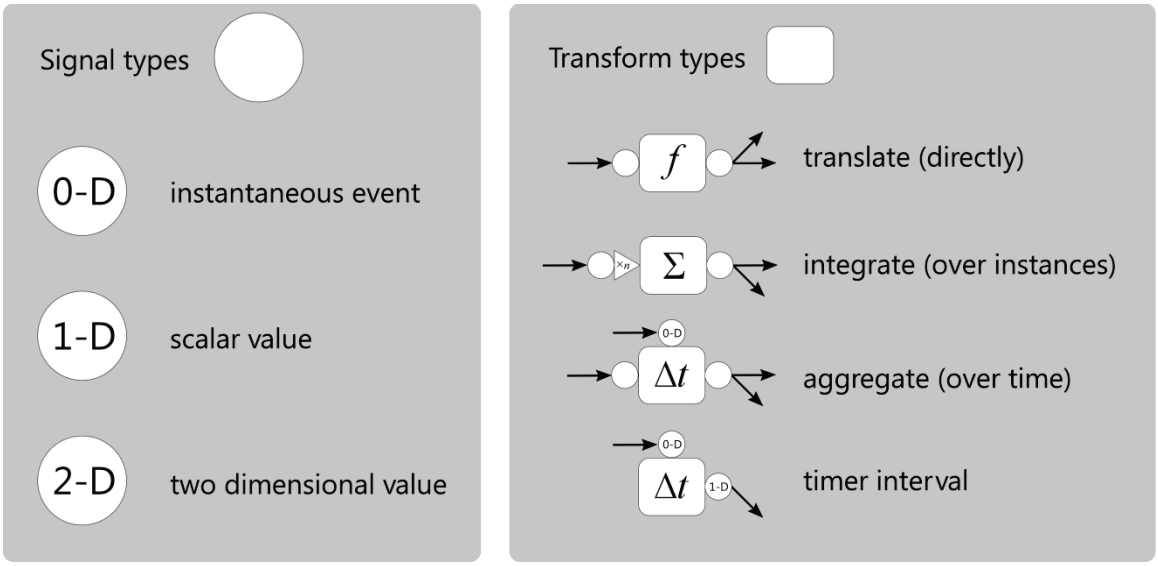


Figure 3-5: Description of the symbols used in the diagrammatic representation of control signals to represent signal types and transform types.

visually composing various input devices to generate alternate control signals but did not present it in the context of a generalized design-space of control signals.

3.2.1 Description of the Diagrammatic Representation of Signal Types and Transforms

Figure 3-5 describes the symbols that are used to represent the various types of control signals and control signal transforms. The signal types—which are meant to encompass both the elemental control signals that input devices can generate as well as the elemental parameters that applications expect—are abstracted to be zero-dimensional (0-D), one-dimensional (1-D), or two-dimensional (2-D). Zero-dimensional signals represent instantaneous and discrete impulse-like signals, such as those generated by a switch being pressed or a timer event firing. One-dimensional signals represent signals that can take on a real-number value, such as the position of a slider or a dial control. Two-dimensional signals represent a pair of real-number values, such as the coordinate of a point on a plane provided by a digital stylus. Three-dimensional signals, which are less common in today's interfaces, are omitted in the interest of keeping the size of the representation concise, but they can easily be accommodated by extrapolating the representations used for the lower dimensions.

The transform types represent the ways in which an input signal of a particular dimensionality or a set of input signals can be transformed into another input signal, possibly of a different dimensionality. They correspond roughly to Mackinlay's *composition operators* (Mackinlay et al.

1990), with the exception of the "aggregate" and "timer interval" transforms, which represent temporal operations. The "translate" transform corresponds to Mackinlay's connection composition operator, and represents operations that take a single input signal source and apply some numeric transform that results in a new value with either the same dimensionality as the original signal (indicated in the symbol by an outgoing horizontal arrow) or a reduced dimensionality (indicated in the symbol by an outgoing arrow towards the upper-right). For example, converting a 1-D slider value from one scale to another is an example of the former type, while checking a 1-D slider value against some threshold and outputting a 0-D signal whenever the threshold is exceeded is an example of the latter type. The "integrate" transform corresponds to Mackinlay's merge composition operator, and it represents operations that takes multiple input signals of a particular dimensionality, and outputs a new value with either the same dimensionality as the original signal or a higher dimensionality (indicated in the symbol by an outgoing arrow towards the lower-right). For example, taking in a set of 0-D signals from specific keys on a keyboard to generate a new 0-D signal to indicate when all of them are pressed at the same time (as with keyboard combinations) is an example of the former type, while combining two 1-D signals and generating a new 2-D signal is an example of the latter type. The "aggregate" transform, which does not have a direct analog in Mackinlay's model, represents temporal operations that process values received from one primary input signal source over a period of time delineated by successive inputs on the secondary 0-D input signal source (indicated in the symbol by a 0-D signal above the box). Examples of this transform include calculating the rate of multiple 0-D key presses or calculating the 2-D displacement of a 2-D input over a fixed window of time. Finally, the "timer interval" transform is a special instance of the "aggregate" transform that outputs the elapsed time (a 1-D signal) between two successive 0-D signals without any other primary input signal sources.

Figure 3-6 presents a list of the most basic instantiations of each transform type that can transform an input signal of one type into another, possibly changing the dimensionality. The figure illustrates how control signals of lower dimensionality can be composed to provide control signals of higher dimensionality, as well as transformations in the other direction in which control signals of lower dimensionality can be extracted from those of higher dimensionality.

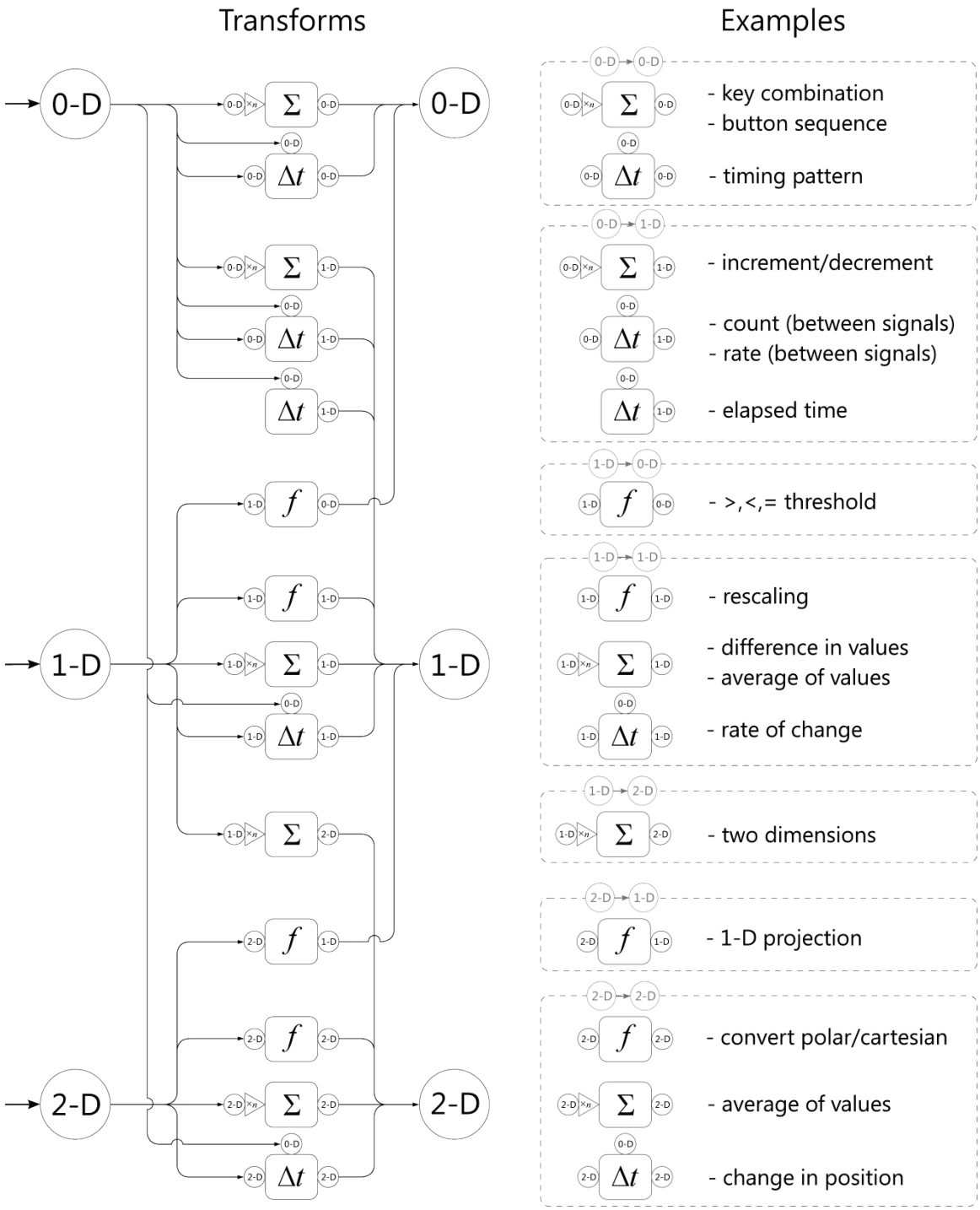


Figure 3-6: List of the basic control signal transforms and their examples for transforming among 0-D, 1-D, and 2-D signals. Corresponding transforms for higher dimensions can be derived by extrapolation, but are not shown here in the interest of clarity.

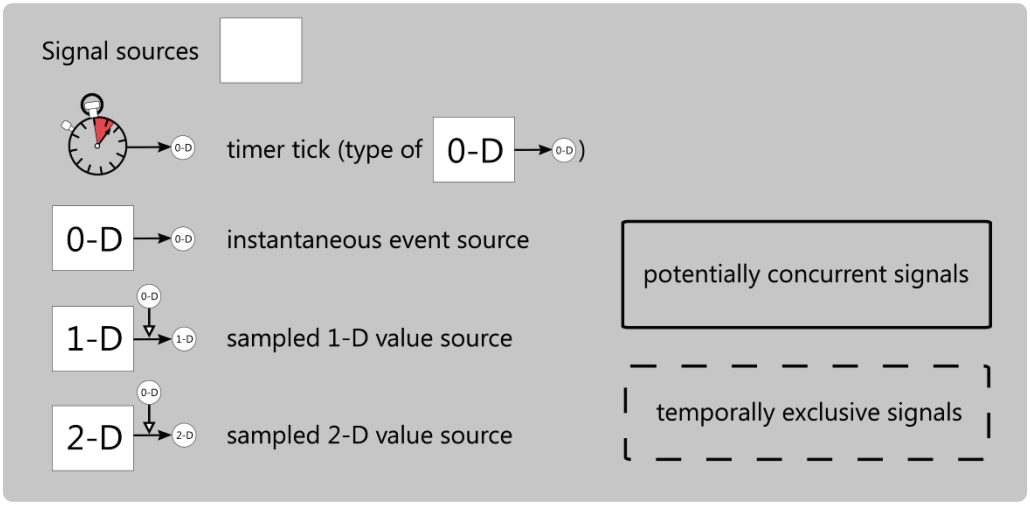


Figure 3-7: List of symbols used to represent signal sources of various dimensionalities. The timer tick source is a special instance of the 0-D signal source that is frequently used as a trigger, such as for sampling inputs from continuously-valued sources such as 1-D and 2-D signal sources. The solid border will be used to group control signals that are "potentially concurrent," meaning that they can be generated simultaneously. The dashed border will be used to group control signals that are "temporally exclusive," meaning that only one of them can be generated at a time.

Figure 3-7 describes the symbols that are used to represent the original sources of the control signals. An input device is defined by a collection of such signal sources, and may also contain transforms that convert these original control signals into alternate forms. To indicate the temporal relationship among control signals, the set of signals are labeled as being *potentially concurrent* if they can be generated simultaneously and *temporally exclusive* otherwise. The solid border encloses potentially-concurrent signals, and the dashed border encloses temporally exclusive signals.

An example of a standard keyboard depicted using this representation is shown in Figure 3-8. In the case of the keyboard, the key-down and key-up 0-D signals for each key are temporally exclusive since one has to happen before the other, but those signals from different keys are potentially concurrent since they can be simultaneously triggered by different fingers. As is evident from the figure, the keyboard supplies numerous 0-D control signals that are potentially concurrent, but does not provide any signals of higher dimensionality directly. However, by employing the appropriate transforms from Figure 3-6, the keyboard can be made to supply 1-D and even 2-D control signals, as in the case of Windows' MouseKeys (Figure 3-9), in which each of the eight keys on the numeric keypad (excluding the 5-key) is mapped to the corresponding radial direction, and the pointer speed is determined by how long the keys are held down.

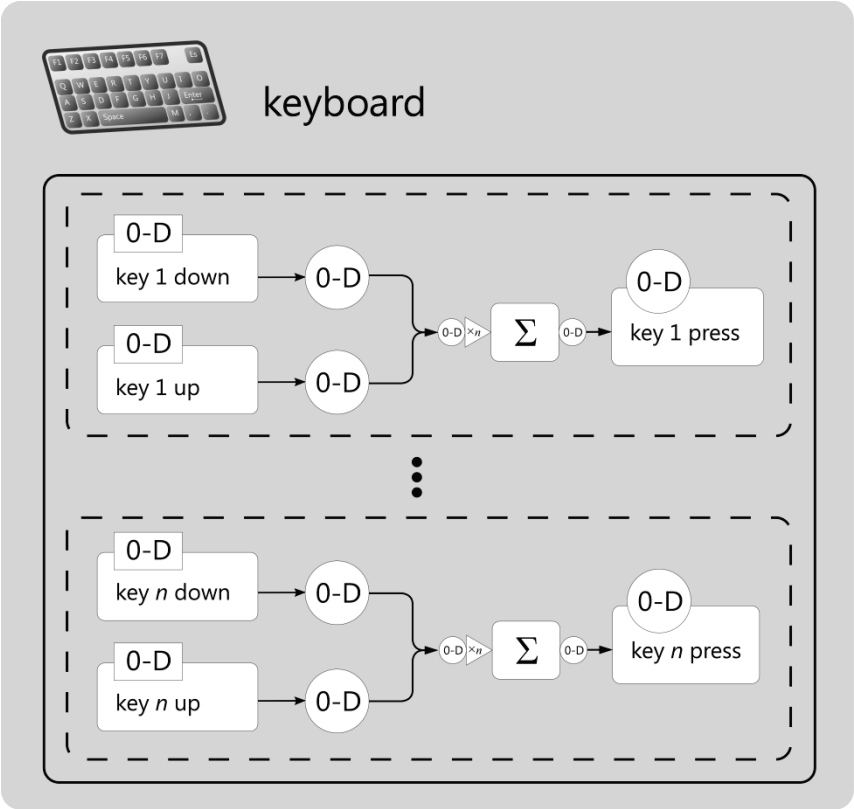


Figure 3-8: An example of the diagrammatic representation of the control signals generated by a conventional keyboard. The keyboard can generate numerous potentially concurrent 0-D signals.

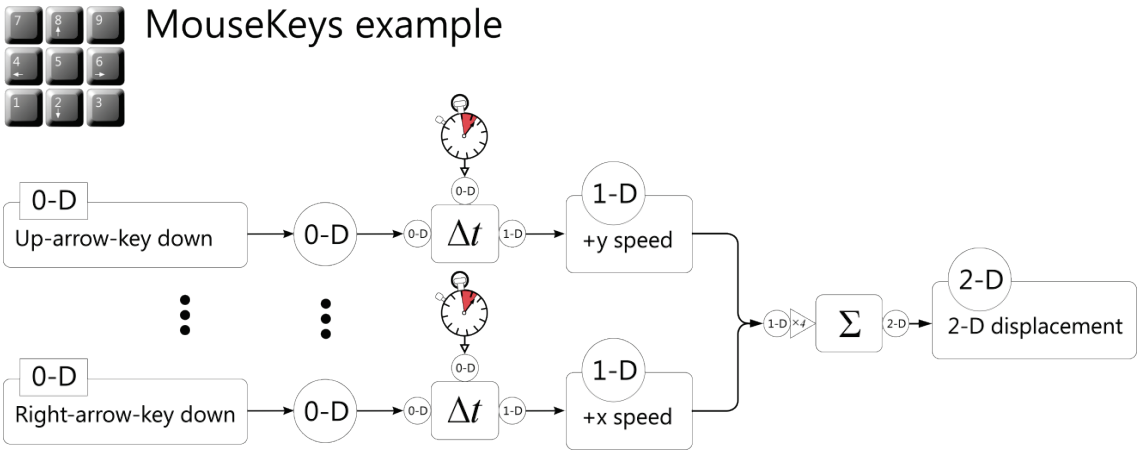


Figure 3-9: An example showing how Windows's MouseKeys feature transforms the 0-D control signals from the keyboard into 2-D application parameter for mouse pointer control.

3.2.2 Comparison of Voice-based Input Modalities to Keyboard and Mouse

Figure 3-10 presents the diagrammatic representation for the speech input modality. Similar to the keyboard, the speech input modality only directly provides 0-D control signals, but in this case, all control signals are temporally exclusive.

Figure 3-11 shows the corresponding representation for a typical mouse with three buttons and a scroll-wheel. It consists of three paired 0-D signals that make up the corresponding button events, a pair of 0-D signals from the scroll wheel which can be converted into a 1-D signal, and a 2-D signal representing the planar displacement of the mouse device, sampled at regular intervals. While all of the composed signals are potentially concurrent with the 2-D signal, the mouse only provides one 1-D signal.

Figure 3-12 shows the representation for the current version of the Vocal Joystick engine. It consists of two 0-D signals from the discrete sounds, and a total of eleven 1-D signals that include pitch value, power value, and the nine vowel sound probabilities. The vowel probabilities are temporally exclusive since they form a probability distribution and thus are interdependent. The figure also shows the composition that leads to the 2-D signal that is used in the Vocal Joystick pointer control application. In this case, the temporally exclusive vowel probabilities are integrated to produce a 2-D polar coordinate representing the overall vowel-probability vector. Its direction (i.e., polar angle) is then combined with the power level to produce the 2-D displacement-vector signal.

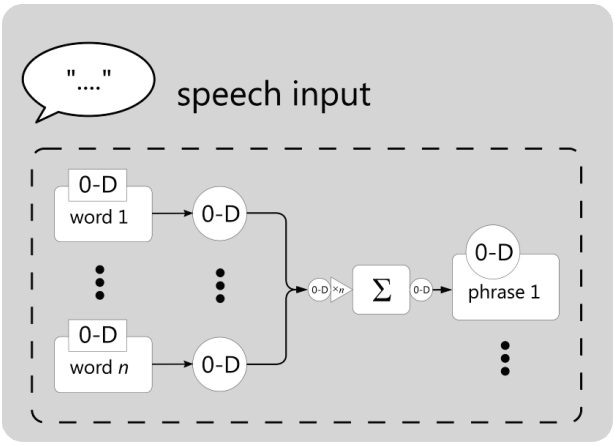


Figure 3-10: An example of the diagrammatic representation of the control signals generated by speech-based input. Similar to the keyboard, speech-based input can generate numerous 0-D signals, but they are all temporally exclusive.

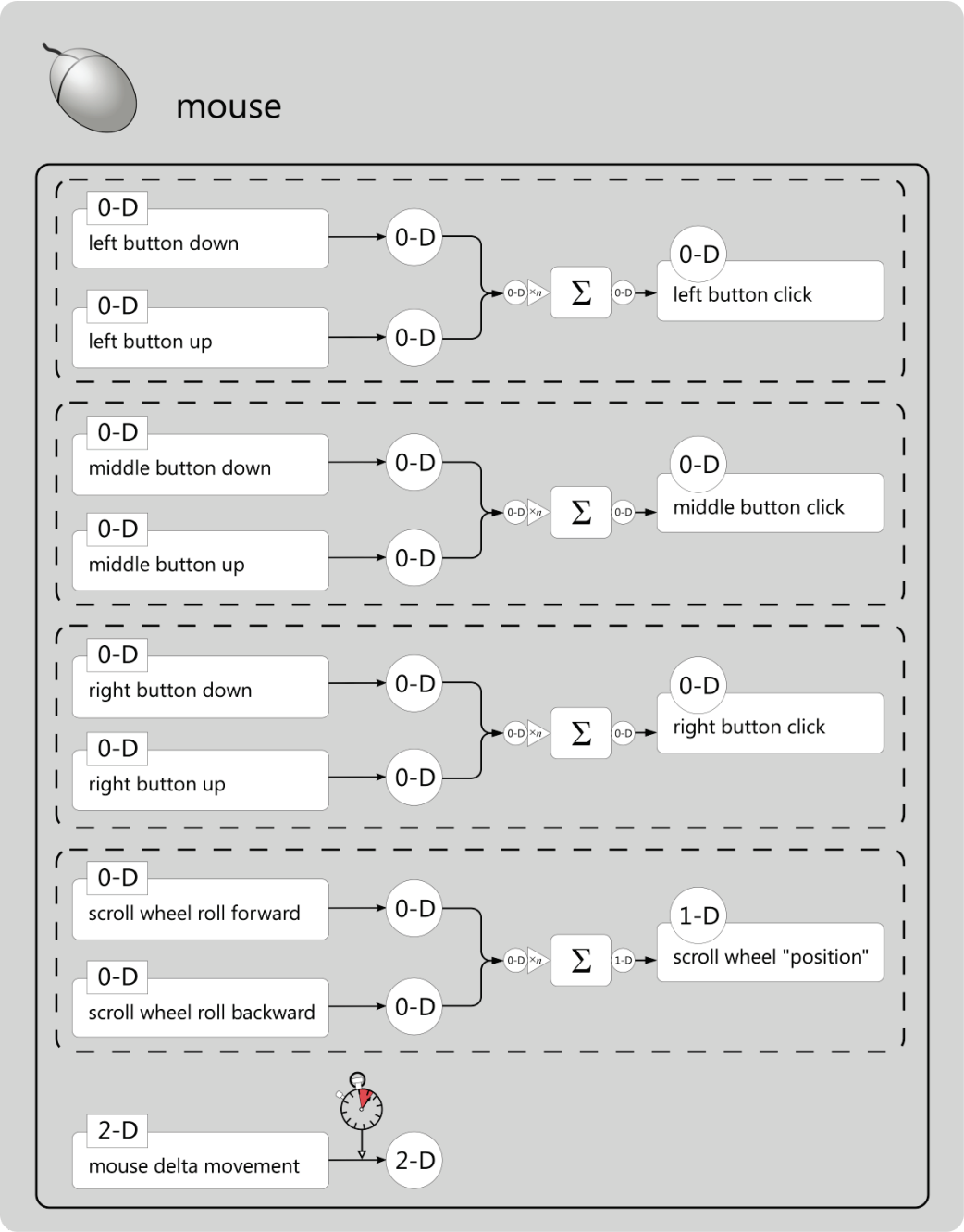


Figure 3-11: An example of the diagrammatic representation of the control signals generated by a conventional mouse. The mouse can generate several 0-D signals as well as a 1-D signal and a 2-D signal concurrently.

What is particularly interesting in the case of the Vocal Joystick engine compared to the mouse is that at the core level, there can be up to three potentially concurrent 1-D signals provided natively by the engine—the pitch level, the power level, and any one of the vowel probability values—and

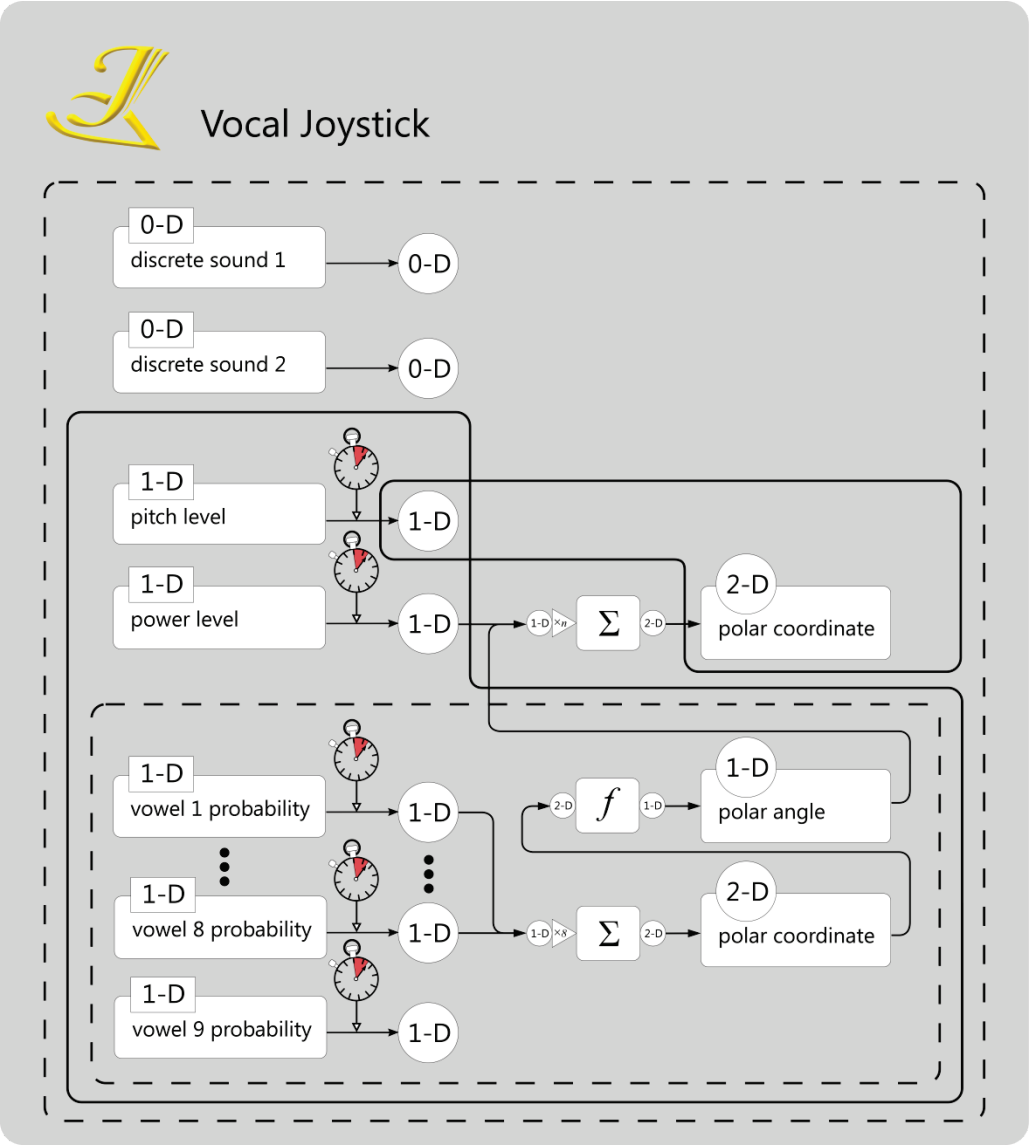


Figure 3-12: An example of the diagrammatic representation of the control signals generated by the Vocal Joystick engine. Unlike the mouse, the Vocal Joystick engine can only generate one 0-D signal at a time, but can generate several 1-D and 2-D signals concurrently.

that there are eleven distinct 1-D signal sources available. This affords the Vocal Joystick engine a wide range of possibilities in the control signals that it can provide by utilizing the transforms shown in Figure 3-6. In fact the set of voice gestures that was introduced in Chapter 6 is an example of this possibility.

3.3 Vocal Joystick Engine Architecture and API

The Vocal Joystick engine has undergone major architectural evolution, and various design decisions have been made at each stage to make the engine more amenable to offering flexible and generalizable input to various applications. The following sections present the key enhancements that have been made since the first version of the engine, and describe the type of data exposed through the API to potential client applications.

3.3.1 Evolution of the Vocal Joystick Engine Architecture

The first version of the Vocal Joystick engine was written in C++, and was embedded directly in the original Vocal Joystick pointer control application that ran only on the Linux platform. Figure 3-13 shows the structure of the first version of the Vocal Joystick engine. While it served to demonstrate the capabilities of the Vocal Joystick, its setup as a monolithic application made it difficult to build new applications and to extend the functionality of various modules within the engine. Its requirement for the Linux platform also limited the number of users who could run it on their own computer, as the majority of personal computers operate on Windows or Macintosh operating systems.

Throughout the course of my dissertation research, I have re-architected the Vocal Joystick engine to be a cross-platform, modular, and reusable library, and to provide added functionality through a managed-code extension library. The structure of the current version of the Vocal Joystick engine and its associated libraries is shown in Figure 3-14.

The first major improvement over the initial version of the engine is the support for multiple platforms, including Windows and Macintosh operating systems. By replacing the original

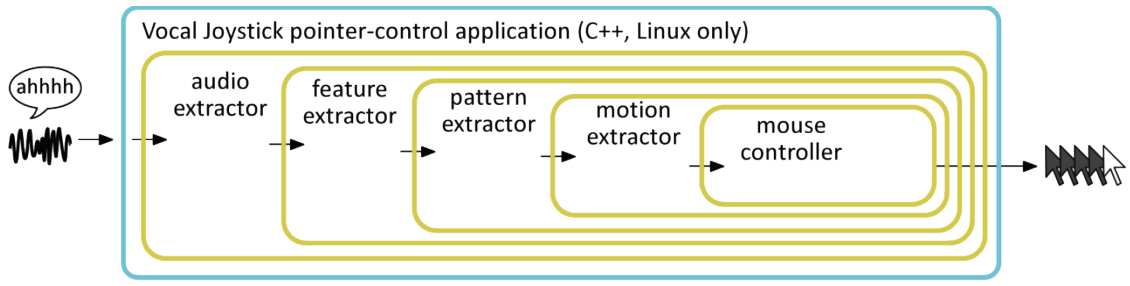


Figure 3-13: High level structure of the first version of the Vocal Joystick engine and its containing pointer control application. The engine was embedded within the application as a single, monolithic pipeline, and ran only on the Linux platform.

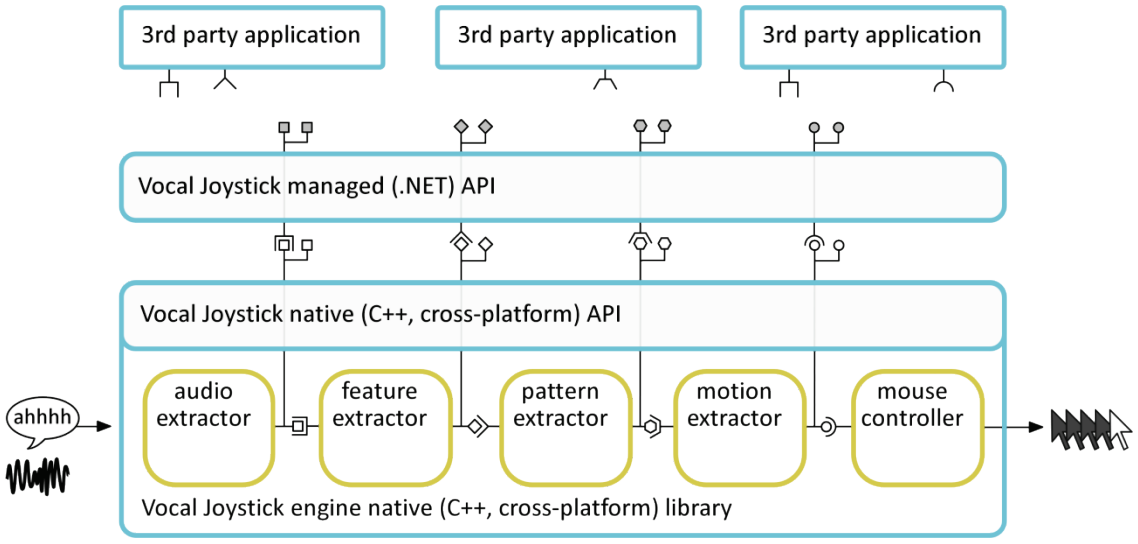


Figure 3-14: High level structure of the current version of the Vocal joystick engine and its associated libraries. The engine has been encapsulated in a cross-platform, modular library that offers an API enabling third-party applications to take advantage of its functionality. A managed .NET API has also been created to wrap the native C++ library, streamlining the process for high-level user interface-oriented languages and technologies such as C# and WPF to incorporate the Vocal Joystick engine functionality as well.

platform-dependent audio input code with a cross-platform audio library called PortAudio,[1] I enabled the Vocal Joystick application to run on Linux, Windows, and Macintosh operating systems. Since the mouse pointer control code was also platform-specific, different mouse controller modules for each operating system were also created and a build process was set up that integrates the appropriate version of the mouse controller module depending on the chosen target platform. The compatibility with the Windows platform was of particular importance given the prevalence of that operating system among general computer users as well as among users of accessible technologies.

The next enhancement that I made to the original Vocal Joystick engine architecture was a major overhaul of the single pipeline structure to create a multithreaded, modularized, event-driven library. Because the original architecture was single threaded, the entire Vocal Joystick processing pipeline—starting from the audio data fetching loop to the mouse pointer controller—was executing on the main user interface thread. This made it impractical to integrate the Vocal Joystick functionality into any other application that wished to provide additional interactive interface components. To solve this issue, I modified the engine to reflect a producer-consumer

[1] http://www.portaudio.com/

thread model, isolating the audio extractor within its own producer thread and the rest of the processing pipeline into a different consumer thread, both of which were separated from the user interface thread.

The engine has also been modularized such that each processing unit was encapsulated from the other units, and the communication mechanism between the units was converted to an event-based callback model instead of direct function calls. This was a key step in converting the Vocal Joystick engine from a single executable into a reusable library, making it possible to easily create new applications based on the Vocal Joystick engine by subscribing to events from the desired modules. Finally, I created a .NET managed-code wrapper around the native library, vastly simplifying the process of creating new applications that take advantage of the Vocal Joystick engine by making it possible to use programming languages and development tools that support rapid user interface development such as C#, Windows Presentation Foundation, and Visual Studio .NET. The outcome of the re-architecting effort described above is the current collection of Vocal Joystick engine libraries that consist of the native C++ library and a managed .NET wrapper library.

3.3.2 Description of Vocal Joystick Engine Modules

This section provides additional technical details regarding the nature of the output generated by each module within the Vocal Joystick engine that is made available to client applications through the event-callback mechanism. A more in-depth technical description of the core Vocal Joystick engine and its algorithms can be found elsewhere (Malkin et al. 2010).

Audio extractor module

The starting point of the Vocal Joystick engine processing-pipeline is the audio extractor module, as shown in Figure 3-14. The audio input from the microphone is sampled at 16 kHz, and the audio extractor module raises an event every 10 milliseconds (100 Hz), making available to subsequent modules a frame of audio data that contains the most recent 40 milliseconds of audio samples[2]. Most clients of the library may not require such low-level access to the raw samples, but it is useful to be able to tap into this stream—for instance, to save the incoming audio to file.

[2] A frame step of 10 milliseconds is a common value used in modern speech recognition systems. A frame size of 40 milliseconds was used to provide additional data for pitch estimation.

Feature extractor module

The feature extractor module receives each audio frame as they are generated by the audio-extractor module, and computes basic audio features such as the average power, pitch, and Mel-frequency Cepstral Coefficients (MFCCs) over the frame. This set of information, also reported every 10 milliseconds, may be used by the clients to generate real-time graphical feedback on the user interface, such as in a volume level meter. Up to this point in the processing pipeline, the functionality and the extracted information are not much different from those of typical speech recognition systems. One notable difference, however, is that the Vocal Joystick library exposes this information to client applications through its API, while most speech recognizers do not.

Pattern extractor module

The next module, the pattern extractor module, represents the core of the Vocal Joystick engine that distinguishes it from typical speech recognition systems. The two main functions performed by the pattern extractor module are vowel classification and discrete sound recognition. While both the vowel classifier and the discrete sound recognizer receive audio features from the feature extractor every 10 milliseconds, they only perform their functions and generate an output when a certain set of criteria is met. For the vowel classifier, the classification process only takes place when voicing is detected, indicated by the presence of pitch. The discrete sound recognizer initiates its processing only when a silent frame is observed, and if the length-constrained set of frames since the previous silent frame started with an unvoiced sound.

The default vowel classifier uses a two-layer multi-layer perceptron (MLP) that takes as input the MFCC features from the most recent frames, and outputs a set of probabilities, one for each of the nine vowel sounds recognized by the Vocal Joystick engine, representing the likelihood of the observed vocalization corresponding to each sound. This set of probabilities is also made available to library clients through the API. There has been significant research conducted by one of our project members, Jonathan Malkin, in creating a new classifier family that can produce smoother probabilities across vowel sounds (Malkin & Bilmes 2009), resulting in the ability for a user to vocalize a gradual transition from one vowel sound to another and have that be similarly reflected in the probability output. The default vowel classifier parameters have been trained on vowel sound samples collected from 10 speakers to serve as the baseline speaker-independent model. The Vocal Joystick engine also includes the ability for each user to tune the system to

better recognize their voice by performing an adaptation procedure, which involves collecting two seconds of the user's vocalization for each of the sounds used in the selected vowel mode.

The discrete sound recognizer uses a hidden Markov model (HMM) based on 42 phonemes trained on the TIMIT corpus and can recognize any word or sound represented as a sequence of those phonemes. The Vocal Joystick engine provides a dictionary file in which the phoneme sequence for each sound to be recognized by the discrete sound recognizer can be specified. While it is possible to use this mechanism to make the Vocal Joystick engine recognize many words and phrases, the default setup includes just two discrete sound definitions in this file—the monosyllabic, consonant-only sounds "ck" and "ch"—for several reasons. First, since the discrete sound recognizer is running concurrently with the vowel classifier and the discrete sound processing only begins after the end of the utterance, any vowel sound that is uttered as part of the discrete sound will be processed by the vowel classifier and possibly cause unwanted actions to execute if the sound was part of a discrete sound. For example, if the discrete sound was the word "click," while the user utters that word, the [i] sound in the word may cause the mouse pointer to move slightly to the left, possibly causing the intended target to be missed. For this reason, the default set of discrete sounds do not contain any voiced sounds. Second, the number of discrete sounds was minimized to keep the recognition error rates low and to leverage the power of existing speech recognizers for processing any additional command words or phrases. Finally, the default discrete sounds were kept short in order to minimize the delay between vocalization and invocation of the associated command. The outputs of the discrete sound recognizer that are exposed through the API include the probabilities associated with each discrete sound, and the discrete sound that was actually recognized (if the highest probability exceeds a preset threshold).

Motion extractor module

The primary function of the motion-extractor module is to convert the vowel probabilities generated by the pattern extractor module into a two-dimensional representation based on the spatial mapping shown in Figure 3-4. An overall *vowel-direction vector* is determined by scaling the unit vectors corresponding to each vowel sound's radial direction by their associated probabilities and summing them up. In the case of 4- or 5-vowel modes, only the vowel sounds along the cardinal directions are considered, and in all cases the middle vowel sound does not affect the vowel-direction vector. In addition to the vowel direction, a "magnitude" along that

direction is also calculated based on the power of the vocalization. Due to the fact that mouse pointer control was the original application for which the Vocal Joystick engine was developed, this magnitude value inherently represents the speed of the pointer movement, and the function that maps the power to this speed was chosen to be appropriate for this usage. The transfer function used in the current version of the Vocal Joystick engine to map power to speed is of the following form:

$$s = \alpha e^{f(p)} + \beta \tag{3}$$

where s is the speed (internally interpreted as pixels per second), p is the observed power value, e is Euler's constant, $f(p)$ is currently a square-root function, and α and β are scaling factors.[3]

The mouse-controller module simply takes the vowel direction vector and the speed magnitude output by the motion extractor module and moves the mouse pointer at the corresponding angle and speed. The mouse controller module also receives output from the discrete-sound recognizer in the pattern extractor module, and issues a left-mouse button click whenever the "ck" sound is detected, and toggles the state of the left mouse button whenever the "ch" sound is detected.

The modularized, event-driven design of the Vocal Joystick engine described above makes it an ideal library on which to build a wide variety of audio applications—from those that process raw audio samples (by registering a callback at the audio-extractor module) to those that take full advantage of the Vocal Joystick engine's non-speech vocalization processing (by registering callbacks at the pattern extractor module). In fact, there have been a number of applications that have been developed using the Vocal Joystick library, both by the author as well as by others. They are presented in detail in Chapter 5, Chapter 6, and Chapter 8.

3.4 Overview of the Vocal Joystick Mouse Pointer Control

While the control signals output by the Vocal Joystick engine can be used to control a wide range of programs and devices, one of the first applications of the Vocal Joystick engine was the control of the mouse pointer. As mentioned in Chapter 2, mouse pointer control—especially continuous steering of the pointer—is a big challenge for existing speech-driven methods.

[3] The transfer function presented here has been simplified from its original form for the sake of clarity. A detailed description of the function can be found elsewhere (Malkin et al. 2010).

The Vocal Joystick pointer control application, as the name suggests, operates very similarly to a physical joystick. With a typical isometric physical joystick such as the IBM TrackPoint, the pointer can be moved along a desired trajectory by applying force on the joystick in the corresponding direction. The greater the force that is applied on the joystick, the greater the velocity of the pointer movement. Similarly with a Vocal Joystick, user's vocalization of one of the radial vowel sounds shown in Figure 3-4 is analogous to a force being applied on the joystick in the corresponding direction, and the power of the vocalization corresponds to the magnitude of the movement velocity. Just as it is possible with a physical joystick to smoothly vary the movement direction and speed of the pointer by gradually changing the magnitude and the direction of the applied force, the user can achieve the same effect using the Vocal Joystick by gradually varying the sound and the volume of vocalization. This ability to smoothly modulate both the direction and the magnitude of movement speed simultaneously is what distinguishes the Vocal Joystick from the other existing voice-based pointer control methods.

Two types of mouse button actions are also available through the Vocal Joystick pointer control: left-button click, and left-button toggle. The left-button click action is executed by uttering the discrete sound "ck." If the operating system's double-click interval threshold is set high enough, it is possible to perform a double-click action by simply uttering the discrete sound twice in succession. The left-button toggle action toggles the pressed state of the left mouse button, and can be executed by uttering the discrete sound "ch." This enables the execution of dragging tasks, by first uttering "ch" to change the left-button state to down, followed by vocalization of vowel sounds to drag the item to the desired location, and finally releasing the button by uttering "ch".

3.5 Applications Built Upon the Vocal Joystick Engine Library

The re-architected Vocal Joystick library has been used to build a number of voice-enabled applications. Chapters 5 through 8 present in-depth discussions of four primary applications. In this section, I introduce some of the other representative applications that have been built both by me as well as by others.

3.5.1 VoiceLabel

VoiceLabel (Harada et al. 2008b) is a system to facilitate mobile labeling of sensor data for building sensor-driven activity-recognition applications. In this system, the primary user is the data collector who carries a mobile sensor as they engage in various activities of interest and later

needs to label the segments of the collected sensor data with the corresponding activity. VoiceLabel seeks to eliminate various obstacles to labeling the sensor data in-situ while the data collector is engaged in the activity by offering hands-free, speech-based input for annotating the beginning of each activity. The usage scenario is as follows: users wear a portable microphone connected to the mobile sensor device, and as they engage in various activities, they utter the name of the activity that they are about to begin, which gets recognized by the speech recognizer and the recognized label gets applied to that time point in the sensor stream. To reduce the amount of false positives and misrecognitions, the system requires users to utter a filled-pause utterance ("uhhh") before saying the name of the activity. Our evaluation showed that the use of filled-pause utterance to demarcate the beginning of the desired utterance effectively reduced the number of segmentation errors by 30%. Filled-pause detection is enabled by the Vocal Joystick library, while speech recognition is performed by the Windows Speech Recognizer.

3.5.2 VoiceVoiceRevolution

VoiceVoiceRevolution (Figure 3-15) is a game that I conceptualized and had built by a University of Washington Computer Science undergraduate student named Justin Pai whom I

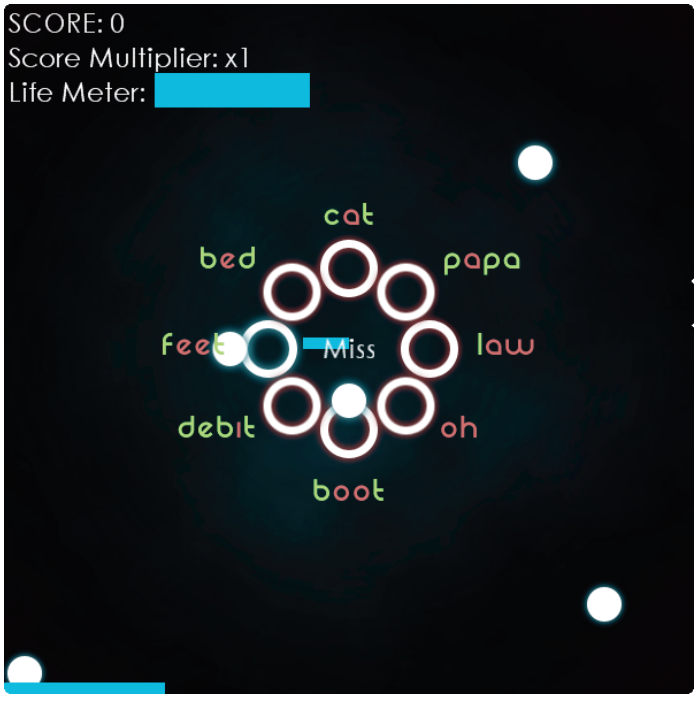


Figure 3-15: Screenshot from the game VoiceVoiceRevolution, created by a Computer Science undergraduate Justin Pai under my supervision, using the Vocal Joystick library.

advised. I drew inspiration from the popular beat-matching games such as Dance Dance Revolution and Rock Band, in which a sequence of visual cues are presented to the player who must respond in a timely manner by executing corresponding actions such as stepping on a directional pad or striking a drum pad. One of the original motivations for creating this game was to enable new users of the Vocal Joystick to learn the directional vowel mappings while also gaining entertainment value in the process. In VoiceVoiceRevolution, visual cues in the form of white discs appear from the eight cardinal and ordinal directions moving towards the center at a fixed speed. The player's objective is to utter the sound corresponding to the direction of the disc at the moment when it overlaps with the corresponding ring located about the center of the screen. The game offers multiple levels with differing speeds and number of vowels to support progression in skill. The game utilizes the Vocal Joystick library's vowel detection functionality, as well as the integrated speech input functionality for navigating the menus.

3.5.3 VoiceRacer

VoiceRacer (Figure 3-16) is another game that I conceived primarily to serve as a way for novice Vocal Joystick users to become accustomed to the smooth steering capability of voice input. The

Figure 3-16: Screenshot from the game VoiceRacer, created by a Computer Science undergraduate Matthew Davis under my supervision, using the Vocal Joystick library.

game was built by a University of Washington Computer Science undergraduate student named Matthew Davis whom I advised. In the game, the player controls a marble through a race track to reach the goal line as quickly as possible while avoiding possible obstacles such as falling off the edge of an unguarded track. The control is essentially identical to the Vocal Joystick pointer control, except that instead of the marble moving, the background track moves in the opposite direction to simulate a top-down racing perspective in which the marble always stays at the center of the screen. The volume of the vocalization is mapped to the speed of the marble. Just as in the VoiceVoiceRevolution game, VoiceRacer supplies multiple levels with varying track complexities and required number of vowels. While the VoiceVoiceRevolution sought to develop the player's memory of the vowel direction mapping, VoiceRacer seeks to develop the player's fluid steering ability.

3.5.4 VoiceBot

The Vocal Joystick library has been used not only to manipulate software interfaces, but physical devices as well. In VoiceBot (House et al. 2009), the Vocal Joystick library was used to control a physical six-degrees-of-freedom robotic arm. Since the robot operates in a three-dimensional space and permits six degrees of freedom, a novel mapping between the Vocal Joystick control signals and the robot control parameters had to be devised. Figure 3-17 shows the two operational

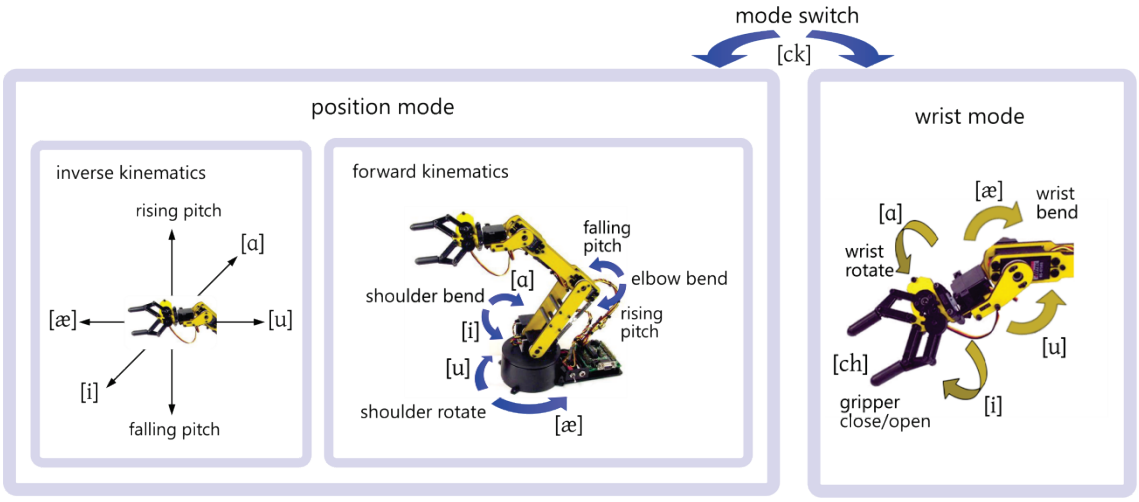


Figure 3-17: Two operational modes and control mappings used in VoiceBot to control a physical six-degrees-of-freedom robotic arm. Position mode enabled gross positioning of the gripper, and wrist mode enabled fine alterations of the gripper orientation. Two types of mappings were investigated for the position mode: inverse kinematics and forward kinematics.

modes supported by the VoiceBot, along with the mappings of vocal parameters to control parameters. The position mode enabled gross positioning of the gripper, while the wrist mode enabled fine adjustments of the gripper orientation. Two different types of mappings were investigated for the position mode. The inverse kinematics mapping enabled direct manipulation of the gripper as a point in 3-D space, with the 2-D Vocal Joystick vowel mapping controlling the motion along the horizontal plane, and the pitch changes controlling the motion along the vertical axis. In the forward kinematics mapping, pairs of vowel sounds and pitch changes were assigned to each rotational joint. The discrete sound "ck" was used to switch between the position mode and the wrist mode, and "ch" to open and close the gripper.

3.6 Summary

Although the Vocal Joystick application has been designed initially to control the mouse pointer in a two-dimensional continuous space, the underlying engine's capability to classify vowel sounds and extract power and pitch information was intended to be generalizable towards various other controls that may not have direct mappings to a two-dimensional space. For example, any subset of the nine vowel sounds may be used to simulate distinct buttons for selecting among up to nine choices. One may also use only a pair of vowel sounds to simulate a one dimensional slider. Volume and pitch can also be used to manipulate a continuous value. Due to this flexibility in the application of the Vocal Joystick signal, once users master the directional vowel sounds and the volume/pitch control, they will then be able to extend that skill beyond pointer control to a variety of input controls. The key characteristics of the Vocal Joystick engine and its capabilities are outlined below:

- *Quick response*: Unlike word-based commands such as "move mouse left" in which the user has to complete the utterance before the system can act upon it, the Vocal Joystick engine can process and respond to the vocalized sound every 10 milliseconds.
- *Continuous modulation of a parameter value*: Due to the rapid sampling rate and the ability to capture continuously-varying features such as pitch and power, the Vocal Joystick engine can provide these values as inputs into applications for manipulating continuous parameter values.
- *Simultaneous multidimensional control*: As human vocalization allows for modifying the vowel sound, power, and pitch at the same time, these parameters can be processed by the Vocal Joystick engine to manipulate multiple parameters simultaneously.

- *Transferable 2-D mapping*: Once the 2-D mapping of the Vocal Joystick is learned, it can be transferred to other analogous mappings such as a vocal version (Harada et al. 2007b) of the marking menu (Tapia & Kurtenbach 1995), to more diverse categories of applications such as controlling a robotic arm (House et al. 2009) or using only one of the dimensions for linear slider control.

Having described the Vocal Joystick engine and the nature of the vocal characteristics and control signals involved, I present in the following chapter a number of in-depth user studies I have conducted to uncover the fundamental performance characteristics of non-speech vocalization as an input modality.

Chapter 4
Performance Characteristics of Non-Speech Vocalization*

There has been little prior research that systematically investigates the fundamental capability of non-speech vocalization as a computer input modality. I have conducted a number of user studies to determine the performance characteristics of non-speech vocalization for basic user interface manipulations, focusing on pointer control and discrete input. One of the challenges in conducting such studies was the fact that there was no group of "experienced Vocal Joystick users" whom I could study since the Vocal Joystick had not yet been made widely available. As a result, I conducted the studies in multiple phases, with each phase focusing on a specific subset of the user groups. The four phases of the user studies are categorized as follows:

- Evaluation of the Vocal Joystick with expert users (research team members)
- Comparative evaluation of the Vocal Joystick with novice users
- Longitudinal study of the learning process of users with and without motor impairments
- Comparative evaluation of discrete input versus speech

* Parts of this chapter are adapted from (Harada et al. 2006, 2008a, 2009).

The following sections describe each of the above four phases in further detail.

4.1 Evaluation of the Vocal Joystick Pointer Control with Expert Users

To better understand what the optimal performance level of Vocal Joystick pointer control may be with significant practice, I conducted a thorough evaluation (Harada et al. 2006, 2008a) with four Vocal Joystick experts using a Fitts' law task paradigm (Mackenzie 1992). The two specific goals of the study were to 1) determine whether Vocal Joystick pointer control can be modeled by Fitts' law (Fitts 1954), a widely adopted model of human motor performance for aimed movements, and if so, 2) to determine the model parameters that characterize the Vocal Joystick such that its *index of performance* (IP) can be compared against other well-studied pointing devices. Before I describe the motivation behind the study setup, I will briefly overview the key underlying concepts of Fitts' law.

4.1.1 Fitts' Law

Fitts' law provides a mathematical formulation of human motor performance based on Shannon's fundamental theorem of communication systems (Shannon & Weaver 1949). It models the human motor system as a channel with a certain bandwidth (measured in bits per second) that is used to transmit information in performing a movement task of a certain index of difficulty (measured in bits). The model assumes a fairly simple 1-D target acquisition task in which the goal is to move from a starting position to a target at a distance A (referred to as the amplitude) whose size is W (referred to as the width) along the *task axis*, or the line containing the shortest path from the starting point to the target. The original model relates the amplitude and width to the movement time (MT) in the following form:

$$MT = a + b \cdot \log_2\left(\frac{2A}{W}\right) \tag{4}$$

where the term $log_2(2A/W)$ is referred to as the index of difficulty (*ID*) of the task (measured in bits), and a and b are empirically determined regression coefficients that characterize the particular modality used. The inverse of the slope coefficient, $1/b$, is referred to as the index of performance (*IP*). This is the information capacity of the motor system involved (including both the human operator as well as the device used) measured in bits per second. Higher *IP* (or lower slope coefficient b) indicates that the particular motor system is more efficient at the target

acquisition task. A notable point is that the model predicts that the movement time should not be affected if the target amplitude and width are scaled by an equal factor.

Since the original formulation of Fitts' law, numerous refinements have been proposed to more accurately capture the underlying phenomenon. For example, the notion of *effective target width* (*We*) is often used to account for the variability in the subjects' actual speed-accuracy tradeoff and to calculate the corresponding *effective index of difficulty* (*IDe*) value for each task condition (Soukoreff & MacKenzie 2004). There have also been numerous alternate formulations of the index of performance measure, such as the *mean of means* formulation by Soukoreff and MacKenzie (2004) which expresses the value as $\frac{1}{y}\Sigma_{i=1}^{y}\left(\frac{1}{x}\Sigma_{j=1}^{x}\frac{IDe_{ij}}{MT_{ij}}\right)$, where y is the number of subjects and x is the number of movement conditions. Zhai et al. (2004) provides a detailed analysis of the benefits and limitations of these different formulations.

For my study, I chose to use a minor variation of the *ID* formulation (Mackenzie 1992) that has been shown to be more accurate (Cover & Thomas 1991), in which *ID* is defined as:

$$ID = \log_2\left(\frac{A}{W} + 1\right) \tag{5}$$

I also chose to use the original *1/b* formulation for *IP* since the other studies against which I wished to compare my results (Card et al. 1987; Epps 1986; Miniotas 2000; Radwin et al. 1990) also used this original formulation.

A visual summary of the concepts involved in Fitts' law is presented in Figure 4-1. While the original formulation of Fitts' law was in the context of one-dimensional target acquisition tasks, subsequent research has shown that the predictive power of Fitts' law applies to two-dimensional tasks (Accot & Zhai 2003; Grossman & Balakrishnan 2005; MacKenzie & Buxton 1992) as well as three-dimensional tasks (MacKenzie et al. 1987).

4.1.2 Motivation for the Study

There were several reasons why I tested the Vocal Joystick against the Fitts' law model. First, if it is determined that the Vocal Joystick can indeed be modeled by Fitts' law, it will enable the prediction of Vocal Joystick performance times for various target acquisition tasks. This is useful in determining whether the Vocal Joystick is usable in interacting with a particular interface that may have specific minimum performance requirements. Second, the model will allow us to

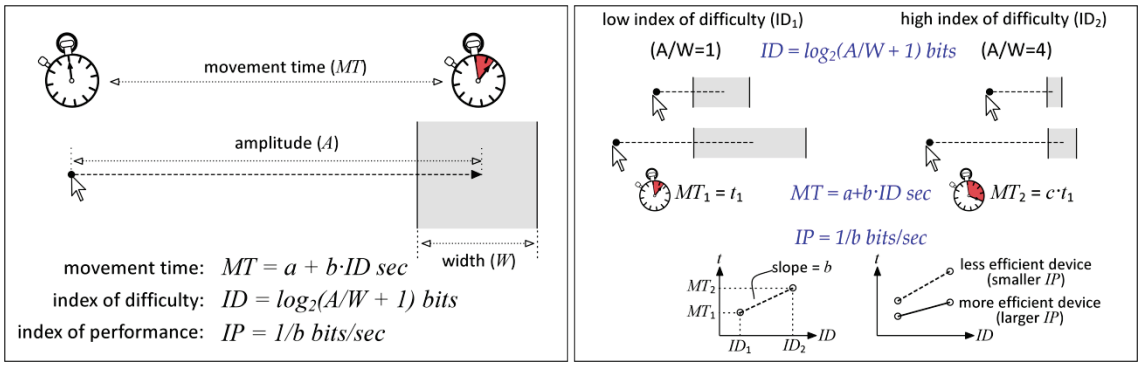


Figure 4-1: Visual summary of the concepts involved in Fitts' law. Fitts' law states that the time (*MT*) to acquire a target in a manual pointing task is directly proportional to the task's index of difficulty (*ID*), which in turn is dependent on the ratio of the target distance (*A*) to the target width (*W*). The index of performance (*IP*) of a pointing device can be determined by measuring various *MT*s at different *ID*s and taking the inverse of the slope of the regression line.

generalize results from future experiments with fewer conditions (given the same general population). Finally, and perhaps most interestingly, determining the relative ratio of the Vocal Joystick's index of performance to that of the mouse will allow the Vocal Joystick to be compared to various other input devices that have been analyzed with respect to the mouse (or any other well studied device) using Fitts' law experiments. As MacKenzie (1992) points out, variations across Fitts' law experiments in their setup and procedure makes comparison of the absolute values of *IP* across different experiments difficult. However, the relative values of the *IP*s among devices within a particular experiment tend to be consistent across different studies that compare those same set of devices. This suggests that analyzing the performance index of a new input device with respect to the well studied mouse can enable comparison of that new input device with other devices that have been studied relative to the mouse.

4.1.3 Study Participants

As the Vocal Joystick system had not yet been made available to the public at the time, the only experts were the four members of the research team, none of whom had any motor impairment. I believe that this population still provides a good estimate for the optimal performance. Regarding the possibility of the result being biased in favor of the Vocal Joystick when comparing it against the mouse, I ensured that the participants were instructed to perform the task as accurately and quickly as possible for both modalities, and the results also validate that there was no significant bias effect (see Section 4.1.7 for further discussion). My definition of an 'expert' user was based on the fact that all four participants were well accustomed to all of the vowel sounds used in the

Vocal Joystick, were very familiar with the speed control response behavior of the system, and have used the system over an extended period of time (more than a month).

4.1.4 Study Apparatus

The experiment was conducted on a Dell Latitude D600 laptop with 1.6 GHz Intel Pentium M processor and 512 MB of RAM running Windows XP. The LCD screen had a diagonal size of 14.1 inches and the display resolution was set to 1400×1050 pixels. The experiment was conducted in full screen mode, with the user's head situated about two feet from the screen. Input into the system was either an external mouse or a microphone. An external optical USB mouse was used for the mouse condition, and software acceleration for the mouse was turned off in the operating system to mitigate the potential confounding effect of control gain (Bohan et al. 2003). A Sennheiser Headset Microphone connected to an Andrea USB sound pod was used for the Vocal Joystick condition. Eight-way mode was used for the Vocal Joystick allowing for the greatest granularity in the radial movement directions.

4.1.5 Study Design

I conducted a fully-crossed within-subjects factorial study with repeated measures comprising the following factors and levels for a total of 192 conditions per participant:

- Modality (M) {Vocal Joystick, mouse}
- Index of difficulty (ID in bits) {2, 3, 4, 5}
- Target width (W in pixels) {12, 24, 32}
- Task axis angle (θ in degrees) {0, 45, 90, 135, 180, 225, 270, 315}

As the target distance (A) is fully determined if the index of difficulty and target width are given, it was not varied independently, and the following values were derived (A=36, 72, 84, 96, 168, 180, 224, 360, 372, 480, 744 and 992 pixels). The task axis angle of 0 degrees corresponded to the target positioned directly to the right of the starting point, while the task axis angle of 90 degrees corresponded to the target positioned directly above (towards the top of the display) the starting point. For each of the 192 conditions, the participants performed 3 trials. The trials were grouped by modality, and the order of the modality was counterbalanced across participants. Within each modality, the order of the conditions was randomized. Since the primary objective for this study was to ascertain the overall effect of the index of difficulty and target width on

movement time for each modality regardless of the vowel sound used, I aggregated the results across the eight task axis for each combination of (*M*, *ID, W*).

4.1.6 Study Procedure

At the beginning of each trial, a target bar of width *W* (and infinite "height", as the bar continued beyond the edge of the screen) was presented on the screen at a distance of *A*/2 from the center of the screen. The cursor was positioned directly opposite of the target across the center of the screen, also at a distance of *A*/2 from the center of the screen. The trial was initiated and the timer started as soon as the cursor moved away from its original position. The participants were instructed to attempt to acquire the target as quickly and accurately as possible. When the cursor was moved above the target, the target bar changed color to indicate that the cursor was above the target. The participant acquired the target by pressing the space bar under both modalities (in the case of mouse, they were told to use the hand unoccupied by the mouse). This was done to normalize the difference in the clicking modality between the two devices, since I was primarily interested in the movement time to the target.

4.1.7 Study Results

The aggregate error rates for the mouse and the Vocal Joystick were 1.4% and 2.3%, respectively. To test for model fit against Fitts' law, linear regression analysis was performed on movement time against *ID*. For both modalities, *ID* significantly predicted movement times ($p < .001$) and also explained a significant portion of variance in movement times ($p < .001$). Figure 4-2 shows the graph of the movement times for each modality plotted against the various *ID*s with linear regressions. In the figure, for each modality and within a particular index of difficulty, aggregate movement times for each of the three target widths were plotted as separate points. The fact that the three points within each *ID* are close to each other reflects the fact that the data follows Fitts' law, as change in width given a fixed *ID* should not affect the movement time (see Equation 5). The *IP* for the Vocal Joystick was 1.65 bits/sec, and for the mouse was 5.48 bits/sec. Revisiting the bias issue mentioned in Section 4.1.3, the *IP* value for the mouse I obtained is within the range of those reported by others (Card et al. 1987; Epps 1986; Miniotas 2000; Radwin et al. 1990). A presence of significant bias in favor of the Vocal Joystick would have probably manifested itself as a lower *IP* for the mouse, making the *IP* of the Vocal Joystick look significantly higher relative to the mouse.

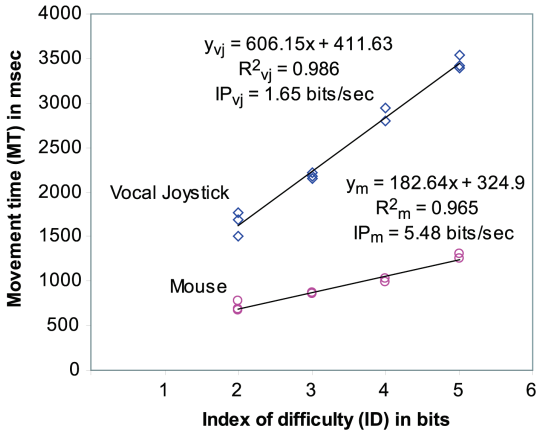


Figure 4-2: Result from the evaluation of the Vocal Joystick with expert users, showing the linear regression of mean movement time (*MT*) versus index of difficulty (*ID*) for the Vocal Joystick and the mouse. For each *ID*, mean *MT* for each of the three target widths are plotted individually. Inverse of the slope of the regression line represents the index of performance (*IP*).

The normalized *IP*s of the Vocal Joystick as well as several other devices that have been studied with respect to the mouse are shown in Table 4-1. Here, numbers less than 1 represent devices that are less efficient than the mouse, and those greater than 1 represent devices that are more efficient than the mouse. The result shows that the expert index of performance of the Vocal Joystick is roughly a third of that of a conventional mouse. Although the relative *IP* of the Vocal Joystick is about 25% lower than that of the conventional hand-operated velocity control joysticks, they are comparable to each other relative to the mouse. Given that the Vocal Joystick is an assistive input device that relies solely on voice and requires no manipulation by the hands, this is promising as it indicates that as the user gains proficiency, they can potentially approach the performance of a hand-operated joystick. Although the eye tracker and the ultrasonic head pointer have higher relative *IP* than the Vocal Joystick, the significantly lower cost of setup and

Table 4-1: Relative *IP*s across devices. Higher number represents more "efficient" devices for target acquisition.

Device	Relative *IP*
Mouse	1.00
Eye tracker (Miniotas 2000)	0.71
Ultrasonic head pointer (Radwin et al. 1990)	0.61
Joystick (isometric; rate-controlled) (Card et al. 1987)	0.43
Joystick (displacement; rate-controlled) (Epps 1986)	0.42
Vocal Joystick	0.30

the absence of reliance on specialized hardware make the Vocal Joystick a viable candidate device for hands-free continuous input. As further improvements to the Vocal Joystick engine are made to increase its accuracy and to make the mapping between volume and pointer movement better reflect the user's intent, the Vocal Joystick performance is expected to increase further and potentially surpass some of these alternate pointing devices.

This work is one of the first to show that a voice-controlled pointer movement follows Fitts' law. This finding allows researchers to apply the same predictive power of Fitts' law that has been used on a number of other pointing devices on the Vocal Joystick as well. In some sense, the Vocal Joystick index of performance obtained from this study provides a target level of performance that users can expect to reach with significant training, but does not necessarily indicate the maximum potential performance, as even the "experts" may continue to improve their performance with further practice, as well as enhancements in the Vocal Joystick control mapping leading to more effective controls.

4.2 Vocal Joystick Versus Speech-based Pointer Control Methods

The Fitts' law study with expert Vocal Joystick users described in the previous section provided a model of the Vocal Joystick's optimal performance characteristics as it pertains to target acquisition tasks. To better understand how the Vocal Joystick fares with novice users and also how it compares to other existing speech-based pointer control methods, I conducted a second comparative study involving participants with no prior Vocal Joystick experience (Harada et al. 2006, 2008a). The two speech-based pointer control methods that were chosen for comparison were Mouse Grid (MG) and Speech Cursor (SC) (introduced in Chapter 2), both features of the commercial speech recognition software *Dragon NaturallySpeaking*.

4.2.1 Study Participants

I recruited a total of nine participants (five males and four females) to participate in the informal comparative evaluation. None of the participants had prior Vocal Joystick experience nor any motor or speech impairments, and all but two were native English speakers.

4.2.2 Study Apparatus

The equipment setup used was the same as that in the expert study presented in the previous section. The version of the *Dragon NaturallySpeaking* that was used for this study was version 8 Professional.

4.2.3 Study Design

For each of the three cursor control methods (SC, VJ, MG), the participant was exposed to a random sequence of 16 trials, comprised of two target sizes (26 pixels and 52 pixels) × 8 directions, and was asked to acquire the target as quickly and accurately as possible. The participants used the four-way version of the Vocal Joystick to accommodate the limited time available to train the required vowels. Due to the fact that the Vocal Joystick and Speech Cursor used for these conditions only allow movement along the four cardinal directions, distance to diagonal targets were measured using Manhattan distance and thus resulted in index of difficulty of 4 for non-diagonal targets and 4.7 for diagonal targets. As the emphasis was less on exhaustively covering all possible target conditions but rather on evaluating performance and preference under scenarios more closely resembling actual usage, I chose target sizes that roughly corresponds to the size of common interface elements such as buttons and links.

4.2.4 Study Procedure

Each participant was introduced to the three systems in counterbalanced order, and for each system, they first went through a training session to adapt the system to their voice and then were given five minutes to practice the particular cursor control method. After the training session, they were asked to perform a series of target acquisition tasks similar to those used in the expert user study but with much fewer conditions. Circular targets were used instead of the vertical ribbons used in the expert user study to make the task be more representative of typical pointing tasks in real user interfaces. At the end of each set of trials for a particular control method, the participants were asked to fill out a questionnaire and rate the method on a 7-point Likert scale based on the 10 categories shown in Figure 4-4.

4.2.5 Study Results

There was a significant difference ($p<0.05$) between the Speech Cursor and the other two modalities in mean target acquisition times, with Speech Cursor being roughly 4 times slower than the Vocal Joystick and the Mouse Grid (Figure 4-3). Surprisingly, there was no significant difference between the Vocal Joystick and the Mouse Grid. This was a promising result, since the Vocal Joystick was able to perform comparably to the faster of the two alternative speech-based pointer control methods, and was significantly faster ($p<0.05$) than the only one of the two alternatives that offered continuous pointer movement.

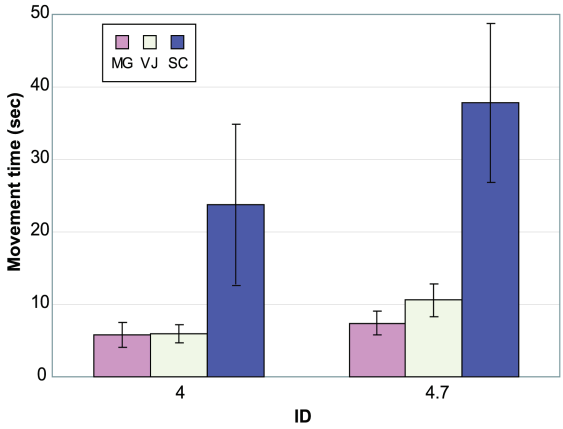


Figure 4-3: Mean movement times for the Mouse Grid (MG), the Vocal Joystick (VJ), and Speech Cursor (SC) across two different Fitts' indices of difficulty (*ID*) for novice users from the Vocal Joystick versus speech-based pointer control methods study. Error bars represent 95% confidence intervals.

Figure 4-4 shows the aggregated user ratings of the three control methods for each of the 10 categories. The ovals group points within each category whose difference was not significant ($p \geq 0.05$). Therefore, any two points within a category that do not lie within the same oval represent ratings that were significantly different ($p < 0.05$). The figure shows that the Mouse Grid was rated most favorably on most categories, but also that the Vocal Joystick ratings were not significantly different from it.

It is interesting to note that despite the Speech Cursor being rated higher than the Vocal Joystick in terms of how memorable the control method was, it was rated to be significantly more frustrating to use than the Vocal Joystick. Based on observation of the participants and the qualitative feedback I gathered during the post-session informal interview, participants indicated that the default Speech Cursor speed setting of 2 was too slow and desired a greater dynamic range on the speed change commands. Also, five of the nine users had substantial difficulty getting the system to recognize some of the direction words (about five unrecognized commands per trial). This may be due to the extremely short acoustic training period provided (three minutes) for the Mouse Grid and Speech Cursor. This result points to the strength of the Vocal Joystick, which was able to perform significantly better (Figure 4-3) given even shorter acoustic training time (eight seconds total).

This comparative evaluation revealed that novice users are indeed able to learn to use the Vocal Joystick given a short training period, and that they were able to achieve performance levels

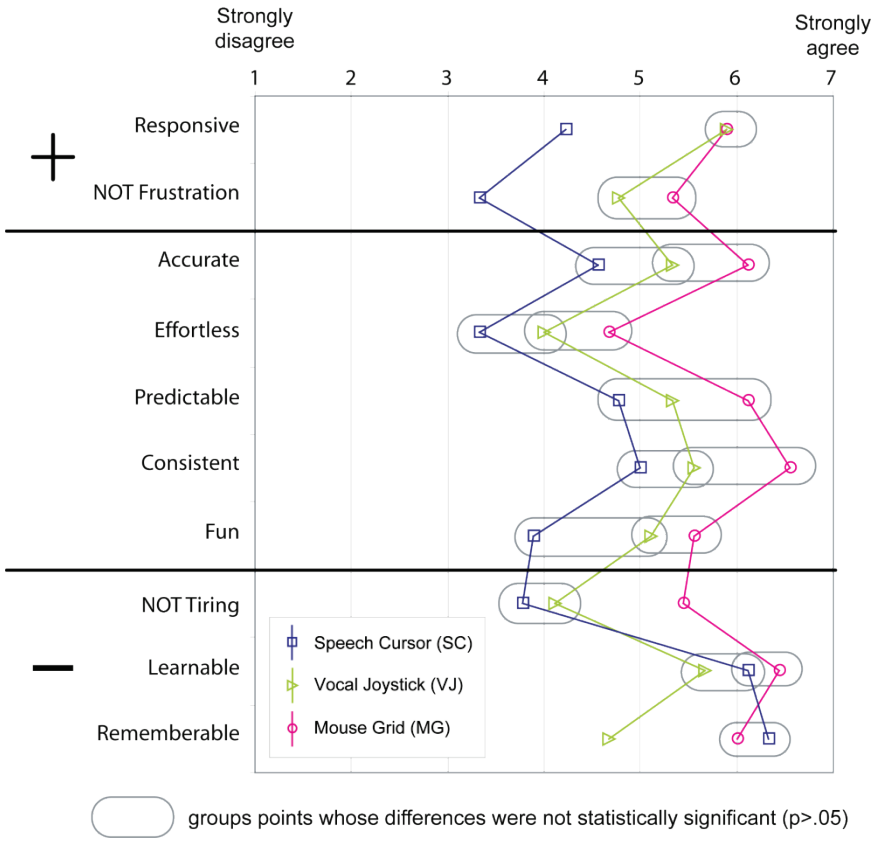


Figure 4-4: Novice users' subjective rating of the three cursor control methods: Mouse Grid (MG), Vocal Joystick (VJ) and Speech Cursor (SC). Items in the region labeled "+" are those in which VJ had a significantly better rating than another technique, and vice versa for the "-" region. VJ did not have any significantly different ratings ($p \geq 0.05$) for the items in the unlabeled region.

comparable to that of Mouse Grid. Mouse Grid is an attractive option from the point of view of its simplicity and reliability, however it cannot be used for non-discrete selection tasks such as path-following. For such tasks, functionality offered by Speech Cursor is necessary, and given the significant difference in both target acquisition times and tracing time, the Vocal Joystick seems to be a viable alternative to these existing speech-based cursor control methods.

4.3 Longitudinal Study of Learning to Use the Vocal Joystick

The user studies up to this point have shown that novice users can use the Vocal Joystick to control a pointer with minimal training and how the Vocal Joystick compares to other pointing devices. They have also shown that the expert Vocal Joystick performance follows Fitts' law and how that compares to the mouse. The two key questions that remain unanswered are, "what is the performance characteristic for the target population of people with motor impairments?" and, "what does the learning curve look like as a person gains more experience using the Vocal

Joystick?" To answer these questions, I conducted a 2.5 week longitudinal study with five motor impaired and four non-impaired participants spanning 10 one-hour sessions for each participant (Harada et al. 2009). In the following subsections, I present the stages of learning the Vocal Joystick that I have identified, followed by the details of the study setup and its results.

4.3.1 Stages of Learning the Vocal Joystick

The flexibility afforded by the Vocal Joystick is also accompanied by a set of unfamiliar controls that the user must learn. Based on my past observations, I identified the following four stages that a user might go through to acquire the skills needed to use the Vocal Joystick. The user study was designed to measure the progression of the participants through these stages as they acquired more experience with the Vocal Joystick.

Vowel production

First, the user needs to be able to produce each of the vowels being used (typically either four or eight vowels) distinctly and consistently. Due to the way in which the vowels were chosen, some of the sounds may not exist in certain languages or dialects. The system is flexible to a degree in being able to accommodate variations in individual pronunciations of the vowel sounds through the use of the adaptation process, so the key factor is that the user is able to produce four (or eight) distinct sounds in the proximity of the original vowel sounds rather than having to be able to produce the original sounds exactly.

Vowel direction mapping

Once the user is able to produce the vowel sounds, they need to memorize which vowel sound corresponds to which direction. As mentioned in Chapter 3, there is a carefully considered rationale behind the choice of the vowel sounds and their positioning relative to each other, but the actual decision of how these vowels should be oriented in the space of radial directions was arbitrary. Therefore, the user will need to become accustomed to this mapping through repeated exposure and memorization.

Volume control

Currently in the Vocal Joystick, the volume of the vocalization is mapped to the velocity of the corresponding pointer movement. At present, an exponential mapping is used between the power of the audio signal registered through the microphone and the resulting pointer velocity (Malkin

et al. 2005). Because most people have never experienced using the volume of their voice to directly manipulate a user interface, they will need to learn this mapping.

Smooth transitions

Being able to produce the right vowel sound and control its volume should allow the user to effectively move the pointer to a desired target. However, if the task demands that the user follow a curvilinear trajectory, it is necessary that the user be able to smoothly transition from one vowel to another and control the rate of transition in conjunction with their volume. Such a skill is essential for performing tasks such as creating drawings or playing games, which require continuously varying motion.

4.3.2 Study Participants

I recruited a total of ten participants for my longitudinal study (one had to drop out due to visa problems). Of the remaining nine, five had some form of motor impairment that affected the use of the hands in controlling a mouse (MI group), and the others had no motor impairments (NMI group). All were native English speakers. Table 4-2 shows the basic demographic information about each participant. Participants P01 through P05 form the MI group, and P06 through P10 form the NMI group. Additional descriptions of the MI group participants are presented next. Sears and Young (2003) provide a more in-depth coverage of some of the physical impairments in the context of computer technology.

P01 has had multiple sclerosis for 7 years, a progressive disease of the central nervous system that affects his ability to type or use the mouse for prolonged periods. He uses Dragon Naturally Speaking at work for dictating text, and mentioned that when he does try to type using the keyboard, his hands feel arthritic and he easily gets tired and sore. He stated that during exacerbations (i.e., sudden worsening of symptoms), his hands feel like they are on fire and he cannot grasp anything. He also noted that he frequently has to clear his throat, as multiple sclerosis can affect the larynx, but he was able to vocalize normally.

P02 has had idiopathic neuropathy since childhood, a disorder affecting the peripheral nerves leading him to experience pain and tingling in his hands when symptoms surface. He mentioned that his medication helps the pain from becoming too severe, and he is able to finger type on a keyboard and operate the mouse, although with increasing discomfort over time.

Table 4-2: Basic demographic information about the participants in the longitudinal study. The table on the left lists the participants with motor impairments (MI group), and the table on the right lists the participants without motor impairments (NMI group).

MI group

ID	Gender	Age	Impairment	Time with impairment	Effect on mouse usage
P01	M	52	Multiple sclerosis	7 years	Fatigue and pain
P02	M	51	Idiopathic neuropathy	Since childhood	Fatigue
P03	F	20	Muscular dystrophy	Since birth	Fatigue, difficulty moving
P04	F	30	Cerebral palsy (CP), Fibromyalgia (FM), Dyslexia	Since birth (CP), 13 years (FM)	Fatigue and spasm, hard to move and slow
P05	F	57	Parkinson's disease	16 years	Erratic movements and lack of reflex

NMI group

ID	Gender	Age
P06	F	30
P07	M	23
P09	M	19
P10	F	20

P03 has had muscular dystrophy since birth, a progressive muscle disorder that affects her range of mobility and manual control. She uses a powered wheelchair, and is able to move her arms but is unable to turn her palms face down or fully extend her elbows. She tried using speech recognition software a number of years ago, but stopped using it due to low accuracy. She can use the keyboard with both hands by using the backs of her fingers, but she mentions that she can only do so for several minutes before she gets very fatigued. She can also use the mouse by gripping it with the back of her two hands, but she finds it very hard to move and also tiring. She prefers to use a touchpad on her laptop computer, which she can operate by using the knuckle on her finger.

P04 has had multiple conditions that affect her motor abilities. She has had spastic cerebral palsy throughout her life, a non-progressive condition that affects her muscle strength and also causes her to spasm frequently when attempting to use her hands or arms. She has also had fibromyalgia for 13 years, a chronic condition characterized by widespread pain in the muscles. She also uses a powered wheelchair with an electronic ventilator that periodically pumps air out through a

breathing tube near her headrest. During the study sessions, she only needed the ventilator occasionally, and therefore was able to turn it off for most of the time, although I found that even when it was on, the Vocal Joystick was not affected. Generally, her speech ability is unimpaired, although she does occasionally experience shortness of breath. She also has dyslexia, which makes it hard for her to process written cues such as the words in the Vocal Joystick vowel compass. She is also able to use a keyboard and a mouse, although it is extremely tiring for her. She prefers instead to use a touchpad on her laptop computer.

P05 has had Parkinson's disease for 16 years, a degenerative nervous system disorder that has reduced her flexibility and reflexes, and now affects both sides of her body. She has mild resting tremors, and her voice tended to tremble and be monotonic, a common symptom of the condition. She can use the mouse, but complains that she often ends up clicking the buttons unintentionally, and has a hard time with continuous motion.

4.3.3 Study Apparatus

The study sessions were conducted using a Dell Optiplex GX280 desktop computer running Windows Vista Business with an Intel Pentium 4 processor clocked at 3.4GHz with 1.5GB of RAM. The computer was connected to a Dell 2001FP 20" monitor displaying 1280×960 pixels at a resolution of 96 dpi. A Plantronics DSP400 USB headset microphone was used for sound input. 8-way mode was used for the Vocal Joystick.

4.3.4 Study Procedure

The study took place in a lab over a period of 10 sessions for each participant. Each session was one hour long, except for the first and last sessions being 90 minutes long due to system introduction and the final comparative assessment using each participant's preferred pointing device. For each participant, the time between consecutive sessions was no less than 3 hours and no more than 48 hours. The participants were compensated for their time by being paid $25 for the first session, $10 for each of the subsequent eight sessions, and $145 for the 10^{th} session.

During the first session, participants were introduced to the Vocal Joystick control method and to the vowel compass, and were shown how to use the vowel feedback tool (Figure 4-5a). In the vowel feedback tool, the user can click on the speaker icon below each word to see and hear a video of the corresponding sound being pronounced. The user can also vocalize and see the system's recognition result, indicated by the yellow arrow. For the user study, after the review of

vowel sounds, the participant's vowel utterances were collected using the Vocal Joystick application to build the initial user profile.

The general structure of each session was as follows. At the beginning of the session, the participant was tested on their vowel recall to see if they remembered the mappings of the sounds to directions. Next, they used the vowel feedback tool (Figure 4-5a) to review the vowel sounds and practice tuning their vocalization. They were given the option of readapting the Vocal Joystick user profile if they felt that the system was not responding well.

Following the vowel testing, the participants engaged in two stages of tasks: a target acquisition stage and a steering stage. During the target acquisition stage, the participants first practiced by navigating through a series of web sites using the Vocal Joystick. During this phase, they were free to adjust the sensitivity of the Vocal Joystick or readapt the user profile if desired. After the practice phase, they engaged in a Fitts' law reciprocal target acquisition task as shown in Figure 4-5b. The factors and levels used in the study were the same as those used in the expert study presented in the previous section.

During the steering stage, the participants first practiced by playing a simple game whose objective was to keep the player's fish away from larger fish on the screen, and they also used a drawing program to attempt to trace a set of figure eight paths. After the practice phase, they engaged in a steering task (Accot & Zhai 1997) in which they attempted to move the pointer

(a) Vowel feedback tool (b) Target acquisition task test screen (c) Steering task test screen

Figure 4-5: The three stages that the participants went through during each study session in the longitudinal Vocal Joystick study. (a) The vowel feedback tool. The speaker icon below each word can be clicked to both hear and see the video of the corresponding sound. The participant can also vocalize themselves and see the system's recognition result, indicated by the yellow arrow. (b) The horizontal targets in the Fitts' law reciprocal target acquisition task (blue bar is the next target). (c) The circular tunnel in the steering task condition with the entry and exit target shown as the vertical bar and the trail of the pointer movement shown in the tunnel.

through the circular tunnel of varying sizes and widths as quickly as possible without going out of bounds (Figure 4-5c). The factors and levels for the steering task were as follows:

- Tunnel radius (R): {100, 200 pixels}
- Tunnel width (W): {100, 140 pixels}

The combination of the tunnel radii and widths yielded four indices of difficulty (ID) of 4.49, 6.28, 8.98, and 12.57 bits. For each of the $R \times W$ conditions, the participants were presented with 8 trials, where half of those trials where clockwise and the other half were counter-clockwise. I aggregated the rotation direction in my analysis.

4.3.5 Study Results

I was able to collect a significant amount of data from the 99 hours with nine participants. I highlight some of the key results from the longitudinal user study below, associating each stage of the study session with the stages of learning presented earlier.

Vowel recall (vowel production and direction mapping)

All participants were able to memorize the vowel-to-direction mapping during the 10-session period. On average, the participants were able to correctly recall all eight vowels and conduct the entire session without the aid of the vowel compass after the 5th session.

The accuracy of vowel *production*, on the other hand, varied widely among participants, such that although they could recall what the sound should be for a particular direction, they had difficulty vocalizing it in such a way that the Vocal Joystick consistently recognized it as the intended sound. A great number of participants had difficulty consistently and distinctly vocalizing the sounds [a], [ɑ] and [ɨ]. Depending on the region of origin, the native pronunciations of the representative words provided for these sounds in Figure 3-4 do not contain the intended sounds, and the user is unable to distinguish them from the adjacent sounds. This is one of the difficult tradeoffs that the Vocal Joystick system has to make. To provide the expressiveness and the ability to smoothly move at arbitrary angles, the system needs the user to be able to produce as many distinct vowel sounds as possible. However, not everyone may be able to produce or even perceive as many distinct sounds, and significant differences may be present among individuals depending on their origin and dialect. Although the system adapts to the user's sounds, it operates best when the sounds are close to the originally trained sounds, which were chosen to be

maximally distinct in acoustic space. To attempt to deal with the sounds that were giving them trouble, participants came up with a variety of alternate representations to help them make the correct sound, such as "all" and "'ol" for [ɑ], and "good" and "err" for [ɨ].

Target acquisition (volume control)

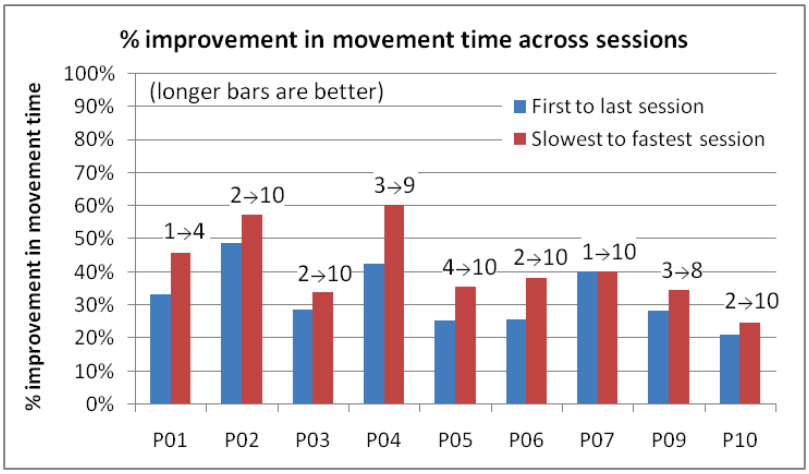
Figure 4-6a shows the percent improvement in average movement time for the target acquisition task between the first and last sessions, as well as between the sessions with the slowest and fastest average movement times. Overall, participants demonstrated improvement in their performance over the 10-session period of at least 20%. Among the NMI group, average improvement in movement time ranged from 25% to 49% (34% to 60% if comparing slowest to fastest sessions), and 21% to 40% (24% to 40% for slowest to fastest) among the MI group.

Figure 4-6b shows the final Fitts' throughput achieved by each participant after the last session using the Vocal Joystick, with higher throughput indicating better performance. The section for P03 and P04 also contains the throughput that each of these participants achieved using a mouse and their preferred device, a touchpad. For the rest of the participants, their preferred device (mouse) throughput was measured but is omitted from the figure due to scale. The MI group's mouse throughput was in the range of 3.9 to 4.6 bits/sec. For the NMI group, that range was 4.9 to 6.2 bits/sec. Note that by the final session, P03 and P04's Vocal Joystick throughput had equaled or exceeded their mouse throughput, and reached 75% and 61% of their preferred touchpad throughput, respectively. This is an impressive result for these two MI participants.

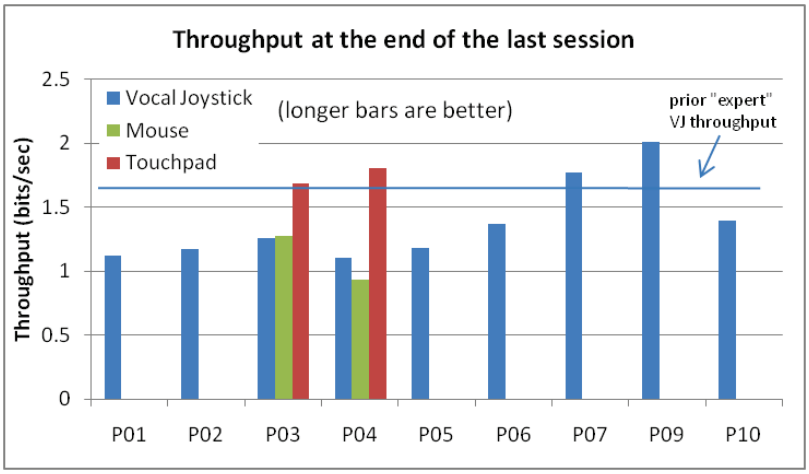
Two of the NMI participants exceeded the previously observed expert Vocal Joystick throughput of 1.65 bits/sec as shown in Figure 4-6b. The average throughput for the NMI group was 1.64 bits/sec, which is comparable to the prior expert VJ throughput. The average throughput for the MI group was slightly lower at 1.17 bits/sec, or 70% of the NMI group. There may be a number of reasons for this difference, which needs to be investigated in more detail, including the difference in age and amount of experience with computers.

If I project the same rate of learning as exhibited during the study into the future by fitting a power curve to the per-session data,[1] then P03 and P04 are projected to attain the previously set "expert" Vocal Joystick throughput level after 36 and 17 more sessions, respectively. If they

[1] The fitted power curve of learning was $TP = 0.94s^{0.15}$ with $R^2 = 0.60$ for P03, and $TP = 0.66s^{0.28}$ with $R^2 = 0.61$ for P04, where *TP* is the throughput in bits per second and *s* is the session number.



(a)



(b)

Figure 4-6: Results from the longitudinal study, with P01 through P05 representing participants from the MI group and P06 through P10 the NMI group. (a) The percent improvement in average movement time between the first and the last session (left bar) and between the slowest and the fastest session (right bar) for each participant. Numbers above the right bars indicate the session numbers corresponding to the slowest and fastest sessions. (b) The Fitts' throughput measure for the Vocal Joystick at the end of the 10th session. For P03 and P04 whose native pointing device is the touchpad, their touchpad and mouse throughputs are also shown. All other participants' mouse throughput ranged from 3.9 to 6.2 (mean of 5.1).

attempt to attain the same throughput as their touchpad, P03 will only need another 44 sessions and P04 another 28 sessions beyond the initial 10 sessions. This shows that the Vocal Joystick has the potential of offering comparable performance to current devices without the need for physical manipulation or an unreasonably long practice period. Figure 4-7 shows the trend of the average Vocal Joystick throughput for each group. It is not clear why there was a slight downward trend around the third quarter of the sessions.

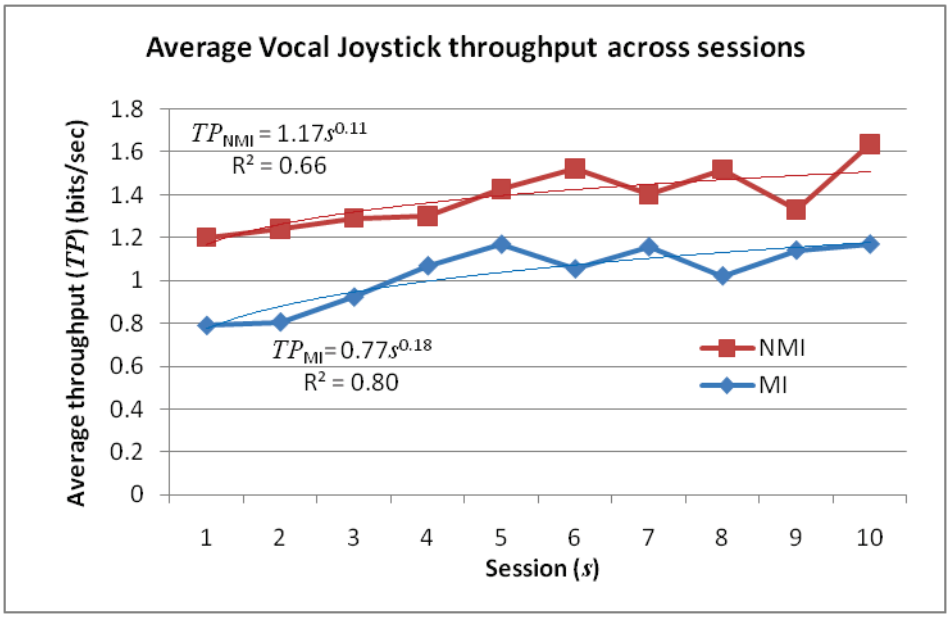


Figure 4-7: The average Vocal Joystick throughput over each session for the participants with (MI) and without (NMI) motor impairments. The fitted power curves of learning are shown for each participant group, in which *TP* is the throughput in bits per second and *s* is the session number.

One of the most salient issues with the current system that surfaced as a result of this study was the need for a better mapping between the volume of the user's vocalization and the speed of the mouse pointer movement. Although this issue has been investigated before (Malkin et al. 2005), a more thorough study needs to be conducted to examine this relationship between vocalization volume and speed with a greater number of users and drawing from the design of similar devices such as isometric joysticks. From the user interface standpoint, this also suggests that an intuitive interface is needed to allow the user to adjust the speed mapping to tune it to their preference.

4.4 Comparative Evaluation of Categorical Voice Input with Speech Input

The studies up to this point have focused on the performance characteristics of voice input for fluid and continuous control of the mouse pointer. While the need for the ability to provide such fluid input via voice has already been stressed, it is also important that categorical inputs, or control signals that are discrete in nature, can also be specified quickly and rapidly. While speech input, especially when it comes to dictation, has the potential to outperform the keyboard, the delay involved in uttering and recognizing a short command can be significantly longer for a general speech recognizer than the time it takes to press a key combination.

As mentioned in Chapter 3, it is possible to process and reinterpret the vowel probability output from the pattern extractor module to generate categorical input signals. For example, the vowel probabilities can be compared against some preset threshold value to yield a binary output of whether or not a vowel is considered to have been recognized. The resulting discrete recognition event can then be used to trigger a corresponding discrete action, such as a key press. The potential advantage of using such a mechanism for generating categorical input is that compared to speech-based input, the processing time required to recognize a vowel sound utterance should be noticeably faster. Also, the ability for the Vocal Joystick engine to recognize nine distinct vowel-sounds means that they can be mapped to nine categorical inputs in addition to the two already provided through the discrete sounds. Finally, the directional arrangement of the vowel sounds makes them—especially the four cardinal vowel sounds—ideal for quickly generating corresponding arrow-key events in various applications that accept arrow-key entries. Keyboard-operated computer games are a prime example of such an application domain in which rapid and timely execution of multiple discrete input actions is required.

To better understand how such a use of non-speech vocal input for generating discrete signals compares to keyboard and speech input, I conducted a simple reaction time (RT) experiment involving directional visual stimuli and responses by one of three input modalities: key (four arrow keys on a keyboard), voice (Vocal Joystick's directional vowel mapping), and speech (words "up," "down," "left," and "right"). The specific objectives and the setup of the experiment can be better understood in the context of the Model Human Processor (Card et al. 1983), which I describe below.

4.4.1 Experiment Background

The Model Human Processor is an abstraction of the human cognitive-perceptual-motor system in the context of performing some basic computer tasks, originally formulated by Card et al. (1983) in the context of human-computer interaction. Under the model, a user interacting with a computer system is depicted as being composed of three interrelated subsystems—the perceptual, cognitive, and motor systems—each with their corresponding "processors" and "memories." Figure 4-8 shows a simplified depiction of the model and the general flow of control from one processor to another. The figure also includes a depiction of an "input processor"—a representation of any deciphering that the computer may need to perform on the input signal from the user—which is not included in the original model since it is not part of the user. This

processor may not account for much in trivial cases such as processing a button press, but if the input is a spoken command, the input processor needs significantly more time to recognize what was uttered before it can be made available as input to any client applications.

A typical processing flow through the Model Human Processor during a simple response task is as follows. Upon the onset of a stimulus (i.e., when a stimulus is generated by the computer system), the user's perceptual processor activates to process the relevant sensory input. If the task at hand is not a trivial automatic-response task (i.e., simple task that the user has learned to react without thinking), the cognitive processor may then make several accesses to the short-term working memory and possibly the long-term memory as well to decipher the stimulus and

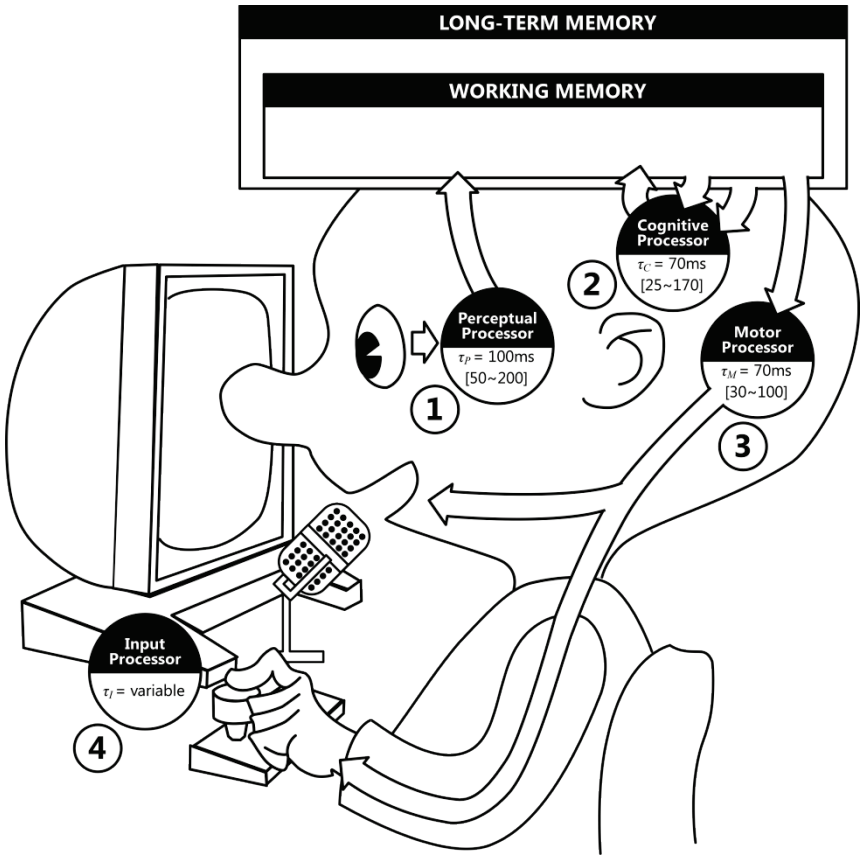


Figure 4-8: A simplified representation of the Model Human Processor adapted from (Card et al. 1983) and modified to include the *input processor*. When a user is presented with a stimulus, the *perceptual processor* (1) processes it and stores its impression in working memory. The *cognitive processor* (2) then accesses the working and long-term memory, possibly multiple times, to determine the appropriate response. Next, the *motor processor* (3) executes the determined response through the appropriate motor system. Finally, the *input processor* on the computer system (4) interprets the input signal received from the user and takes appropriate action. Each processor is annotated with the average *cycle time* required for one unit of processing, along with their typical value ranges in brackets.

determine the appropriate response. Finally, the motor processor converts the response action into motor signals and transmits them to the relevant motor systems such as the hand or the mouth. The Model Human Processor also provides empirically-determined "cycle times" for each processor, corresponding to the average amount of time it takes a particular processor to process one unit of input. The perceptual processor cycle time (τ_P) has been determined to be on the order of 100ms, with typical values ranging from 50 to 200ms. The cognitive processor cycle time (τ_C) averages 70ms with a range of 25 to 170ms, and the motor processor cycle time (τ_M) also requiring about 70ms with a range of 30 to 100ms. The input processor cycle time (τ_I) is completely dependent on the specific type of input modality and recognition algorithm used. Another useful measure aside from cycle times is the total time spent in each processor. Henceforth, this measure will be indicated as $T_{<processor>,<modality>,<task>}$, where <*processor*> is any one of the four processors introduced above, <*modality*> is the input modality involved (one of *key* (*k*), *speech* (*s*), or *voice* (*v*) in the case of my experiment), and <*task*> is the type of reaction time task involved, which are described next.

There are two types of reaction time tasks that are commonly used to study human task performance: *simple reaction* (*si*) tasks and *choice reaction* (*ch*) tasks. The difference in the response times between these tasks manifests itself in the cognitive processor time. In a simple reaction task, there is only one type of stimulus and response, and the user is instructed to issue the response as soon as they perceive the stimulus. In a choice reaction task, the user is presented one of a number of possible stimuli, each with a corresponding unique response that the use must correctly execute. In the case of a simple reaction task, $T_{C,*,si}$ (the asterisk in the subscript indicating that the value of the corresponding slot does not matter) is equal to τ_C. This reflects the fact that it only requires one cycle of the cognitive processor to connect the representation of the stimulus in the working memory to a corresponding response action. In the case of a choice reaction task, $T_{C,*,ch}$ is calculated as:

$$T_{C,ch} = I_C \log_2(n+1) \tag{6}$$

where I_C is an empirically determined constant and n is the total number of stimulus types. Card et al. (1983) provides a value of 150 ms/bit as an estimate for I_C independent of modality. There are also indications that vocal response times in general tend to be slower than manual response times by roughly 30% to 60% (Baron & Journey 1989; Nebes 1978; Starkweather et al. 1984), but it is not clear how much of the difference is attributed to the cognitive versus the motor

processor. Since the main focus is not on the actual times spent in each processor but rather on the total reaction time, I simply assume that $T_{M,<modality>,*}$ is dependent on modality.

Figure 4-9 summarizes the various temporal measures that are of interest in reaction time tests. *Reaction time* ($RT_{<modality>,<task>}$) is defined as the time between the onset of the stimulus and the onset of the response. For my experiment, since the stimulus is assumed to be visual, $T_{P,*,*}$ can be considered to be a constant with a value of τ_P. The *response onset* corresponds to the earliest point at which the muscle group responsible for executing the physical response action starts to fire (e.g., for a button-press response, when the finger starts to move, and for a voiced response, when the vocal cord starts to vibrate). *Execution time* ($ET_{<modality>}$) represents the time it takes to complete the physical response action (e.g., the time between the start of the movement of the finger and the completion of the button press). *Processing time* ($PT_{<modality>}$) is the time it takes for the system to process the response signal from an input device and generate a corresponding software input event, referred to as the *input onset*. For keyboard input, this is a negligible value, but for speech input using a general recognizer, the value can be significant. The sum of the elapsed time from stimulus onset to input onset is referred to as the input time ($IT_{<modality>,<task>}$).

An approximate quantification of the values presented above for different modalities and task types can be derived by referencing the average processor times included in the Model Human Processor as well as results from other studies. Table 4-3 summarizes such approximations for

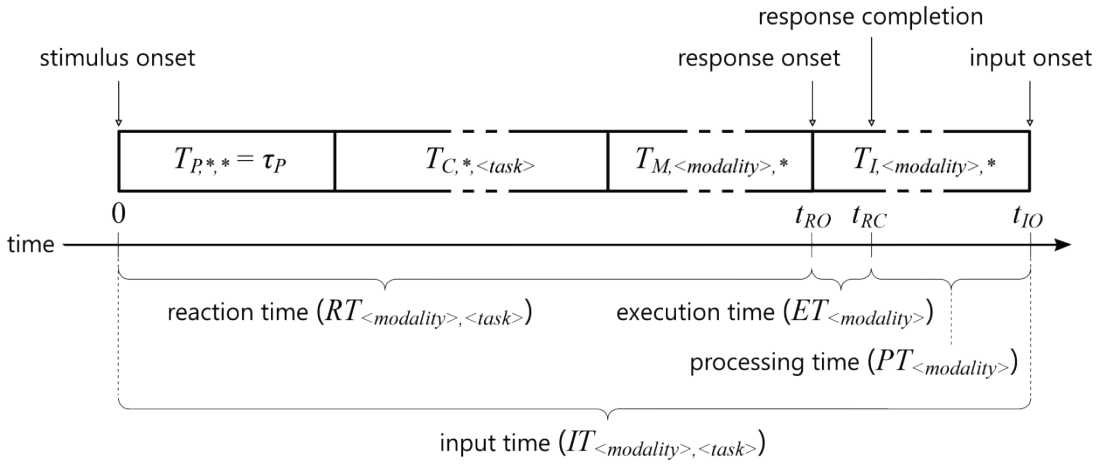


Figure 4-9: Summary of the relevant measures in a reaction time task. The notation $T_{<processor>,<modality>,<task>}$ represents the time taken in the corresponding processor of the Model Human Processor given a particular input modality (which could be one of *key*, *speech*, or *voice* in the context of my experiment) and reaction task type *task* (which could either be *simple* or *choice*). An asterisk in the subscript indicates that the value of the corresponding slot does not matter.

Table 4-3: Summary of the approximate quantifications of various reaction time measures for the key and speech modality under the simple and choice reaction tasks using visual stimuli, based on data from other experiments. Data for $RT_{v,*}$ are missing since prior studies measuring non-speech vocal reaction times were conducted using auditory instead of visual stimuli, which is known to yield different results.

Measure	Approximate value	Derivation
$RT_{k,si}$	273 [105~470] ms	$T_{P,*,*} + T_{C,*,si} + T_{M,k,*}$ $= \tau_P + \tau_C + \tau_M$ $= 100\ [50\sim200] + 70[25\sim170] + 70[30\sim100]$ $= 240\ [105\sim470]$ (Card et al. 1983) 260 [200~370] ms (Nebes 1978) 320 [270~390] ms (Venables & O'connor 1959)
$RT_{k,ch}$	518 [428~648] ms	$T_{P,*,*} + T_{C,*,ch} + T_{M,k,*}$ $= \tau_P + I_C \log_2(n+1) + \tau_M$ $= 100\ [50\sim200] + 150 \times \log_2(4+1) + 70\ [30\sim100]$ $= 518\ [428\sim648]$ (Card et al. 1983)
$RT_{s,si}$	369 [250~480] ms	334 [250~420] (Nebes 1978) 362 [???~???] (Baron & Journey 1989) 410 [360~480] (Venables & O'connor 1959)
$RT_{s,ch}$	722 [???~???] ms	722 [???~???] (Baron & Journey 1989)

$RT_{k,si}$, $RT_{k,ch}$, $RT_{s,si}$, and $RT_{s,ch}$. There is not enough data to approximate the values for $RT_{v,si}$ and $RT_{v,ch}$ since most of the prior work investigating reaction times of non-speech vocal input have used auditory stimuli (Izdebski 1980; Izdebski & Shipp 1978; Shipp et al. 1984; Starkweather et al. 1984; Till et al. 1981) which have been shown to yield significantly faster reaction time results compared to visual stimuli (Teichner 1954; Venables & O'connor 1959). These values will also be used to verify the corresponding measures in my experiment.

Based on the model and representation presented above, the following hypotheses can be formulated about the differences between keyboard, speech, and non-speech voice modalities with regards to their performance in reaction time tasks (summarized visually in Figure 4-10).

- *Hypothesis #1*: For trained users, the difference in reaction times across modalities is expected to be relatively small since $T_{P,*,*}$ and $T_{C,*,ch}$ should be constant across modalities. In particular, $RT_{v,*}$ and $RT_{s,*}$ should be nearly identical given that the same motor system is involved and thus the same amount of time should be spent in the motor processor. $RT_{k,*}$ may be slightly faster than $RT_{v,*}$ and $RT_{s,*}$ given that the vocal cord activation may take longer to generate an audible sound compared to the movement of a finger.

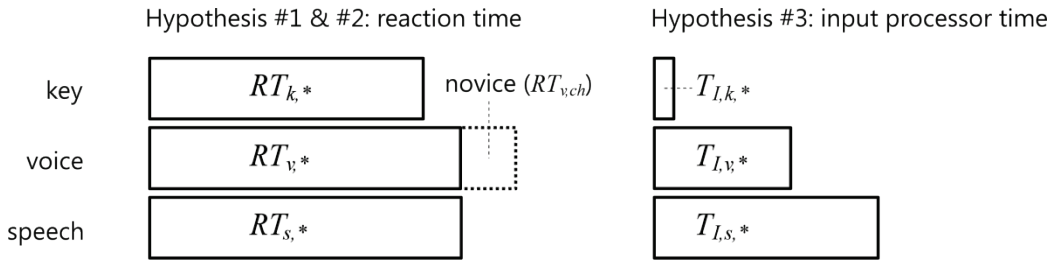


Figure 4-10: Summary of the hypotheses of the experiment. Hypothesis #1 states that reaction times across modalities should be relatively similar, especially between voice and speech for trained users. Hypothesis #2 states that for a novice user, the voice modality may incur additional time to recall the unfamiliar sound-to-direction mapping. Hypothesis #3 states that the input processor time should be almost zero for the key modality, and that it should be significantly less for the voice modality compared to the speech modality.

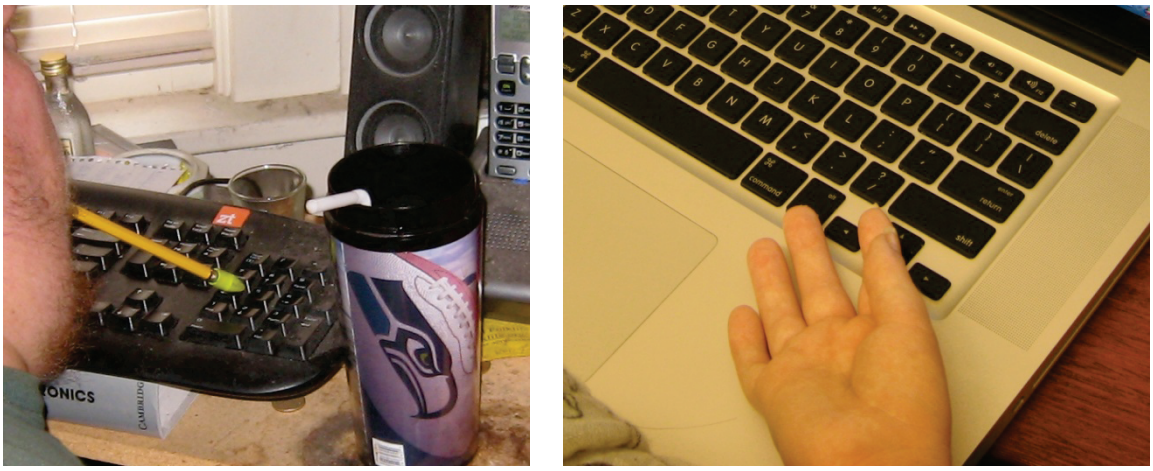
- *Hypothesis #2*: For a novice user under the choice reaction task using the voice modality, the unfamiliarity with the sound-to-direction mapping would most likely lead to a longer $T_{C,v,ch}$ and thus longer $RT_{v,ch}$ compared to $RT_{k,ch}$ or $RT_{s,ch}$.
- *Hypothesis #3*: For either type of tasks, $T_{I,k,*}$ is expected to be significantly faster than $T_{I,s,*}$, and $T_{I,v,*}$ to lie somewhere in between, since $T_{I,k,*}$ should be close to zero and non-speech vocalization should take less time to utter and be recognized compared to a speech command given its relative simplicity.

4.4.2 Experiment Participants

Eight participants ranging in age from 21 to 34 were recruited to take part in the experiment. Two of the participants had motor impairments that affected the use of their hands for manipulating a keyboard and mouse (MI), and the rest did not (NMI). All eight participants had participated in prior Vocal Joystick user studies and had used the Vocal Joystick for an average of four hours each and learned the directional vowel mappings. One of the MI participants (P1) who had arthrogryposis multiplex congenita was unable to use the hands to manipulate a mouse or keyboard but was able to use a mouth stick to press the keys on a keyboard (Figure 4-11a). The other MI participant (P2) had muscular dystrophy and was able to use the back of her fingers to press the keyboard keys.

4.4.3 Experiment Setup

Each participant took part in a series of reaction time (RT) paradigm trials in which they were presented with a visual stimulus in the form of a blue arrow (100×100 pixels) appearing on a computer screen (15” LCD monitor with resolution of 1400×900 pixels) pointing in one of the



(a) (b)

Figure 4-11: Two of the MI participants who participated in the reaction time study. (a) P1 had arthrogryposis multiplex congenita and was able to use a mouth stick to press the keys on a keyboard. (b) P2 had muscular dystrophy and was able to use the back of her fingers to press the keys.

four cardinal directions, and asked to respond to the stimulus in a manner dependent on a particular condition. The two independent factors and their levels were:

- Modality: {key, voice, speech}
- Task type : {simple, choice}

The *key* modality involved pressing one of the four arrow keys on a full-sized USB desktop keyboard. The *voice* modality involved the utterance of one of the four cardinal vowel sounds in the Vocal Joystick vowel compass (Figure 3-4), processed by the Vocal Joystick engine. Finally, the *speech* modality involved the utterance of one of the four direction words ("up," "down," "left," or "right"), processed by the Windows Speech Recognizer. The Windows Speech Recognizer was set up such that the general dictation grammar was disabled, and only the simple grammar consisting of the four directional words was active, thereby minimizing the perplexity of the language model and optimizing its performance for this limited vocabulary set. For both the voice and speech modalities, each user underwent a basic adaptation process to train the system specifically for their voice. Under the voice modality, the user vocalized the four vowel sounds for two seconds each while the Vocal Joystick engine recorded them and updated its acoustic model. Under the speech modality, the user read several paragraphs of passage as part of the Windows Speech Recognizer voice training wizard process .

Under the *simple* task type, the stimulus was always an up arrow, and the participant was instructed to respond using the same response (up key for the *key* modality, word "up" for the *speech* modality, and the upward vowel sound for the *voice* modality) as soon as they saw the stimulus. Under this mode, the wait time between the end of one trial and the appearance of the next stimulus was randomized between 1 and 2 seconds to prevent the participants from anticipating their presentation. This range of pre-stimulus interval was found in past studies to yield the fastest reaction times (Izdebski 1980; Shipp et al. 1984). Under the *choice* task type, the participant was asked to to generate a response corresponding to the indicated direction of the arrow stimulus as quickly and accurately as possible. In this mode, the delay between the participant's response and the presentation of the next stimuli was fixed at 1 second.

For all trials, both the reaction time ($RT_{<modality>,<task>}$) and input time ($IT_{<modality>,<task>}$) were measured. In the *key* modality case, the response onset and input onset were both considered to occur simultaneously when the key-down event was generated. In the case of both the *voice* and *speech* modality, the response onset was detected when a voiced audio frame was first detected by the Vocal Joystick engine after the stimulus onset. Under the *voice* modality, the response completion event corresponded to the moment when the Vocal Joystick engine's single-vowel detector observed a silent audio frame after a minimum of 40ms of sustained vowel sound. At that point, a vowel-detected event was raised, causing a corresponding arrow-key's key-down event to be emulated, marking the input onset time. Under the *speech* modality, the response completion event corresponded to the moment when the user finished uttering the direction word, and the input onset was marked by the Windows Speech Recognizer raising the speech-recognized event and causing the corresponding arrow-key's key-down event to be emulated.

4.4.4 Experiment Results

The results from the NMI group are presented first, followed by those from individual MI participants, as the latter results highlight some interesting differences due to the way their use of the keyboard is affected.

The aggregate results from the NMI group are summarized in Figure 4-12. The comparison of $RT_{<modality>,si}$ across modalities show that under the simple reaction task, the voice and speech reaction times were comparable, while the key reaction time was significantly faster than either voice or speech. This confirms hypothesis #1. While the reaction time for the key modality was

well within the expected average range derived in Table 4-3, the speech reaction time was at the high end of the expected range. This could be due to the way in which I measured the onset of the speech vocalization using voicing detection through the microphone, as opposed to a more elaborate mechanism used in some of the previous studies such as laryngographs.

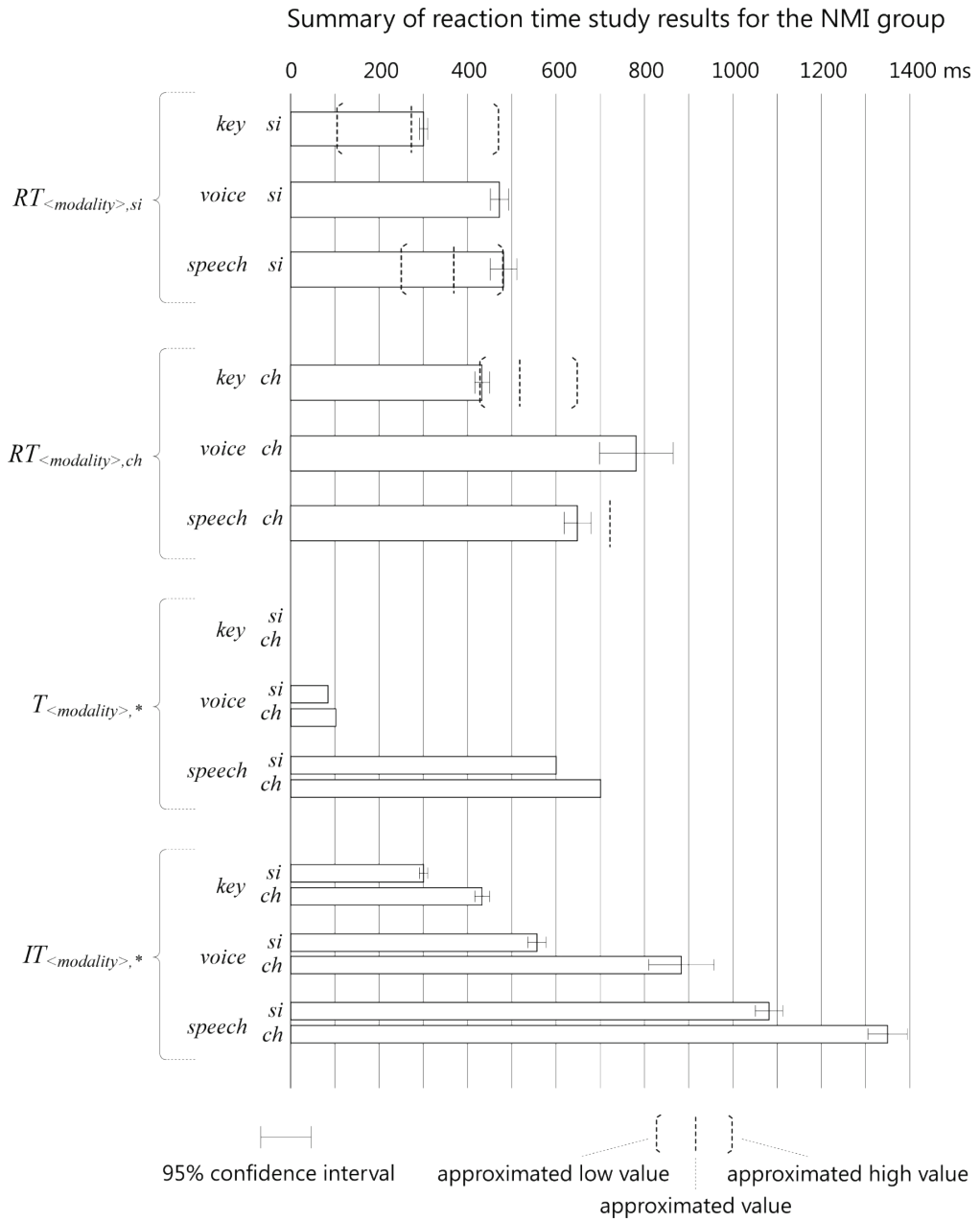


Figure 4-12: Summary of the results from the reaction time study for the NMI group. Error bars represent 95% confidence intervals, and dashed lines represent the approximate values derived in Table 4-3.

The comparison of $RT_{<modality>,ch}$ across modalities show that under the choice reaction task, for the voice modality was slower than the speech (782 milliseconds versus 649 milliseconds). This could be seen as supporting hypothesis #2, since even though the participants have had four hours of exposure to the Vocal Joystick, their association of the vowel sounds with their corresponding directions may not yet be automatic. The fact that this difference between voice and speech manifested itself in the choice reaction task but not in the simple reaction task indicate that the slowdown is most likely due to the extra cognitive processing required. Therefore, theoretically speaking, it is reasonable to expect that once the user gains true proficiency in the vowel-to-direction mapping, that their reaction time under the choice reaction task will approach that of the speech modality (or possibly become slightly faster, as suggested by the results from the simple reaction task).

The side-by-side comparison of $T_{<modality>,*}$ between task types within each modality verifies that the input processor time did not depend on the task type, as expected. There is a slight difference in the input processor time for the speech modality between the simple and choice reaction tasks (600 milliseconds versus 701 milliseconds, respectively), but this may be due to the fact that the input processor time includes the response completion time, which on average would be longer under the choice task since words such as "right," "left," and "down" take longer to utter than "up" which was used for the simple reaction task. Also, the results reveal that $T_{s,*}$ was significantly longer than $T_{v,*}$ (on average, 651 milliseconds versus 93 milliseconds, respectively), supporting hypothesis #3. The fact that $T_{v,*}$ was almost negligible is a promising validation of the ability for the non-speech vocal input to be processed extremely quickly.

Finally, the comparison of the total input time $IT_{<modality>,*}$ reveal that overall, the key modality was the fastest (as expected), at 301 milliseconds for the simple reaction task and 433 milliseconds for the choice reaction task. However, the voice modality was significantly faster than speech in both task types (557 milliseconds versus 1,082 milliseconds for the simple reaction task, and 883 milliseconds versus 1,350 milliseconds for the choice reaction task), even with the longer reaction time for the voice modality under the choice reaction task. As mentioned above, if it is assumed that with practice $RT_{v,ch}$ could approach $RT_{s,ch}$, then $IT_{v,ch}$ can be expected to decrease even further by up to 133 milliseconds ($RT_{v,ch} - RT_{s,ch}$) to 750 milliseconds, making it almost 45% less than $IT_{s,ch}$ (1,350 milliseconds).

The results from participant P1 of the MI group are summarized in Figure 4-13. As expected, the key modality was significantly slower compared to the NMI group, but interestingly the voice and speech modality were also slower across the board relative to the NMI group. This may be due to the impact of the participant's motor impairment on the respiratory capacity and reflexes, even though the participants had unimpaired speech. Another interesting point is that under the choice reaction task, the reaction times across all three modalities were comparable. The fact that the key modality was almost as slow as the voice and speech modalities is most likely due to the difficulty of coordinating the movement of the mouth stick quickly in response to the stimulus. This is a promising sign for the comparative advantage of the voice modality over the key

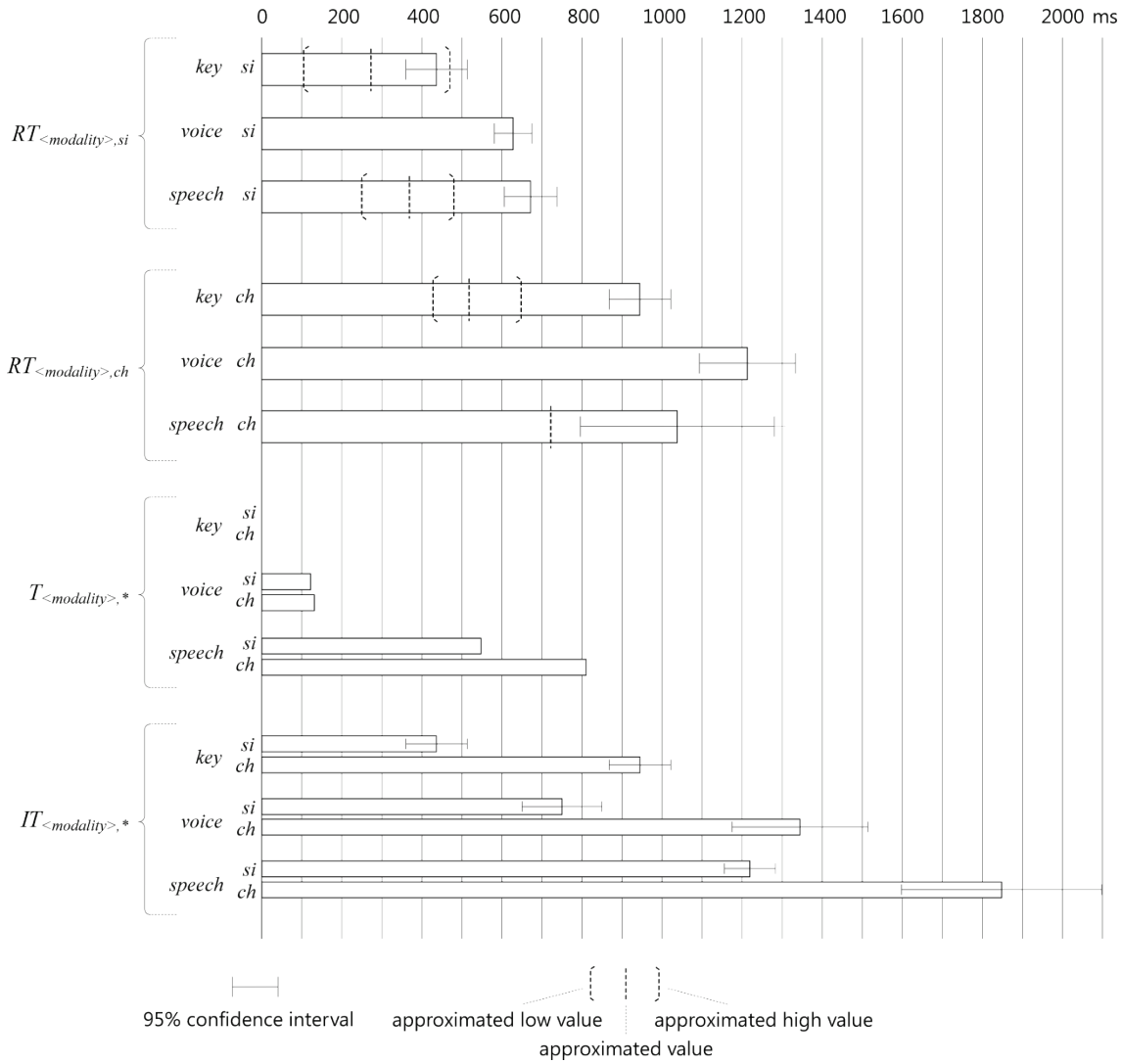


Figure 4-13: Summary of the results from the reaction time study for the MI participant P1.

modality for this participant. The $IT_{<modality>,*}$ comparison indicates that for the choice reaction task, the difference between the key and voice modality is much smaller than in the NMI group's case. Applying the same adjustment as for the NMI group to account for the potential decrease in $RT_{v,ch}$, $IT_{v,ch}$ (originally 1,345 milliseconds) could reduce to 1,169 milliseconds, making it comparable to $IT_{k,ch}$ (945 milliseconds) and almost 40% faster than $IT_{s,ch}$ (1,848 milliseconds).

Finally, the results from participant P2 of the MI group are summarized in Figure 4-14. Similar to P1, the key modality was significantly slower than the NMI group average, and the difference

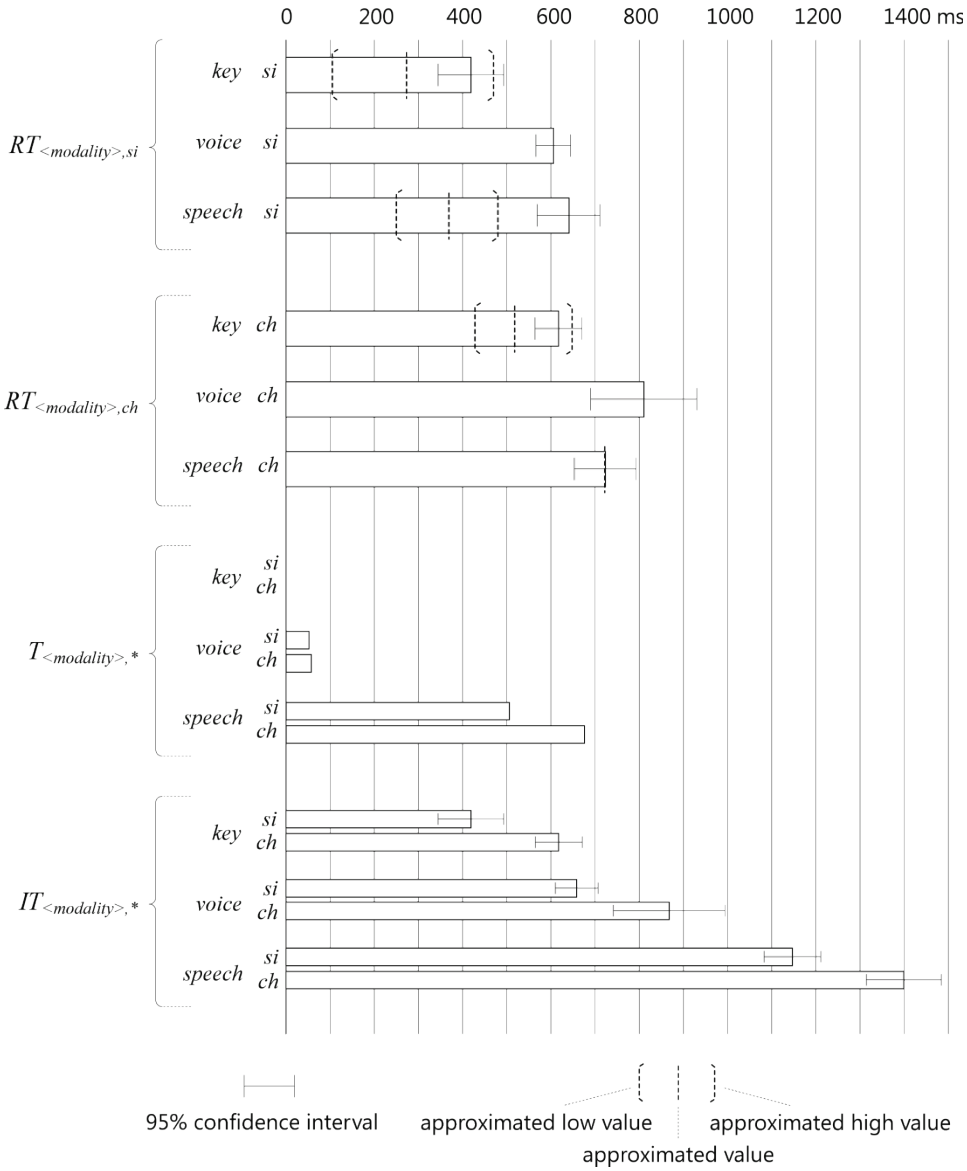


Figure 4-14: Summary of the results from the reaction time study for the MI participant P2.

between $RT_{k,ch}$ and $RT_{v,ch}$ was also smaller. Applying the same adjustment to $RT_{v,ch}$ as the previous groups, $IT_{v,ch}$ could reduce to 780 milliseconds, making it only 25% slower than $IT_{k,ch}$ (618 milliseconds) and almost 45% faster than $IT_{s,ch}$ (1,399 milliseconds).

4.5 Summary

In this chapter, I described four representative user studies that I conducted to investigate the performance characteristics of voice input for fundamental interaction tasks. The first study involving expert Vocal Joystick users established an initial benchmark for pointing task performance that could be expected once a person gains significant experience using the Vocal Joystick. The study also provided one of the first pieces of evidence for the fact that voice-driven target acquisition using the Vocal Joystick follows Fitts' law ($MT = 606.15 \times ID + 411.63$ milliseconds, $R^2 = 0.986$). The relative performance of the Vocal Joystick was roughly 30% compared to the conventional mouse, and 70% compared to rate-controlled joysticks. The second study with novice users showed that the Vocal Joystick can yield significantly better performance than the Speech Cursor method even with minimal training, and comparable performance to the Mouse Grid method while offering the ability to steer the pointer, which is not possible with the Mouse Grid. The third study observing both users with and without motor impairments learn the Vocal Joystick over multiple weeks showed that in less than 10 hours of use, the users without motor impairments were able to reach or surpass the expert level measured in the first study, and the users with motor impairments reached 70% of the expert level performance with trends showing continued improvement. Finally, the fourth study investigated the use of non-speech vocalization for discrete categorical input and revealed that non-speech vocalization can offer a faster method for specifying categorical input compared to speech-based input.

The findings described above indicate that there is potential in non-speech vocal input as a modality for enabling fluid hands-free control of computer interfaces. To situate these findings in a more realistic setting, I have a number of concrete applications that leverage the functionality offered by the Vocal Joystick engine. In the next three chapters, I present three representative applications that I have built: VoiceDraw, VoiceGames, VoicePen, and Voice Controller.

Chapter 5
VoiceDraw*

The user studies presented in the previous chapter led to a better understanding of the basic performance characteristics of non-speech vocalization as application control-signals. To better gain insight on how such a novel modality may be applicable in a practical context, I subsequently investigated its use in various concrete application domains. Through such exploration, my goal was to uncover common interaction patterns that can be extracted into a generalizable toolbox for design of other voice-driven applications. The next three chapters present the details of those applications.

In this chapter, I present an application called VoiceDraw (Figure 5-1), a drawing application built specifically to leverage the capability of the Vocal Joystick engine and speech-based input to enable completely hands-free creation of artistic drawings. The following sections provide a description of the features of the VoiceDraw application, my design process, including user-centered design sessions with a self-described "voice painter," and present lessons learned that could inform future voice-based design efforts.

* Parts of this chapter are adapted from (Harada et al. 2007b).

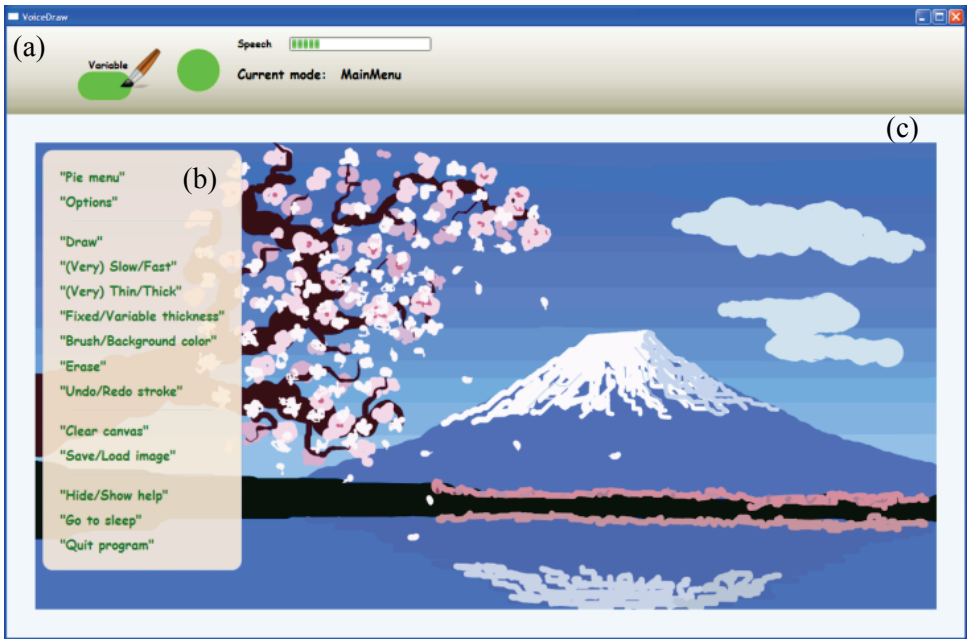


Figure 5-1: A screenshot of the VoiceDraw application showing (a) the status bar, (b) help overlay, and (c) canvas area. The author created this painting using only his voice in about 2.5 hours.

5.1 Motivation

Creative self-expression and artistic endeavors can play a vital role in enhancing people's quality of life, including those with various types of disabilities (Warren 1997). Despite the challenges that motor impairments can pose on an individual's ability to manipulate physical art mediums such as paint brushes or drawing pencils, numerous people have overcome such disabilities through creative re-adaptation of existing tools. For example, the Association of Mouth and Foot Painting Artists of the World (AMFPA)[1] represents artists with various disabilities affecting the use of their hands who create artwork using their mouths or feet.

However, for those with moderate to severe motor impairments, manipulation of physical tools may be difficult or impossible. Even those with some ability to manipulate physical artistic media may find the process arduous enough to be a barrier to engaging in creative activity. Computer applications hold promise for enabling such individuals with limited motor abilities to engage in creative activities with reduced overhead of manipulating physical tools. Painting programs on a computer can simulate physical brush strokes or even provide artistic effects not possible in the physical domain.

[1] http://www.amfpa.com/

Given the hands-free nature of voice-driven input, the fluid control required for creating freeform drawings on a computer, and the ability for the Vocal Joystick engine to provide such fluid input, the potential benefit of creating a drawing program driven by the Vocal Joystick engine was clear. To investigate this possibility, I developed VoiceDraw (Harada et al. 2007b), a fluid drawing application that allows users with motor impairments to create artwork by using non-speech properties of their voice.

This chapter presents the design, implementation, and evaluation of VoiceDraw. My process included interviews, field investigations, and user-centered design sessions over a period of two weeks with a self-described "electronic voice painter" who formerly used Dragon Dictate and Microsoft Paint to produce his own artwork. Based on my sessions with this artist, I made numerous iterative design refinements. Ultimately, the contributions of this work include the VoiceDraw system, new voice-based interaction techniques (e.g., Voice Marking Menu and continuous undo), and some lessons learned, which can be used to inform future designs for voice-based user interfaces.

The projects most similar to VoiceDraw are art installations and exhibits. Levin et al. (2004) present interactive art installations in which the sounds generated by the participants are captured by the system and rendered as an artistic projection onto a publicly visible surface. Although the system controls the appearance of the image based on the phonetic features of the incoming audio, it does not provide users with a level of control comparable to traditional painting.

An exhibit called the VoicePainter[2] in New York allows children to make vocalizations into a microphone while the system populates a digital canvas with random pre-stored patterns in response to sound. Based on the descriptions in the press release, it does not appear that the features of the voice are correlated with the patterns selected.

Beyond voice, EyeDraw (Hornof & Cavender 2005) enables children with severe cerebral palsy to draw using their eyes and an eye tracker. The latest version supports the ability to add straight lines, rectangles, ellipses, and predefined icons, but no free-form strokes as I do in VoiceDraw. The creators of EyeDraw note the limitations of eye-based interaction in supporting "free eye" drawing due to the inability of the eye to move in slow continuous movements, and the overloading of drawing and viewing with movements of the eye (i.e., the Midas Touch problem).

[2] http://www.voicepainter.com/index.php

Another program called the EyeWriter developed through the EyeWriter Initiative[3] offers a similar set of functionality, using dwelling to select tools and to specify vertices of a polygonal shape in a connect-the-dots fashion. The program is more focused on creating drawings of letters in a fashion similar to graffiti tags, for which it was developed. The mouse movements are smoothed so that there is a bit of a lag between the gaze position and the pointer position. It does not support drawing of free-form, non-linear paths.

In exploring the combination of a head tracker and speech recognizer for supporting hands-free pointer control, Malkewitz (Malkewitz 1998) compares a simple circular figure painted with his system to that made by the mouse. However, the paper does not provide any further discussion regarding the design issues raised.

5.2 Interview with a "Voice Painter"

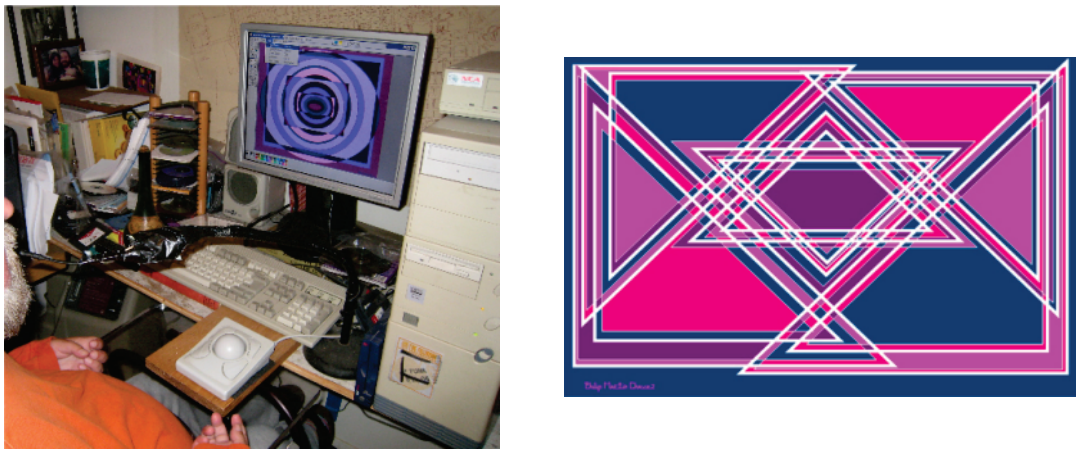
Despite the absence of tools that enable drawing on a computer using voice, I weas able to find an artist who refers to himself as an "electronic voice painter." To gain an understanding of the current challenges he faces with his existing tools, I conducted a series of user-centered design sessions, starting with a situated semi-structured interview.

5.2.1 Background

When I interviewed him in 2006, Philip Chavez[4] had been creating art on his computer using his voice for over 15 years. He has had a spinal cord injury at the C4-C5 level for about 30 years, and has limited movement in his shoulder. As a result, he has no dexterity in his elbow or wrist, and no sensation in his hands. His speech is unimpaired, but he does have limited lung capacity, which restricts his ability to produce long uninterrupted vocalizations.

His inspiration to create art on his computer came after a period of successive tragic events, leading him to seek an outlet for his emotions. He experimented with Microsoft Paint, a basic painting program that ships with Windows 95 and later Microsoft operating systems. With Paint, he used Dragon Dictate speech recognition software to create his first "voice art" piece. Since then, he has produced hundreds of pieces of art, many of which are viewable on his website. Many others have been on display at various exhibits.

[3] http://www.eyewriter.org/
[4] The artist has specifically requested that we use his real name and that we credit his artwork.



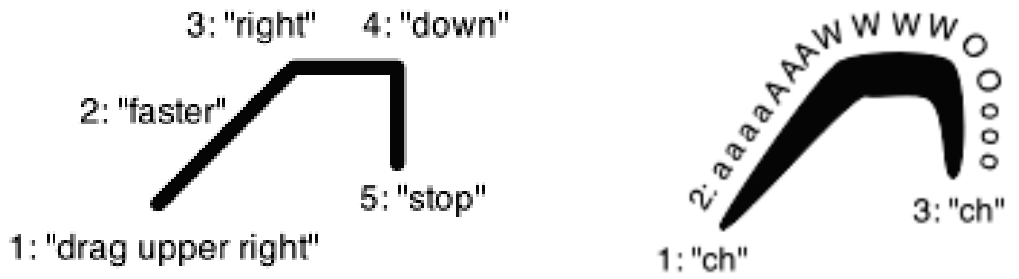
(a) (b)

Figure 5-2: (a) The voice painter's computer setup, and (b) one of the art pieces produced by the voice painter using his current tools prior to being introduced to VoiceDraw.

Much of Mr. Chavez's work is abstract, composed mainly of straight lines, geometric shapes, and solid color fills. This is due in part to the constraints of his current tools, but he also attributes much of his style to Jackson Pollock and the abstract expressionists, as well as his Navajo and Mescalero Apache heritage. He is also a strong believer in R. Buckminster Fuller's concept of "ephemeralization," of doing more with less (Fuller 2000). Mr. Chavez told me, "there are many things that can be accomplished faster if you use very basic tools … When your options are too great, it can inhibit the creative process." This is reflected in the experience he had when he once tried using Adobe Photoshop and found it too frustrating due to the large number of features, particularly since many of them were not easily accessible using speech commands.

5.2.2 His Current Tools

The current computer setup used by the voice painter is shown in Figure 5-2a. The primary mode of his interaction with the computer is through speech commands using Dragon Dictate Classic (version 3) and a microphone on a stand. He is also able to use a trackball by controlling his arm from the shoulder and using the back of his hand to move the trackball and to click on buttons. Despite the lack of tactile sensation in his hands, the voice painter has gained enough precision to click on targets as small as a standard scroll bar arrow. However, he still prefers to use speech as his primary modality, and only relies on the trackball when speech fails. A drawback of using the trackball is that it causes neck and shoulder strain, so he prefers speech.



(a) (b)

Figure 5-3: Sample interaction of drawing a stroke using (a) speech-based cursor control, and (b) VoiceDraw. With speech-based control, changes are discrete and incremental. With VoiceDraw, changes are continuous and fluid. Volume can be mapped to stroke thickness or brush speed.

The voice painter has tried a number of mouse alternatives, including an eye tracker, head tracker, and a mouth-operated joystick, but has settled on speech for a number of reasons. He found the eye tracker to be difficult to control with the level of precision he desired, and found the process of turning off the tracking mode to be too cumbersome as he frequently needs to recline his powered wheelchair to stretch out his back. With the head tracker, he found the neck and shoulder strain to be too great for prolonged use. He was also unable to get his desired level of control with the mouth-operated joystick, and did not like the fact that it could get quite messy.

5.2.3 Current Drawing Process

For the voice painter, the first step in interacting with his computer typically begins with the activation of his speech recognizer by issuing the verbal command "wake up." Up to this point, the speech recognizer was in "sleep mode," where all verbal commands and extraneous sounds were ignored except for the "wake up" command. (If the speech recognizer was not already running, it can be manually invoked using the trackball.) The voice painter then uses a sequence of verbal commands to open a Microsoft Paint file that has been pre-configured to the appropriate dimensions. He then begins the drawing process.

Once Microsoft Paint is launched, the voice painter selects a tool from the tool palette or a color from the color palette by using the MouseGrid feature (introduced in Section 2.1.3). "Strokes" are made using discrete speech commands. The voice painter begins moving the mouse cursor in one of the eight 45° angles using a speech command such as "drag upper right." The pointer begins moving in the specified direction at the specified speed, both of which can be altered on the fly by uttering commands such as "much faster" and "move left." The artist also occasionally moves his

trackball while the pointer is in motion to intentionally introduce randomness into the stroke. Issuing the "stop" verbal command terminates pointer movement. Figure 5-3a illustrates how the voice painter makes a stroke using speech commands. To change the color of the brush, he must use the MouseGrid or trackball to select the desired color from the color palette. He also often uses the paint bucket tool to fill an area with a solid color. A sample piece produced by the voice painter using this current process is shown in Figure 5-2b. The time spent on each piece ranges from a few hours to 40 hours, with some taking up to 100 hours.

5.2.4 Limitations of the Current Setup

The voice painter pointed out a number of limitations that he encounters when using his current tools. One problem is when the speech recognizer fails, especially for mid-stroke corrections. When the voice painter is in the process of making strokes that consist of numerous direction changes similar to those shown in Figure 5-2b, if the speech recognizer fails to recognize a command at the exact right time, the pointer continues moving beyond the desired point. To recover from such errors, the entire stroke needs to be undone, which can be very costly if the error was near the end of a long or complex stroke. This is a general problem with so called "passive modes" and the lack of continuous control using speech commands. Once a command establishes a mode (e.g., brush movement), another command must be successfully issued to stop or amend it. The use of the eraser tool is also plagued by this problem, as an area larger than desired may be erased accidentally, requiring the whole erasure to be undone.

Another limitation raised by the voice painter pertains to the expressiveness afforded by the current tool. He stated his frustration, saying, "I can't get real brush strokes, and can't get the texture I'm thinking about." He also pointed out that he would often have a vivid image in his mind that he wished to express; the image often included curves and smooth shapes, but with the current tools, he was only able to achieve a crude representation using ovals or by manipulating individual pixels. This motivated me to prioritize supporting continuous curves as an essential feature in VoiceDraw's repertoire.

When I asked the voice painter whether he feels attached to his tools given their role in shaping his artistic style, he stated:

> *"I think I'm out-growing this [Microsoft Paint] and am ready to explore new options. My artwork and progression is limited by the tools I have. I'm excited about the possibility of expanding that."*

5.3 Design Goals

Based on the interviews with the voice painter, I focused on the following design goals for my initial implementation. I acknowledge that they are based on my interaction with only one artist, and that they will need to be supplemented by feedback from additional users in the future. However, I believe that the insights and design goals derived from an initial deep exploration can provide a solid starting point for iterating towards a more general design in the future.

Continuous and fluid input

A number of issues raised during the interview can be traced to the fact that the current method and tools lack necessary direct manipulation and fluidity. For example, the voice painter cites the ability to express realistic brush attributes such as variable thicknesses and smooth curves as being key to expressing some of his artistic visions. Also, his use of the trackball for introducing randomness during pointer movement reflects his need to have direct, immediate control over his output. To support such realism and expressiveness, the new tool must be able to:

- provide fluid direct manipulation of the brush, and
- provide the ability to simultaneously manipulate brush characteristics during motion.

Sustaining the flow of creative process

The voice painter stressed the importance of not interrupting the flow of creativity:

> *"When I'm in the zone, the cumbersome maneuver to get the [stroke] angle I want hinders the creative process."*

He also expressed the following:

> *"If I can verbally change the angles without stopping, it probably wouldn't interrupt the creative process so much, and I'll be able to create much, much better pieces."*

In light of these sentiments, I strove to enable fluid control over brush angles and characteristics (e.g., thickness) without the need for accessing menus, toolbars, or issuing discrete commands, all of which can break the creative process.

Reducing strain

One important consideration in reducing the artist's burden is to keep training time to a minimum. The voice painter mentioned that he has had to retrain his speech recognizer numerous times due

to the computer crashing or his adapted profile not working well. On the other hand, the voice painter appreciates the reduced physical strain of speech-based interaction compared to using his trackball or a head tracker. He said:

> *"Doing it by voice really allows me to work much longer."*

Flexible support for reversing actions

His inability to undo and redo just a small portion of his strokes in Microsoft Paint has been a big problem for the voice painter. This is particularly problematic when the most recent stroke was the result of a long or complex process.

> *"Having to redo an entire stroke after a mistake at the very end is extremely painful... I also save at each stage so that I can figure out how I did the piece later on."*

Voice Draw was designed to support all of these goals, as described in the next section.

5.4 VoiceDraw System

I now present an overview of the VoiceDraw application followed by the details of its features. I describe how the design of each feature was influenced by the feedback obtained in my user-centered design process. I divide the features into two categories: those that are specific to stroke creation itself, and those that are more general features of the VoiceDraw application.

5.4.1 Application Overview

A screenshot of the VoiceDraw application is shown in Figure 5-4. The screen area consists of a status bar at the top of the screen, a canvas area below it, and a translucent help overlay showing the valid speech commands within the current context. Although non-speech vocalization can be used to operate VoiceDraw, speech commands are also provided as shortcuts for users that may want them.

In contrast to the sequence of discrete speech commands that must be issued using Dragon Dictate, a typical VoiceDraw stroke proceeds as follows. The brush head is moved to the desired canvas location by making a continuous vowel utterance based on the 2-D sound-map. The discrete sound "ch" sets the brush down. The user then vocalizes a vowel sequence corresponding to the desired path while varying the vocalization volume to get the desired variation in stroke thickness. The same discrete sound "ch" lifts the brush off the canvas (Figure 5-3b).

5.4.2 Stroke Creation

I decided not to overload the application with extraneous functions, but to focus on highly controllable stroke creation that is currently not possible using speech under existing systems.

Brush speed

VoiceDraw provides four preset brush speed settings (very slow, slow, fast, very fast). Once a particular brush speed is chosen, the brush moves at that speed for as long as the user continues to vocalize. The current brush speed is reflected in the status bar as the length of the stroke in the brush preview (Figure 5-4).

In the first iteration of the prototype, the brush speed was mapped directly to the vocalization volume, allowing the user to dynamically vary the brush speed while the stroke thickness remained constant. However, based on the findings from my field investigations and feedback from the voice painter, the default mapping was changed to the current setup, where speed is based on the brush itself (e.g., slow brush or fast brush) and volume controls stroke thickness. This gave the voice painter a greater sense of artistic control.

Stroke thickness

The current version of VoiceDraw maps the volume of the vocalization directly to the thickness of the stroke (see Figure 5-3b). The user can change their volume mid-stroke, which will result in a stroke with varying thickness. There are four preset brush thickness settings (very thin, thin, thick, and very thick) that define the maximum thickness achievable when the volume is at its highest. There is also an ability to set the brush thickness to a fixed mode so that the entire stroke

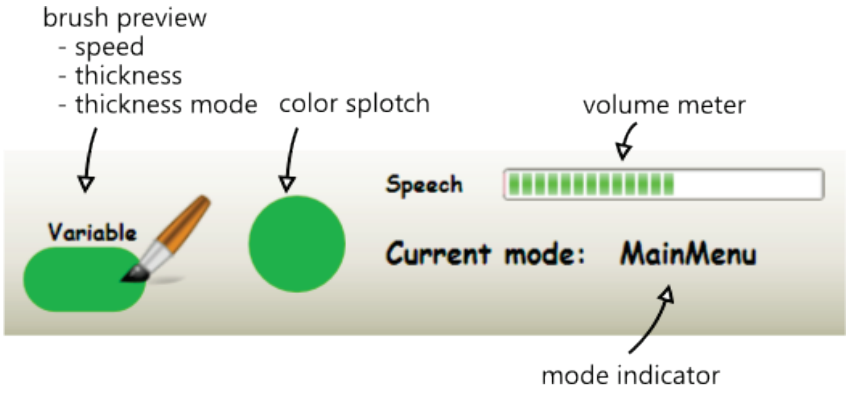


Figure 5-4: Status bar of the VoiceDraw application showing the brush preview, current color splotch, volume meter, and the mode indicator.

will be drawn with the same preset thickness, in which case the volume can be used to control the brush speed. The currently selected brush thickness is reflected in the status bar and the thickness of the stroke in the brush preview (Figure 5-4).

The initial version utilized the pitch of the vocalization to control thickness, but I discovered that users' change in pitch resulted in a change in the recognized vowel, even when users were sustaining the same vowel. The voice painter also mentioned that he is not good at controlling his pitch, so I instead mapped volume to stroke thickness. At the same time, a fixed-thickness option was also added to address the difficulty the voice painter experienced in sustaining a constant volume for an extended period of time due to his limited lung capacity.

In the ideal case, one could simultaneously use volume and pitch to vary two continuous parameters of the brush, but more work needs to be done on the underlying Vocal Joystick engine before this becomes feasible across all users. Even *if* such work is done, however, my findings here indicate that users may not be comfortable controlling multiple brush parameters in this way. Instead, it may be more effective to streamline the brush selection process and use different brushes with different characteristics (e.g., fast brush or slow brush) along with one intuitive mapping, such as between volume and stroke thickness.

5.4.3 Application Features

Verbal commands

Because the voice painter was more familiar with issuing commands through speech, I kept the ability to access all features using direct speech commands. The user can issue a number of commands to directly manipulate brush properties, such as "(very) slow/fast," "(very) thin/thick," "fixed/variable thickness," "draw," as well as changing to other states such as the color picker and erase mode.

To show which commands are valid in each mode, I implemented a floating help overlay with a list of commands that are always valid in the current mode, as well as non-speech vocalizations that are valid (Figure 5-1b). The voice painter found this to be particularly helpful in the beginning as I experimented with various sets of commands and wordings. To ensure that no part of the canvas is ever hidden behind the help overlay, the overlay automatically repositions itself to the side opposite the brush.

(a) (b)

Figure 5-5: The Voice Marking Menu supports menu navigation using only non-speech vocalizations. The menu is invoked by issuing the discrete sound "ck". (a) The user finishes uttering "aww" (right), and is about to open the submenu by issuing the discrete sound "ch"; (b) the user finishes uttering "eee" (left) within the submenu and is about to execute the command by issuing the discrete sound "ch".

Voice Marking Menu

In response to the feedback about not breaking one's creative flow, I implemented a *Voice Marking Menu* (Figure 5-5) that allows the user to invoke menu commands without leaving VoiceDraw's non-speech vocalization modality. This design was modeled after marking menus (Tapia & Kurtenbach 1995).

When the Voice Marking Menu is invoked by the discrete sound "ck", it presents the top-level menu items as pie wedges. An arrow next to their label indicates menu items that have submenus. The user can highlight a pie wedge containing the desired menu item by uttering the vowel sound corresponding to the direction of the menu item from the center of the menu. The menu item can be selected by making the discrete sound "ck", which executes a command or opens a submenu. The menu can be canceled by uttering the middle vowel-sound [ə].

Continuous, incremental undo

VoiceDraw supports a novel undo feature in addition to the standard whole stroke undo found in Microsoft Paint. The standard per stroke undo works just like normal undo (executed by issuing an "undo stroke" speech command) and removes the most recently added stroke from the canvas. In contrast, the continuous undo feature (Figure 5-6) erases incrementally from the end of the stroke while the user continuously utters the "up" vowel sound ([æ]). It is also possible to reverse the process by uttering the "down" vowel sound ([u]).

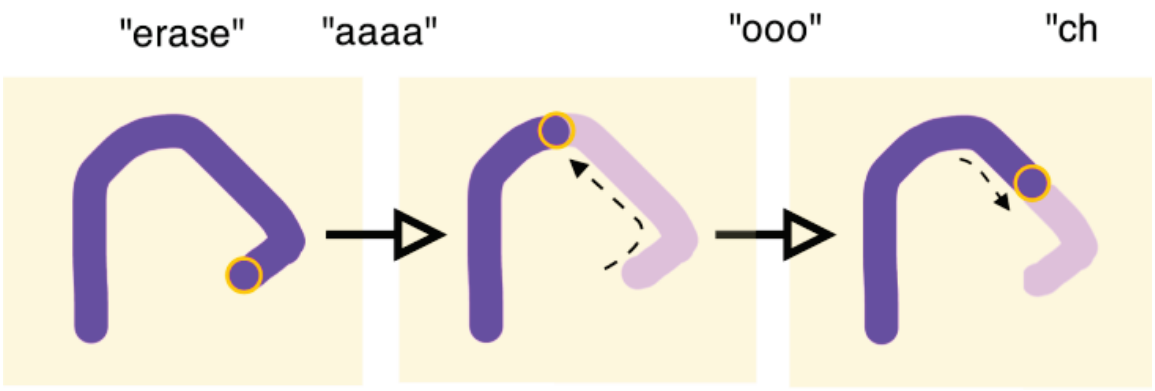


Figure 5-6: Continuous undo and redo are initiated by issuing the "erase" command, followed by continuous utterance of the "up" vowel ("aaa" as in "cat") to incrementally erase from the end of the stroke, and continuous utterance of the "down" vowel ("ooo" as in "boot") to incrementally restore the portion of the stroke. The change is committed with the discrete sound "ch".

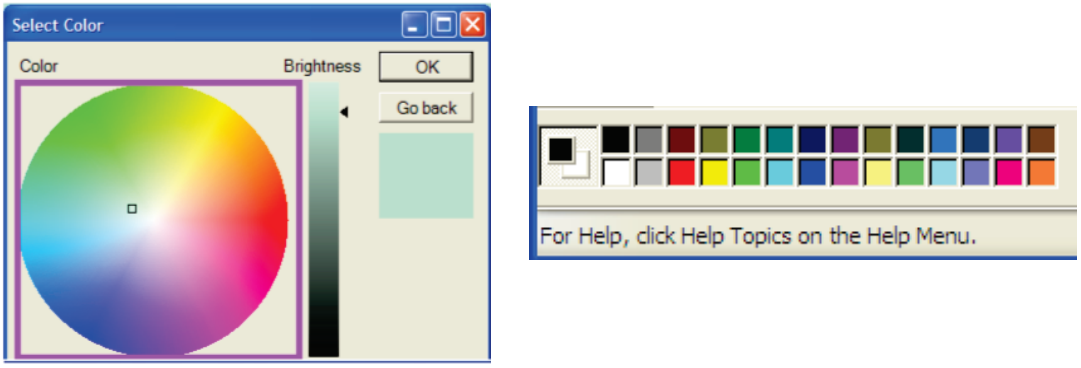
Color picker

In the color dialog (Figure 5-7), I let the user specify the color and brightness separately using non-speech vocalizations. The circular color wheel can be navigated using the same 2-D vowel mapping that controls the brush, and the vertical brightness is adjusted by using just the "up" and "down" vowels. I could have used individual sliders for each of the RGB or HSV values, but that would have required the user to explicitly switch among each slider. The color wheel/brightness representation was preferred by the voice painter over other representations. The color wheel also allowed the voice painter to quickly locate a desired color instead of being restricted to a limited palette, and without having to move the cursor into a tiny palette square.

5.4.4 Implementation

The VoiceDraw application is written in C#, leveraging the Windows Presentation Foundation (WPF) and Extensible Application Markup Language (XAML) from the .NET Framework 3.0. The .NET version of the Vocal Joystick engine library presented in Section 3.3 was used to facilitate integration with the front-end application.

The speech command recognition functionality is integrated into the VoiceDraw application through the use of Microsoft Speech API version 5.3. For my sessions with the voice painter, a desktop computer running Windows XP was used. An untrained user profile was used throughout the process for the speech recognizer, as it provided sufficient accuracy given the limited size of



(a) (b)

Figure 5-7: (a) The VoiceDraw color wheel (currently active) and the brightness slider, used with continuous 2-D vowel sounds. (b) Microsoft Paint's color palette with small color squares that the user must click with the pointer—a difficult task for anyone, but especially for users who cannot use a pointing device.

VoiceDraw's command grammar. The total training time required for the Vocal Joystick engine is only 18 seconds, at 2 seconds per vowel.

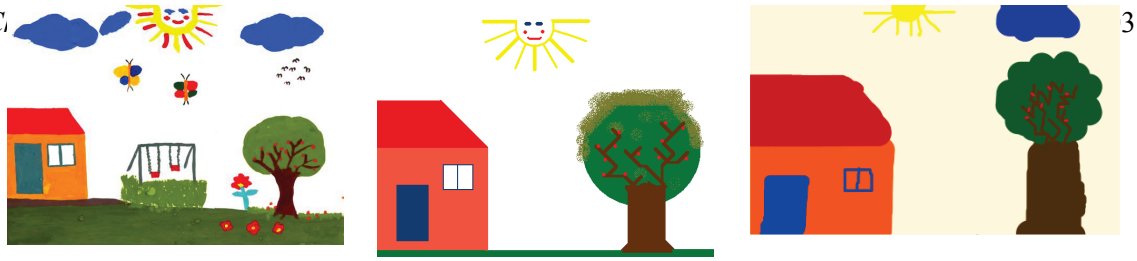
Due to the fact that I implemented the brush strokes as a collection of scalable vector graphics within the WPF framework, there were a number of benefits and shortcomings that I had to consider. The benefit of choosing this representation is that the resulting artwork is resolution-independent, and that operations such as the continuous undo and redo become possible. The downside to this approach is that it is not easy to replicate some of the features of bitmap-based painting programs such as Microsoft Paint, in particular the "flood fill" operation. I provide a compromise by enabling the production of freeform strokes with variable properties such as thickness, and providing extra thick brushes to efficiently fill in large areas.

5.5 Outcomes

This section summarizes the qualitative assessment that was conducted to determine the effectiveness of VoiceDraw, and the lessons learned from the process.

5.5.1 Tasks

To assess the usability of VoiceDraw, I asked the voice painter to engage in two tasks. The first was a directed painting task in which I presented him with a target painting (Figure 5-8a) and asked him to create similar ones using his current tools and VoiceDraw. The second task was an



(a) (b) (c)

Figure 5-8: Results from the directed painting task, with (a) the target drawing taken from a school art class, (b) the replica using Microsoft Paint and Dragon Dictate, and (c) the replica using VoiceDraw. The replicas took approximately 45 minutes and 1 hour to create, respectively. Note that the voice painter had relatively little experience with VoiceDraw at this point.

undirected painting task where the voice painter created whatever he liked using VoiceDraw, while I observed.

The target drawing for the directed painting task[5] was chosen to explore how each tool can support the portrayal of a scene that consists of both rectilinear and non-rectilinear elements. The voice painter was told not to worry about replicating every detail exactly, but to portray as much as he felt was necessary for a good representation of the original scene.

The results of both tasks are shown in Figure 5-8 and Figure 5-9. For the directed painting task, VoiceDraw took about one hour, which was 15 minutes longer than Microsoft Paint. For the undirected painting, VoiceDraw took about 3 hours, while a comparable version which the voice painter had previously created on Microsoft Paint was estimated to have taken 9 hours.

5.5.2 Reflections and Lessons Learned

The results from the painting tasks, especially the undirected painting task (Figure 5-9), are confirming of VoiceDraw's design. The detailed views of the paintings in Figure 5-9c and Figure 5-9d indicate the richer vocabulary of strokes supported by VoiceDraw over the voice painter's previous method. VoiceDraw's strokes exhibit unconstrained smooth paths and variable stroke thicknesses. The voice painter expressed great satisfaction in having been able to produce a piece that much more closely mimics his original inspiration, Jackson Pollock.

Another exciting result was that the Vocal Joystick version of the expressionist style painting under the undirected task took almost a third as long as a similar painting using the previous tools, resulting in overall reduced fatigue. The voice painter mentioned that he wanted to take a

[5] Taken from http://soe.ucdavis.edu/ms0506/180E/CheungC/

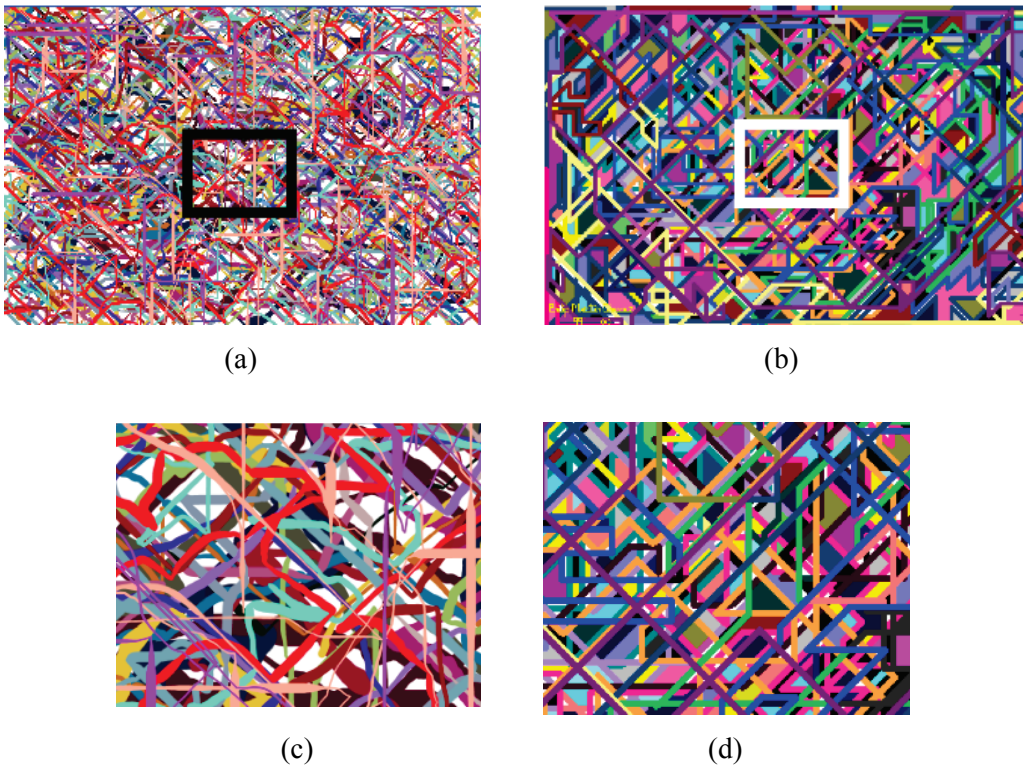


Figure 5-9: Results from the undirected painting task. (a) The result using VoiceDraw. (b) One of the earlier pieces that the voice painter had created using his previous tools from which he drew inspiration from memory to create (a). The detailed view of a region on each painting (shown below each) reveals the difference in the stroke attributes with each tool. The VoiceDraw painting took 3 hours; the Microsoft Paint version was estimated to have taken 9 hours.

sip of water more frequently when using VoiceDraw, but he did not complain of any significant vocal fatigue. The voice painter cited the ability to control all aspects of the program using voice as one of the biggest benefits:

> *"At one point, I realized I went for two hours straight, haven't hit the recliner, haven't used the arm ... it was a big plus not to have to use the trackball ... going in and out of drawing, changing the color, all that was verbal."*

It was also encouraging that the voice painter learned to reliably produce all of the vowel and discrete sounds after 30 minutes of guided training. He memorized the mappings of the vowels to the directions after two days (approximately 4 hours of VoiceDraw use). These observations give us a good indication of the learnability of the non-speech vocalization scheme. In particular, the subjective quality of the voice painter's new work is more fluid and curvilinear than his prior work, and arguably more in the spirit of his inspiration, Jackson Pollock. Most importantly, the voice painter himself thought so.

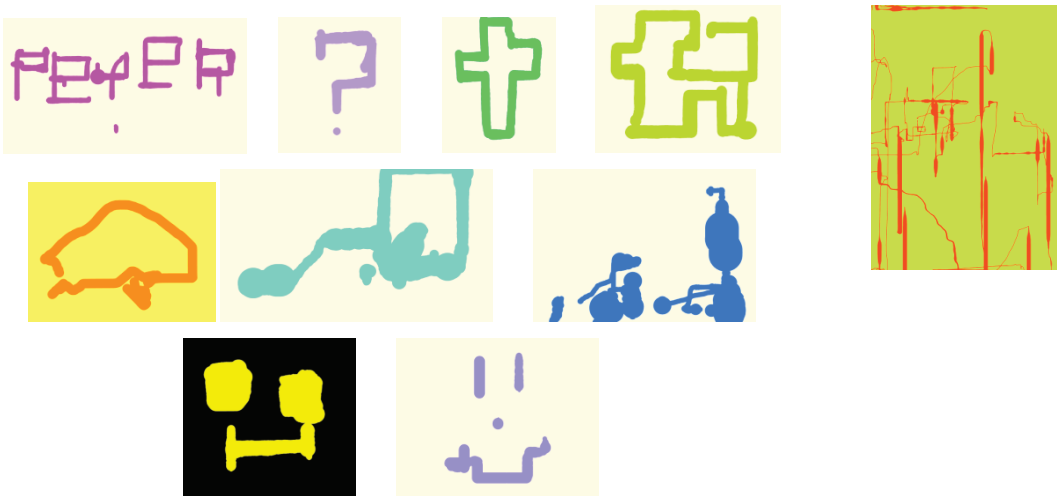


Figure 5-10: Drawings made by children trying out VoiceDraw as part of a public exhibit at the University of Washington. All drawings were produced within several minutes after only a few minutes of training.

5.5.3 Remaining Challenges

Initially, the voice painter expressed a desire to have full program access in VoiceDraw without using any speech commands. However, after he started using the Voice Marking Menu, he decided he preferred speech commands (a reason I offer both). When asked for the reason behind his changing preferences, he stated that he felt more comfortable with the speech commands since he was more familiar with them from his previous setup, and also for their directness compared to the hierarchical organization of the Voice Marking Menu. It remains to be seen whether this preference would change given a longer trial period.

One unanticipated challenge was the limit of the voice painter's lung capacity and its impact on the types of strokes that he could generate. Because each of his utterances could last only one or two seconds, most of his strokes tended to be short segments, and did not fully take advantage of the ability to continuously vary the vowel quality for long smooth curves. Over time, however, as he became comfortable with the vowel mappings, he was able to incorporate longer vowel sweeps and create curves.

5.5.4 Observations from a Public Exhibit

VoiceDraw was showcased at a public exhibit called the Engineering Open House at the University of Washington's Electrical Engineering department, where people from the general public, mainly children between the ages of 7 and 18 (along with their parents), were given the

opportunity to try out the application. Each person was given few minutes of introduction to the vowel sounds and how to control the application, and was then allowed to use the system for a few minutes to produce their artwork. A total of 99 people tried VoiceDraw, most of them using the 4-vowel mode (cardinal directions only) due to limited training. The results (Figure 5-10) demonstrate that VoiceDraw can be used with minimal training for creative self-expression.

5.6 Summary

This chapter presented VoiceDraw, an application that investigates the use of voice as a viable modality for creating freeform drawings on a computer without manual interaction. Through five field investigations involving a self-described electronic voice painter, I identified a number of key design goals and solutions for upholding them. The results from this work suggest that VoiceDraw can significantly enhance the process of voice-based drawing, and that it opens up a new realm of creative expression that was previously not available to people with motor impairments. VoiceDraw creates qualitatively different art from the voice painter's existing tools due to its provision of continuous, fluid control through manipulation of vocal parameters. Since working with Mr. Chavez, VoiceDraw has also been used by an elementary school in the United Kingdom for an art event, as well as by a local disability services agency. The experience gained through building the VoiceDraw application can inform future design of applications optimized for voice.

Chapter 6
VoiceGames

6.1 Motivation

There are several reasons why the exploration of voice-driven input for games is of interest. First, for groups of people with motor impairments computer games and game-like applications such as SecondLife[1] may provide one of the few avenues for entertainment and social interaction that play an important role in improving the quality of life (Riemer-Reiss & Wacker 2000). By investigating non-manual input modalities for game control, computer games can be made more accessible to those with physical conditions affecting the use of their hands, which include 294,000 children in the U.S. under the age of 18 (or one in 250 children) who have been diagnosed with arthritis or other rheumatologic conditions.[2] Second, computer games are beginning to employ modalities beyond the traditional keyboard and mouse or console controllers, as is evidenced by the recent popularity of games such as Rock Band,[3] Karaoke Revolution,[4] and various other games available on systems such as the Nintendo Wii.[5] Voice

[1] http://secondlife.com/
[2] http://www.emaxhealth.com/97/18782.html
[3] http://www.rockband.com/
[4] http://www.konami.com/kr/

input holds potential for expanding the array of input methods available for future games. Third, an increased understanding of how to effectively control computer games can also inform how to better design general user interfaces for interaction using non-manual modalities.

To explore this space, I created a novel voice-based system called the VoiceGame Controller that makes it possible to play a wide variety of games hands-free using voice. To evaluate its efficacy, I recruited a total of five participants with and without motor impairments to attempt to play several categories of games, from classic arcade games requiring simple directional input, to a more complex game such as a first-person shooter, using both a conventional input device and the VoiceGame Controller in turn.

There is little prior work that can be found in terms of generalizable voice-driven input techniques for computer games. Most of what exist can be classified as either individual games specifically designed to use voice input in their own way within the game, or tools that map speech commands to key stroke combinations.

6.1.1 Games Specifically Designed to be Controlled by Voice

There have been a number of games designed specifically to be controlled using the player's voice. Most of them use relatively simple vocal features such as pitch and volume. Commercial games such as Karaoke Revolution and Rock Band track the pitch of the players' singing and generate a score based on how close they are to the target pitch track.

PAH![6] and shout n dodge[7] are Flash-based side-scrolling shooter games that continuously map the volume of the microphone input to the vertical position of a spaceship. PAH! also enables the user to shoot by making the plosive sound "pah." Racing Pitch[8] uses the pitch of the audio input to control the speed of a slot car, with higher pitch vocalization causing the car to travel faster.

Similarly, in academia Igarashi presented a set of simple games (Igarashi & Hughes 2001) that demonstrated the use of pitch changes and sound detection to control simple movements of the game character. His games included a side-scrolling shooter style game, in which the player can change their character's vertical position to avoid incoming obstacles by making rising or falling

---

[5] http://wii.com/
[6] http://www.designer.co.il/pah/
[7] http://www.weebls-stuff.com/games/shout+n+dodge/
[8] http://secretexit.com/games/racingpitch

pitch sounds, another side-scrolling game, in which the player could make the character jump for varying durations determined by the duration of any vocalization to avoid rolling obstacles, and a simplified shooter game in which each utterance of a plosive sound such as "tah" caused the trigger to be fired. In the Humming Tetris, Sporka et al. (2006) mapped various pitch inflection "gestures" to various controls such as move left, move right, and rotate.

6.1.2 General Tools to Control Games Using Speech

Speech input in games has primarily been used for communication among multiple players (Gibbs et al. 2006; Wadley et al. 2005) or for command-and-control-style applications (Tse et al. 2007). There have also been some tools developed to enable a player to use their voice to control various aspects of computer games. Most common are tools that translate spoken commands into specific actions, usually in the form of keystroke combinations. Tools such as Shoot,[9] Voice Buddy from eDimensional,[10] and VR Commander[11] have been specifically marketed for voice control of games, and map spoken commands to a sequence of keystrokes. Such functionality can be quite useful in games such as flight simulators in which there are a large number of possible commands that can be executed at any one time that are often mapped to obscure key combinations. Other general "speech macro" tools such as Vocola[12] and WSR Macros[13] offer similar functionality, but do not include any game specific mappings. Most of these tools act as keystroke generators, mapping various spoken utterances to keyboard events, and are often used in conjunction with the keyboard or mouse as an augmentative modality. One obvious limitation of these tools is that they cannot be used on their own to play games that require input from pointing devices such as the mouse or joystick.

6.1.3 Controlling Games Using Today's Speech Recognizers

Within the context of computer games, dictation may be used to utter phrases in various chat modes, especially in multiplayer games that support text communication among players. As I mentioned in the previous subsection, command-and-control can be used to map phrases to discrete inputs, most often keyboard events and possibly mouse button events.

---

[9] http://clans.gameclubcentral.com/shoot/
[10] http://www.edimensional.com/index.php?cPath=23
[11] http://www.vrcommander.com/
[12] http://vocola.net/
[13] http://code.msdn.microsoft.com/wsrmacros

Aside from dictation and command-and-control, there are also capabilities available in today's commercial speech recognizers that enable the execution of pointing tasks using speech. Features such as the MouseGrid and SpeechCursor, as described in Chapter 2, offer the ability to control the mouse pointer to a certain extent. Although these features do enable basic pointing and clicking and steering tasks to be performed using speech, the significantly longer time it takes to do so compared to the mouse and their discrete nature makes them unsuitable for any task that requires rapid pointing or smooth steering, as games often do.

6.2 Input Taxonomy of Computer Games

To better understand the current limitations of speech as a game control modality and to identify potential areas of improvement, I created a simple taxonomy of computer games based on the types of input devices and the nature of input signals that the game expects (Table 6-1). This is by no means an exhaustive categorization of computer games. Instead, I chose the dimensions of the taxonomy to highlight the characteristics of computer games that currently make them especially challenging to play using speech input and that also present opportunities for innovation.

I placed the focus on computer games that are currently available for personal computers, and thus excluded games for dedicated gaming consoles. While I believe that exploring games for other platforms will be equally interesting and valuable, my current focus is towards understanding and iterating upon ways to make the large number of existing applications accessible via voice. As such, the two primary input devices I include as part of my taxonomy dimensions are mouse and keyboard.

The first two columns of Table 6-1 contain the category description and examples of games that fit into each category. The first two dimensions of the taxonomy simply classify whether the game requires keyboard or mouse input. Dimension C indicates whether or not it is important for the game to be able to receive rapid continuous stream of input signals that possibly vary over time, such as receiving key repeat events from a key being repeatedly pressed or held down for some duration in the case of a keyboard, or a stream of changing 2-D coordinates in the case of a mouse. If the value of this dimension is a no, then for keyboard input, it represents games that simply need to receive individual key press events, and for the mouse, games that simply need pointing and clicking. Dimension D indicates how important it is that the delay between the moment when the player decides to take some action and the moment when the game receives the

Table 6-1: Input taxonomy of computer games, highlighting the factors that make them challenging for current speech input methods to control and that provide an opportunity for improvement. The two right-most columns offer a subjective rating of the "playability" of the games in each category using either the speech modality or the Voice Games Controller.

Category	**Examples**	A: Requires keyboard?	B: Requires mouse?	C: Δ/time important?	D: Timing important?	Playable with speech?	Playable with VG Controller?
Strategy	Sokoban, Two Rooms	Y	N	N	N	Y	Y
Skill and action	Frogger, Pac-Man	Y	N	N	Y	N	Y
Skill and action	Tetris	Y	N	Y	Y	N	Y
Strategy	MineSweeper	N	Y	N	N	Y	Y
Skill and action	Duck Hunt	N	Y	N	Y	N	Y
Skill and action	FishTales	N	Y	Y	Y	N	Y
Strategy	SecondLife	Y	Y	Y	N	Possibly	Y
Skill and action	Doom	Y	Y	Y	Y	N	Possibly

input be as short as possible. Broadly speaking, dimension D distinguishes between two classes of games that Chris Crawford defines in his seminal book *The Art of Computer Game Design* (Crawford 1984) as "skill-and-action ('S&A') games (emphasizing perceptual and motor skills) and strategy games (emphasizing cognitive effort)."

The last two columns provide indications of the playability of each category of games via existing speech input methods or the VoiceGame Controller. While the notion of "playability" is a highly subjective measure, the description for how they were rated for the speech modality based on its input characteristics is described in the following section.

I placed the focus on existing computer games, and as such, on games that are primarily designed to be played using the keyboard and mouse. This is so that the input techniques that arise from the investigation into playing such games using voice can then be applied to the control of other types of applications on a personal computer.

6.3 Characteristics of Speech as Game Control

Below are some general observations about the speech input modality that make it ideal for certain types of games and limited for others in the context of a personal computer. I focus on issues related to interaction methods, rather than on other unrelated yet important issues such as social appropriateness and environmental effects.

6.3.1 Strengths

Due to the richness of human language, there are hundreds of thousands of words and phrases that can be uttered by a person, which can in turn be mapped to just as many discrete commands to be executed on a computer. For computer games, this can be beneficial as there are far fewer reasonable keystroke combinations that can be used for the same purpose. The ability to utter commands that express the actual intended action in natural language rather than needing to remember and recall arbitrary keystroke combinations can also offer an easier learning curve for novice users and potentially a boost in performance (Tse et al. 2007).

6.3.2 Limitations

The time it takes a person to complete uttering a word or a phrase can be a significant factor in games that require sub-second timing. The processing time required to recognize the spoken utterance also adds to this delay (see Section 4.4).

Related to this limitation of delay is the limitation on the maximum number of utterances that can be recognized within a short time period. While it is possible to pre-map a speech command to rapidly generated a sequence of events, when the need arises for dynamically generating such an input sequence at a rapid and varying rate, the per-utterance delay imposes a limit on how many such utterances can be recognized within the short timeframe.

Another major limitation of speech recognition-based input is its inability to specify smoothly varying input. Output from a speech recognizer is by nature discrete, as the output is not generated until a word or a phrase has been recognized, and the result of a single recognition is a single event, with the recognized utterance as the parameter. This discrete nature makes it significantly challenging to specify continuous input such as smooth motion of a pointer through two-dimensional space. While techniques such as MouseGrid and SpeechCursor do exist for controlling a pointer through speech, they are significantly hampered by their discrete nature and are not suitable for situations requiring quick variation of movement speed or direction, which is

essential in many computer games (Table 6-1). Referring back to the taxonomy, the games that can be controlled effectively by current speech input methods are constrained to strategy games that do not require rapid sequences of input or precise timing.

6.4 VoiceGame Controller

I developed a system called the VoiceGame Controller to address the three primary limitations of current speech-based computer game input, namely the execution delay, consecutive input latency, and the discrete nature of pointer control. The VoiceGame Controller augments, rather than replaces, the current speech-based input methods by providing the option to use the dictation and command-and-control modes when non-time-critical speech input is appropriate.

The VoiceGame Controller is built upon the Vocal Joystick engine. Aside from using the Vocal Joystick engine for mouse pointer control, the vowel probabilities are also used to generate discrete output, wherein for each frame, an output can be generated indicating which of the nine vowel sounds the incoming sound most likely represents. Due to the fact that the per-frame vowel probabilities can be reported starting at the *onset* of the vocalization rather than having to wait for the end of the utterance in the case of speech recognizers, the feedback can be provided to the user immediately and continuously as the user vocalizes. The engine can also recognize non-vowel sounds, or discrete sounds, such as “ck” and “ch.”

I continue to leverage the pointer control capability of the original Vocal Joystick application, which in itself already expands the realm of games playable by voice to include the three categories of mouse-only games in Table 6-1. I also added several new interaction techniques in the VoiceGame Controller: voice gestures, keyboard input, and integration with existing speech-based input.

6.4.1 Voice Gestures

To address the delay and latency issues of speech commands, I built a set of input methods I call *voice gestures* that take advantage of the immediate availability of vocal features from the Vocal Joystick engine while also being able to generate discrete output.

The four voice gestures I currently support are *single vowel*, *long vowel*, *double vowel*, and *pitch sweep* (Figure 6-1). Aside from the gesture detection events that can be fired for each voice

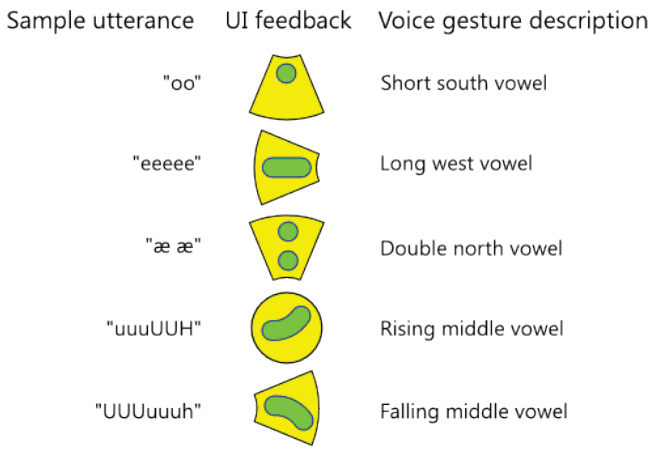


Figure 6-1: Sample of each voice gesture type recognized by the VoiceGame Controller.

gesture, there are also corresponding gesture-started and gesture-ended events that provide greater control over when to initiate some action in response to a voice gesture.

The short vowel voice gesture is simply a short sustained utterance of a single vowel sound. The actual duration that defines a short vowel is configurable for each user, but the default range is 40 milliseconds to 300 milliseconds. The long vowel voice gesture is simply a longer version of the short vowel voice gesture, with default duration range of 600 milliseconds to 1500 milliseconds. This voice gesture is inspired by Goto et al.'s "filled pause detection system" (Goto et al. 1999), in which the system recognizes uttered hesitation (e.g., "umm…") to trigger various functionality such as auto-completion of a partially uttered search query. The double vowel voice gesture comprises two short vowel voice gestures of the same vowel sound uttered in sequence, with a default maximum inter-vowel pause of 500 milliseconds. The pitch sweep voice gesture involves either increasing or decreasing the pitch while uttering a vowel sound, resulting in ether a rising voice gesture or a falling voice gesture. This voice gesture is inspired by Sporka's humming and whistling interfaces (Sporka et al. 2004), which used various pitch inflection patterns to map to input events such as movement of the pointer and pressing of arrow keys.

6.4.2 Keyboard input

To be able to play keyboard-based games, I include the ability to map the vowel sounds to key presses. The VoiceGame Controller can generate key events such as key down, key up, and repeating events to simulate a key being held down. These events can be mapped to any

combination of a particular vowel sound (one of nine) and a corresponding voice gesture (upon onset or upon detection), as well as by one of the discrete sounds.

6.4.3 Integration with Traditional Speech Recognizer

I leverage the power of traditional speech recognizer's command-and-control capabilities by integrating with the Windows Speech Recognizer, part of Windows Vista and later Microsoft operating systems. The command-and-control mode can be used for issuing one of many commands. However, due to the limitations of speech input described earlier, this input mode is more suitable in situations when the input is not expected to be time-critical, such as activating controls in a flight simulator or specifying actions in SecondLife.

6.5 Evaluation of VoiceGame Controller

The results from the comparative analysis of reaction times of key, voice, and speech input (see Section 4.4) showed that the voice input modality has the potential to offer significant performance gains over general speech input, approaching the performance of key input. To determine how this advantage translates to actual game play, I conducted a preliminary evaluation of the VoiceGame Controller with various types of computer games.

6.5.1 Games Examined

To examine the potential of the VoiceGame Controller to control various types of computer games, I chose four games from different classifications in the taxonomy presented in Table 6-1.

Two Rooms

Two Rooms from Armor Games[14] (Figure 6-2a) is a puzzle game in which the player controls two square pieces, each in isolated "rooms," by using the space bar to toggle between the control of each piece and arrow keys to move the active piece around the puzzle space. The objective of the game is to maneuver the pieces into a goal region by navigating around sliding doors that can be opened by crossing over corresponding buttons. Although there are certain levels in the game that require precise timing in toggling between the pieces, the primary emphasis of the game is on the ability to maneuver the pieces in the desired directions accurately to navigate around doors without accidentally crossing any unintended buttons. While the main objective is to solve each level's puzzle, there is also a secondary metric of completion time that is reported for each level.

[14] http://armorgames.com/play/3006/two-rooms

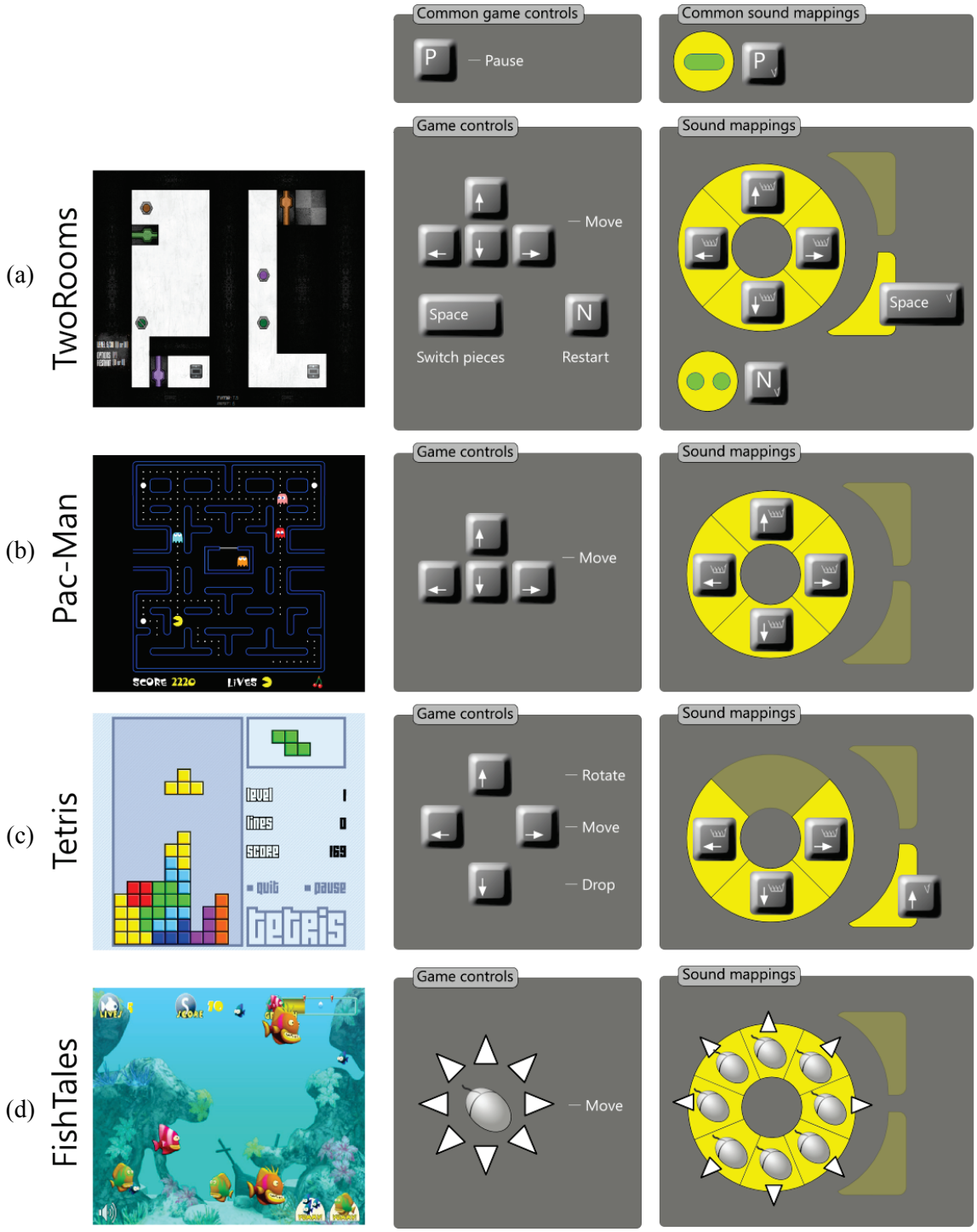


Figure 6-2: Native input mappings (middle column) and the corresponding VoiceGame Controller voice mappings (right column) for each of the four games (left column). The native input mappings show the keys (or mouse) used and their corresponding actions in the game. The VocieGame Controller voice mappings show which vowel or discrete sounds (represented using the vowel compass introduced in Figure 3-4) are used to generate the corresponding key (or mouse) events. For sounds mapped to key events, sustaining that sound emulates the key being held down throughout the duration of the vocalization.

Pac-Man

Pac-Man (Figure 6-2b) is a classic arcade game in which the player controls a yellow character through a maze, attempting to eat all the white dots while avoiding the four ghost characters. The Pac-Man character is constantly in motion with a fixed speed, and the player can change the direction of movement by pressing one of the four directional arrow keys. The game requires that the player be able to issue such directional changes at the precise moments so as to avoid contact with a ghost character, or to make a turn into a maze corridor before moving past it. The player's score corresponds to the number of white dots that the Pac-Man consumes, as well as the number of ghost characters that are eaten when under the influence of a power pellet.

Tetris

Tetris (Figure 6-2c) is a classic puzzle game in which the player attempts to rotate and position a sequence of falling two-dimensional tiles into place such that they fit to form solid horizontal lines, which then disappear from the playing field. The objective of the game is to continue to arrange the tiles to make lines disappear until the tiles stack up to the top of the playing field and no space remains for additional tiles to be introduced. The score increases for each line that disappears as well as the number of lines that each new piece drops. As the pieces drop discretely at a constant rate, the player can move the piece left or right using the arrow keys, rotate the piece by 90 degree increments by pressing the up arrow key, or cause the piece to drop faster by holding down the down arrow key for the desired duration. To be able to quickly move a piece into the desired position, it is necessary to be able to execute a continuous sequence of left or right movements by either repeatedly pressing or holding down the corresponding key.

FishTales

FishTales[15] (Figure 6-2d) is a Flash-based game in which the player controls a red fish in a two dimensional scene by directly moving the mouse to the desired position in the scene. The objective of the game is to eat fish that are smaller than the player's fish by moving the player's fish over the target fish, while avoiding contact with larger fish. The player's fish grows in size as it eats a certain number of smaller fish, enabling it to eat larger fish as the level progresses. The score is based on the number of fish of various sizes that the player's fish was able to eat. Because all of the fish are moving at varying speeds and direction, it is necessary that the player be able to smoothly and rapidly steer the mouse pointer.

[15] http://flashgamesite.com/play334game.html

6.5.2 Evaluation Setup

I conducted an evaluation of the VoiceGame Controller with five participants, two of whom had some form of motor impairment (one with muscular dystrophy and the other with arthrogryposis multiplex congenita). Each participant came in for four 90 minute sessions, each separated by no more than two days, for a total of six hours per participant.

During each session, the participant was given the chance to play each of the four games described above using the VoiceGame Controller. In the first session, the participants were introduced to the VoiceGame Controller system and the directional vowel mappings, and were then guided through the process of adapting the sounds and tuning the responses. They were also introduced to each of the four games, and were given the chance to play them using their native device for five minutes each. Throughout the four sessions, each participant had the chance to play each of the four games for about 45 minutes total using the VoiceGame Controller.

At the end of the final session, a timed evaluation was conducted in which the participants were asked to play each of the four games for five minutes using each of the following input modalities: VoiceGame Controller, speech, and native device. For the speech modality, speech commands were mapped to each of the key events in the keyboard-based games (e.g., uttering "left" in Tetris will simulate pressing the left arrow key once). The speech input modality was omitted for the mouse-based Fish Tales game because the existing speech-based cursor control methods of Speech Cursor and Mouse Grid were too slow to be able to play the game during pilot testing. For the native device modality, the three participants without motor impairments used the mouse and keyboard, while the participant with muscular dystrophy used the touchpad and keyboard and the participant with arthrogyposis used the mouth stick to operate the keyboard, including the MouseKeys feature for controlling the pointer.

6.6 Results

Figure 6-3 summarizes the quantitative measures from the evaluation described above. The data is presented in two groups, with the first group (NMI) representing the averages across the three participants with no motor impairments, and the second group (MI) representing those across the two participants with motor impairments. The bar graphs compare the performance of each type of input modality relative to the VoiceGame Controller, across the four games. For the Two Rooms game, the measure of performance was based on the average completion time across all

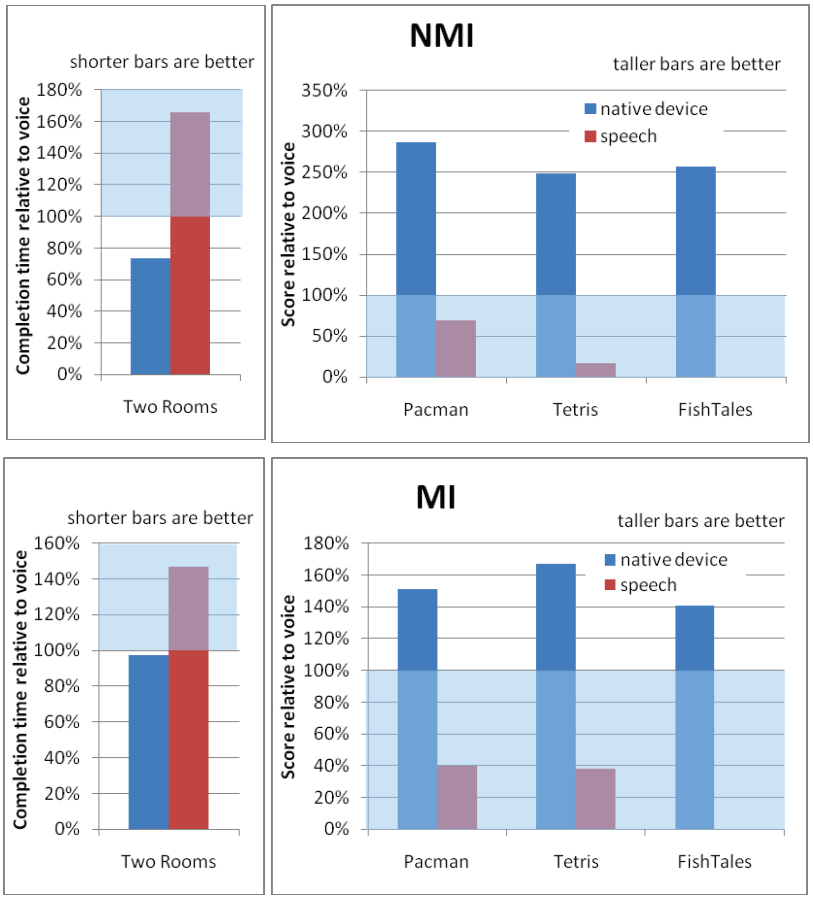


Figure 6-3: Average VoiceGame Controller score for Pacman, Tetris, and FishTales relative to each of the other two modalities. The average level completion times for Two Rooms are shown in separate graphs, since in that game, lower time is better.

the completed stages. As such, a shorter time and thus a shorter bar in the graph represents better performance. The other three games were measured based on the average score achieved, in which case a larger number and thus a taller bar in the graph represents better performance. In both cases, the vertical axes represent those performance measures relative to the VoiceGame Controller, with 100% representing the performance achieved by the VoiceGame Controller on average. Shaded regions indicate performance measures that are worse than the VoiceGame Controller.

Compared to the speech modality, the VoiceGame Controller performed significantly better across all games for both groups. For FishTales, the fact that the participants were able to play it at all using the VoiceGame Controller is a clear win over the speech modality under which the game could not be played. Within the NMI group, VoiceGame Controller was 40% faster than

speech in Two Rooms, and the average score was 1.5 times higher in Pacman and 6 times higher in Tetris. Within the MI group, VoiceGame Controller was 32% faster than speech in Two Rooms, and the average score was 2.5 times higher in Pacman and 2.6 times higher in Tetris.

The better performance by the native devices was expected, as was previously seen in the comparative studies described in Chapter 4. Within the NMI group, the VoiceGame Controller performance was 40% worse than the native device for Two Rooms, and between 60% and 65% worse for the other three games. For the MI group, the VoiceGame Controller performance was comparable to their native devices for Two Rooms, and between 60% and 70% for the other three games. While even for the MI group the native device performed better than the VoiceGame Controller, both of the MI participants expressed that the physical fatigue induced by attempting to manipulate their physical devices was less desirable than the hands-free nature of the VoiceGame Controller. None of the participants indicated vocal fatigue to be an issue throughout the study.

6.7 Summary

This chapter presented VoiceGame Controller, a program that leverages non-linguistic voice input to offer fluid and rapid input for controlling a variety of computer games without needing to use one's hands. The significantly better performance obtained by the VoiceGame Controller compared to standard speech-driven input suggests that voice input can become a viable modality for people with motor disabilities to play many of the mouse and keyboard-centric games that have previously been beyond reach for them. The expressiveness and non-manual nature of voice-driven input can also serve as a new game input modality for the general population, expanding the realm of interactive entertainment.

Chapter 7
VoicePen*

7.1 Motivation

Pen-based input on tablet computers is used for a wide variety of applications such as note taking, sketching, architectural design (Gross & Do 1996), graphic design, website design (Newman et al. 2003), and animation (Davis et al. 2008). Typically, users manipulate a pen on the tablet's surface by controlling pen position, whether the pen is touching the surface, and the state of the pen's barrel button(s). This allows users to draw brush strokes, click buttons, and operate menus.

However, many tasks can benefit from simultaneous continuous manipulation of multiple parameters. For example, in drawing applications, one may want to change brush thickness and opacity at the same time (Figure 7-1), or change the color of the brush while stroking (see Figure 7-2 for examples of other brush parameters that may be continuously manipulated). This is in contrast to discrete manipulation such as picking a different brush type. When creating two-dimensional animations, the user may wish to specify moving elements that not only change their location but also their orientation or scale simultaneously.

* Parts of this chapter are adapted from (Harada et al. 2007a), a joint work with primarily T. Scott Saponas.

Figure 7-1: Example of a drawing created with pen input augmented with non-linguistic vocalization. The pen pressure controlled the brush thickness and non-linguistic vocalization controlled the opacity simultaneously as the strokes were being drawn.

The digital pen is a well-suited input modality for positioning, direct manipulation, sketching, and performing other single-handed input on a tablet computer. Its primary advantage is that it allows a person to use their fine motor control for direct manipulation of objects as well as leverage already acquired skills of using analog drawing instruments. The pen input space usually consists of at least the pen position above or on the screen surface and whether or not the pen is in contact with the surface. In some cases, pens can also indicate a pressure being applied by the user, the state of one or two barrel buttons on the pen, pen angle to the surface, and pen height from the surface (when hovering), although many of these parameters are available only on high-end non-display tablet devices. Tablet PCs make use of this input space in traditional WIMP interfaces as well as in those specific to digital pens, such as Windows Journal or Denim (Newman et al. 2003). Position is usually used for targeting actions, such as clicking a button or menu, or for making brush strokes when the pen is dragged on the surface. Barrel buttons are used for actions such as bringing up context menus. Pen pressure is often used for controlling brush stroke thickness or

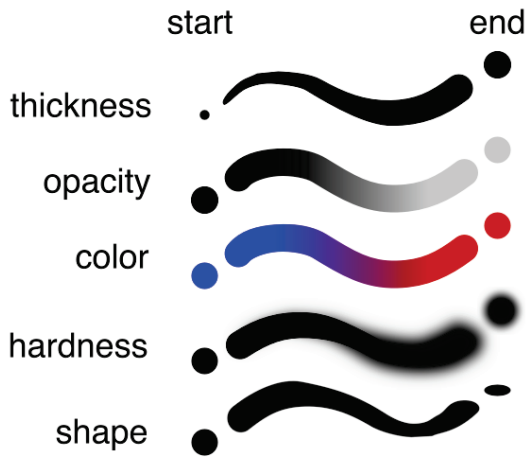


Figure 7-2: Various parameters of a brush stroke that may be manipulated while the stroke is being drawn.

invoking a context menu. However, in most cases, at most one additional continuous parameter can be manipulated in parallel with the pen position.

Investigations into bimanual interfaces such as Bricks (Fitzmaurice et al. 1995) and TouchMouse (Hinckley et al. 1998) offer some promise for controlling multiple continuous parameters simultaneously. For example, a track pad or keyboard could be used to control a drawing brush's opacity while the pen's pressure controls thickness. However, tablet PC's, PDAs, or tabletop displays do not have a readily accessible keyboard much of the time and as such there are limited possibilities for traditional bimanual interaction.

In this project, I investigated voice as a potential secondary modality due to its low cost and expressiveness. Previous work (Cohen et al. 1997; Oviatt et al. 2000) has utilized speech to augment pen in the form of spoken commands, such as saying "show me all the hotspots in this area" while drawing a circle on a map. However, such discrete commands do not allow fluid control of continuous parameters. Instead, I focused my investigation on the use of non-speech vocalization as an augmentative input modality for fluid control of continuous parameters.

I propose four new voice-augmented pen input techniques primarily related to stroke-making. I describe how varying loudness while making continuous vowel sounds can be used to control continuous parameters such as the brush drawing parameters of thickness and opacity. Focus is placed on tasks that require simultaneous input of multiple continuous values.

The rest of the chapter is organized as follows: first, I survey the related work in bimanual input and integration of pen and voice as parallel input modalities. I then describe the four interaction techniques that I have developed that offer voice-augmented pen input for simultaneous manipulation of multiple parameters. Next, I describe the prototype implementation of these interaction techniques in a system called VoicePen, and discuss the results of a feasibility study I conducted to explore the potential of the four techniques.

7.2 Related Work

I looked to applications of bimanual interaction as a potential source of interaction techniques in which the input parameter controlled by the second hand may also be controlled by voice. A large body of work has examined the domain of bimanual interaction. I survey only a small sample of work from this research domain that has fostered many novel and interesting interaction techniques. I also discuss previous research that examines the use of pen pressure in tablet applications, novel drawing applications, and uses of spoken input in conjunction with pen input.

7.2.1 Bimanual Interaction

Much research has been directed at investigation of using two hands to interact with computers. From this work a number of interaction techniques have been proposed. Bricks, one of the earlier systems presented by Fitzmaurice et al. (1995), demonstrates how simple graspable input devices manipulated by both hands can significantly enhance the input vocabulary. The Toolglass and Magic Lens widgets by Bier et al. (1993) enable the use of the non-dominant hand for positioning a see-through tool palette for operations by the input device in a user's dominant hand. Kurtenbach et al. (1997) also explored the use of similar widgets in the context of a commercial paint system using a puck and stylus. Hinckley et al. (1998) explored the use of a new input device called the TouchMouse along with a standard touchpad for performing map navigations.

Researchers have also evaluated the human capability and efficacy of many bimanual input techniques (Balakrishnan & Hinckley 1999; Buxton & Myers 1986; Hinckley et al. 1997). Such work describes the benefits of bimanual interaction, including the lowered cognitive load compared to unimodal interaction and time efficiency due to parallelization (Leganchuk et al. 1998; Owen et al. 2005). However, Kabbash et al. (1994) suggests bimanual interaction may not yield better performance when not designed appropriately. Many of these research activities were spurred by the initial theoretical framework for bimanual interaction called the kinematic chain

presented by Guiard (1987). My exploration of using voice as the secondary modality to augment pen input draws inspiration from these projects exploring bimanual interaction.

7.2.2 Pressure Input

A number of input techniques have been proposed that leverage the pressure sensitivity of digital tablets. Ramos et al. (2004) present a study exploring the human ability to specify distinct stylus pressures and a taxonomy of "pressure widgets" that leverage this ability. Another technique called Zliding, also proposed by Ramos et al. (2005), examines stylus pressure for control of the scale of zooming. They compare their zooming method to using discrete keys and an isometric input device. They found their technique to be comparable to the other alternatives, and cite the users' preference for their technique in being able to perform the task with just one hand. Ramos and Balakrishnan (2007) have also suggested a technique called pressure marks, where the user varies the pen pressure throughout a stroke to issue commands. They also situate a number of pressure-based interaction techniques within the context of an application for annotating digital video sequences (Ramos & Balakrishnan 2003). My work utilizes vocal parameters in a manner similar to pen pressure, i.e., as a continuous input modality, which suggests its potential applicability in similar interaction methods as those described above involving pen pressure. Further investigation is necessary to determine how the controllability of vocalization compares to that of pen pressure.

7.2.3 Drawing

Several systems have been proposed to investigate the use of bimanual interaction techniques in the domain of artistic drawing and other creative activities. HabilisDraw (Butler & Amant 2004) allows a user to manipulate various virtual drawing objects projected on a touch sensitive table using both hands. Raisamo (1999) presents the use of a trackball and a mouse to manipulate virtual carving sticks to sculpt a two dimensional block. However, none of them offer continuous and simultaneous control of multiple drawing parameters in conjunction with a digital pen.

7.2.4 Pen + Voice Input

Many researchers have explored combining pen input with speech (Cohen et al. 1997; Julia & Faure 1995; Oviatt et al. 2000). However, most of the work focuses on the integration of spoken commands with pen gestures, and does not explore the use of continuous non-linguistic vocalizations as a form of input. There have been a number of voice-based systems that attempt

to provide continuous control using non-linguistic vocalizations. Igarashi et al. (2001) proposed the use of non-linguistic vocal parameters as a means for achieving direct manipulation via voice. Subsequent systems, in addition to the Vocal Joystick described in this dissertation, have used voice for continuous input (Mihara et al. 2005; Sporka et al. 2004), however primarily as a replacement for mouse pointer control and not in the context of multimodal augmentation of a pointing device.

7.3 Interaction Techniques

Given these characteristics of pen and non-linguistic vocalization, I sought to explore the potential of combining the two modalities for performing the following three categories of tasks on a tablet PC. The first category of tasks I explored is brush stroke creation. Here the user's objective is to draw a certain stroke using the digital stylus as a virtual brush. In existing systems, a user may pre-select a brush color, shape and size, and begin creating the stroke using the stylus, possibly modifying the pressure of the pen against the tablet surface to continuously vary the thickness of the stroke if the tablet is pressure sensitive. The second category of tasks is object manipulation. Here the stylus is used to "pick up" objects displayed on the screen and move them about the workspace, possibly manipulating their geometric features, e.g., by scaling and rotation. The third category is workspace navigation. Here the user controls the current view of the workspace they are interacting with by performing actions such as zooming, scrolling and panning. In cases of applications that deal with three-dimensional representations, navigation may involve a more complex manipulation of the viewport location and orientation.

I implemented four pen-voice interaction techniques (Figure 7-3) to investigate the feasibility of combining pen and non-linguistic vocalization to support the above categories of tasks. In all cases, the main principle is to augment pen input with voice, providing continuous and simultaneous input to manipulate an additional parameter beyond those able to be controlled by the pen position and pressure. I focus on vocal manipulation of a single scalar parameter value, and use two vowel sounds ("aw" and "oo") to either increase or decrease the value. Due to the inherent arbitrariness of mapping vowel sounds to a change in some value, any other of the many possible vowel combinations could have been used. The two vowels I used were chosen based on the relative ease of maintaining the mouth shape while vocalizing the vowels. The loudness of vocalization is mapped to the rate of change in the corresponding direction. The parameter value

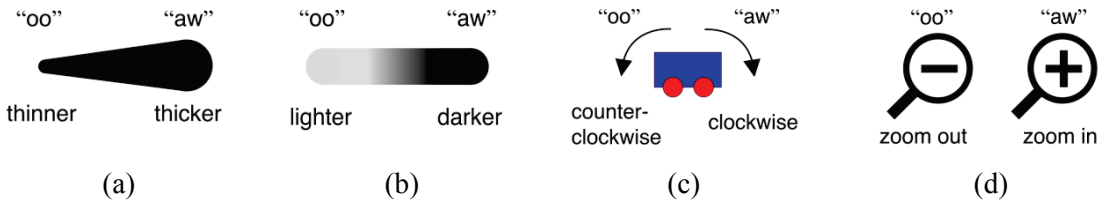


Figure 7-3: Four interaction techniques provided by the VoicePen prototype. (a) In thickness control mode, uttering "oo" continuously decreases the brush thickness while "aw" increases it, as long as the user sustains the utterance. The user can vary the loudness of the utterance to control the rate of change in either direction. (b) In opacity control mode, uttering "oo" continuously decreases the opacity while "aw" increases it, as long as the user sustains the utterance. (c) In rotation control mode, uttering "oo" continuously rotates the object counter-clockwise while "aw" rotates it clockwise, as long as the user sustains the utterance. (d) In zoom control mode, uttering "oo" continuously decreases the current zoom level while "aw" increases the zoom level, as long as the user sustains the utterance.

begins increasing or decreasing as soon as the user begins vocalizing, and stops changing as soon as the user stops vocalizing.

7.3.1 Brush Stroke Thickness Control

In the brush stroke thickness mode (Figure 7-3a), the pen is used to specify the path of the stroke, with pressure sensing turned off so that it does not affect the thickness of the stroke. Instead, the vowel sounds are used to control the thickness. The same technique can be extended to control a variety of other brush properties, such as those listed in Figure 7-2. This mode was used to determine whether the user is able to manipulate one parameter using voice while at the same time using the pen to control the position of the brush. This also simulates the situation in which the pointing device may not have pressure sensitivity, such as when using a mouse or a PDA or limited versions of tablet computers. As shown in Figure 7-3a, the user vocalizes the vowel "oo" to make the brush thinner and "aw" to make the brush thicker.

7.3.2 Brush Stroke Opacity & Thickness Control

Under the opacity control mode (Figure 7-3b), the pen's pressure is activated to control the thickness of the stroke, while the user's voice is used to vary the opacity of the ink. Such a mode may be used to create brush strokes with effects similar to watercolor, where the lightness of the ink varies across the stroke. As with the brush stroke thickness mode, a different brush property may be substituted for opacity. This mode was also designed to investigate whether users are able to manipulate two continuously varying parameters simultaneously using two different modalities. The vowel "oo" is used here to make the brush lighter, while the vowel "aw" is used to make the brush darker.

7.3.3 Object Translation & Rotation

Object manipulation (Figure 7-3c) mode allows the user to move an object around the screen by directly tapping down on the object and dragging the stylus around while holding it down on the tablet surface. I augment this basic manipulation with the ability to control the rotational orientation of the object using voice in parallel to the translation of the object's location using the pen. Such a feature may be used when creating an animation, in which an object needs to undergo translation and rotation simultaneously. The technique can also be applied to other transformation parameters such as scaling. Figure 7-3c shows that the vowel sound "oo" is used to rotate counter-clockwise and "aw" is used to rotate clockwise.

7.3.4 Workspace Navigation

I also explored controlling one aspect of workspace navigation using voice, namely zooming in and out of a large workspace (Figure 7-3d). In tasks such as map navigation, exploring the space typically involves having to manipulate a scroll bar or a zoom control widget, requiring the user to take the pointer off of the primary point of interest, when bimanual control is not available. Igarashi et al. (2000) and Jul et al. (1998) show that users can easily get lost when panning or zooming within large documents without proper feedback and navigation control. I explore the ability to use voice to perform the zooming while the pen remains on the primary point of interaction. The sound "oo" is used to zoom out and "aw" is used to zoom in.

7.4 Usability Study

To understand the feasibility of my interaction techniques, I conducted a usability study in the lab with a prototype implementation. Here I describe the prototype, VoicePen, and the tasks that participants engaged in using it. The study results and participant feedback provide insight into the possibilities of simultaneous non-linguistic voice and pen input.

7.4.1 VoicePen Prototype

The VoicePen prototype is written in C# and XAML on the Microsoft .NET 3.0 platform. Vowel recognition and volume tracking is performed using the VocalJoystick engine (see Chapter 3). The canvas and network components are based on the SketchWizard library (Davis et al. 2007). A Compaq TC1100 tablet PC was used to run the VoicePen prototype software.

The interface consists of a *drawing canvas* or *workspace* on a tablet PC (Figure 7-1). The workspace can be put in several modes, one for each of my proposed interaction techniques. Depending on the mode, voice and pen input have different effects on the canvas.

In *brush thickness* mode, the user can draw brush strokes where the thickness of the stroke is adjusted through voice interaction. *Stroke opacity & thickness* mode allows similar brush stroking except thickness is controlled through the pen's pressure on the tablet's screen surface and opacity of the stroke is controlled through voice interaction. In both of these modes, the brush preview next to the canvas shows the current thickness or opacity of the brush, which can be changed even when the user is not actively inking. The third mode is *object translation & rotation* mode; where an object such as the car in Figure 7-3c can be dragged around by the pen while simultaneously being rotated by the user's voice. VoicePen's fourth mode is *canvas zooming* mode. Users can use their voice to zoom in and out of a canvas while simultaneously creating brush strokes. Zooming allows navigation of a much larger canvas than the available screen space as well as the ability to make very detailed drawings. In all of my techniques, the volume of the user's vocalization reported by the underlying Vocal Joystick engine was linearly mapped to the rate of change of a parameter value (e.g., brush size or rotation speed).

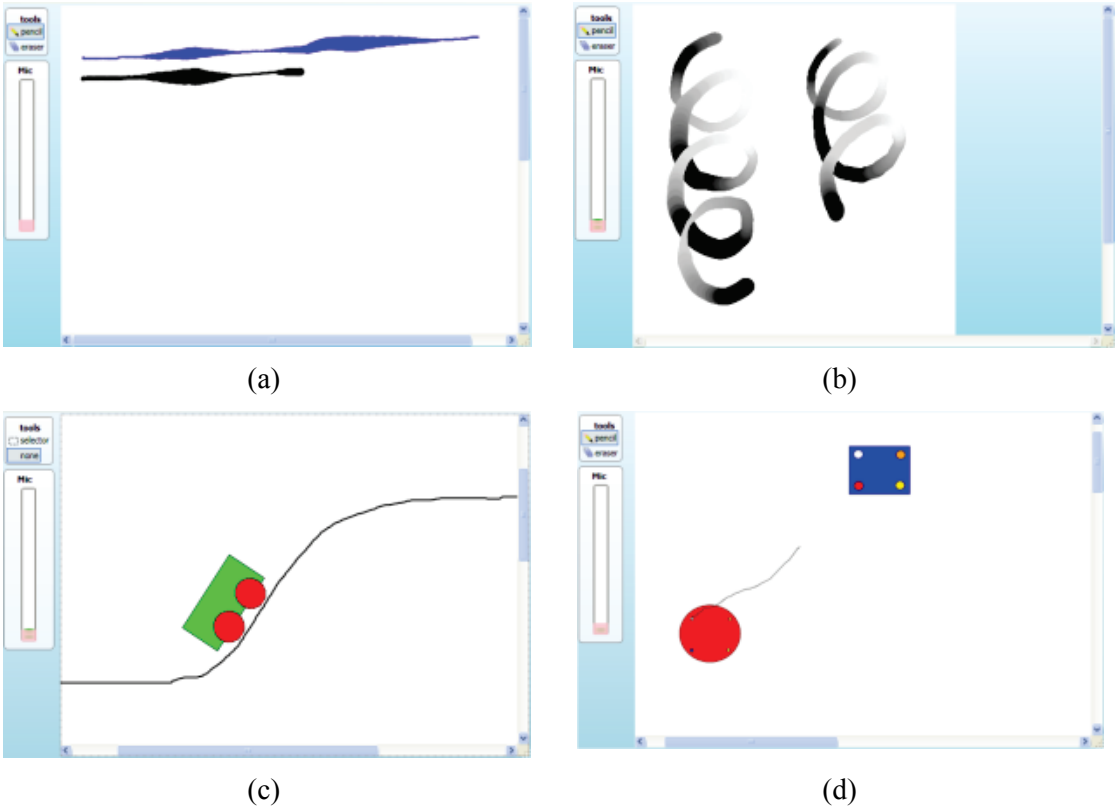
7.4.2 Study Participants and Tasks

Seven people participated in the study. Three were male, four were female, one was left-handed, six were experienced with tablet computers, six were university students, and three were computer science students. Participants ranged in age from 18 to 45. All studies took place in a user study lab. Participants used a tablet PC with VoicePen while wearing a headset microphone. Sessions lasted approximately one hour. Participants' sessions began with a brief demographic questionnaire and VoicePen training to adapt to the user's voice. The adaptation process consists of having the participants make each of the "oo" and "aw" vowel sounds for about two seconds at a comfortable volume.

The study session consisted of participants using each of the four interaction techniques to complete a task. Prior to trials in each mode, I explained to the participants the interaction technique including a printed reference for the vowel mapping (Figure 7-3). Participants were then given a blank canvas to experiment with the current VoicePen mode for a few minutes before starting that task's five trials. For each interaction technique, participants completed five

trials of the task using VoicePen in the corresponding mode. Figure 7-4 shows one trial of each of the tasks. In the cases of *brush thickness* mode and s*troke opacity & thickness* mode, each trial asked the participants to reproduce a target stroke with varying thickness or varying thickness and opacity, respectively. The user was asked to use VoicePen in the current mode to draw a stroke as similar as possible to the target stroke (user strokes were drawn next to target strokes).

In the *object translation & rotation* mode, the tasks consist of using the pen to translate a car image along a path while using voice to rotate the car such that the car's wheels stay parallel to the target path. The participants were asked to try to animate the car with more emphasis on smoothness than precision. Trials of the *canvas zooming* mode consisted of zooming into a small beginning target and starting a stroke in the center of that target and, while continuously stroking, zoom out to find the end target and zooming into the end target to finish the stroke on a very small sub-target point. These targets can be seen in Figure 7-4d.



(a) (b)

(c) (d)

Figure 7-4: Screenshots of the four tasks used in the study: (a) brush thickness task, (b) stroke opacity & thickness task, (c) rotation task, and (d) zoom task.

7.5 Results and Discussion

An emergent theme from the study is that the participants found the first two trials of each technique challenging and the last two easier and often fun. The participants attributed their initial difficulty to several factors: coordinating the control of multiple simultaneous variables (position & thickness, position & rotation, etc.), learning the mapping of vowels to directions of change, adapting to volume sensitivity, and making certain vowel sounds. After only one or two trials, many users said they no longer had to think of the vowel mappings. Users also found that with a little experimentation they had adequate control of volume. Multiple users remarked that they were "happy" or "pleased" with how close their drawings were to the target strokes (This was more common with their last two trials of a task even though those trials were meant to be more difficult). Despite users' lack of familiarity with VoicePen's multimodal input, everyone was able to use VoicePen to complete all the tasks.

Prior to the study I did not know whether using VoicePen for an hour or longer would be straining or tiring. When asked about voice strain only one of the participants said the study was tiring or straining. However, that user also said the continuous vocalizations did not "bother them much" and they would still be happy and able to continue using VoicePen for a while longer.

No single interaction technique was a dominant favorite in terms of difficulty or enjoyment. Some users found the rotation task fun and easy. Other participants really enjoyed and became proficient with the stroke opacity & thickness mode. For example one user remarked while controlling brush thickness with their voice, "this is cool … this is fun … I like this. I think it's pretty easy to control … these are like watercolor or something. They are really pretty." Another user said of drawing using pen pressure for thickness and voice for brush opacity, "This looks so good. This is so much fun. I really like this one." Another participant said, "I see how this could be really useful once you get used to it." I believe the variance in favorite tasks suggests that none of my study tasks are the "killer app" for VoicePen, but that each of the interaction techniques has appeal to some subset of users.

7.5.1 Vowel Mappings

In each of VoicePen's modes, the vowels "oo" and "aw" are mapped to opposite directions of change of a parameter (brush thickness, brush opacity, rotation angle, and zoom level) and volume is mapped to the magnitude of change. I asked each of the participants about the

mappings I chose, other vowel mappings that might make more sense for them, and what other kinds of vocal mappings they would like.

Some users suggested that they might prefer a direct mapping of vowel and volume while performing continuous vocalization. In this situation, if a user wants to make a medium thick stroke, the user must maintain a medium volume continuous "aw" sound. However, these users also said they might not want that type of mapping all the time. They speculated the existing method of vocalization, adjusting a parameter up or down, would probably be best for them in most circumstances.

For brush thickness and zooming, some users said the shape of their mouth when making a vowel sound contributed to whether they liked the mapping or whether it felt natural. One user stated that "it was nice to have 'aw' for going bigger [in thickness] because your mouth is more open." Users suggested that it made sense and was easy to remember if an open mouth vowel, such as "aw," corresponded to thicker or zoomed in and a narrow mouth vowel, such as "oo," decreased thickness or zoomed out.

Some users said that just the tone of the vowels also played a role, that "'oo' and 'ah' for lighter and darker felt really natural because 'oo' has a *light* sound to it and 'ah' sounds *dark*." All of these comments suggest that there may not be a single mapping that is inherently intuitive across users, and that mappings should be customizable by each user based on users' preferences.

Several participants suggested non-vowel mappings such as pitch and loudness for controlling thickness or opacity. One user suggested making a louder vocalization for a thicker brush and a quieter vocalization for a thinner brush. Another user offered the idea of using pitch to control opacity, with high pitch making the brush lighter and low pitch causing the brush to become darker. These non-vowel mappings suggested by the participants are interesting alternatives to VoicePen's vowel mappings and should be explored in future work.

7.5.2 Participants' Techniques & Strategies

While using VoicePen to complete the study tasks, participants developed their own techniques and strategies. One example of this is some participants' use of short loud bursts to make a large change (such as zooming all the way out or reaching the maximum thickness) followed by a quiet continuous adjustment from that new level. Multiple participants discovered this possibility on

their own and found it useful during several trials of both brush tasks as well as zooming tasks. Knowing this user strategy, I could alter the design of the interaction techniques to take advantage of loud vowel bursts.

In the rotation tasks, some users preferred to do rotations in one smooth constant loudness vocalization until the car had rotated to the desired orientation. Other participants found it better to make a series of very short quiet vocalizations until the car was properly rotated. Some users initially tried to rotate the car through one continuous vocalization where they start with their loudness very soft and increase it until the car is rotated properly. However, this leads to users overshooting their desired orientation and engaging in a correction rotation in the opposite direction. These participants eventually adopted one of the previously mentioned approaches of a constant appropriate loudness or multiple short vocalizations.

Participants had various strategies for adjusting brush thickness and opacity. Many users paused their pen movement mid-stroke to adjust the thickness or opacity of the brush with their voice (using the brush preview to the left of the canvas as a guide). However, after a couple of trials, participants began to anticipate brush changes they wanted to make and were able to use loudness and timing to make brush changes without pausing. This approach allowed users to replicate the target strokes by making one continuous fluid stroke of their own. In fact, by the last two trials of each brush task participants needed little or no pausing of the pen for brush adjustment.

Another unanticipated behavior is that multiple users continuously vocalized even when the vocalization had no effect. For example, when the brush thickness reached the maximum value, participants continued to vocalize the vowel despite the absence of a corresponding change. Users said they did this because it felt more "natural" to them to always be vocalizing as long as they were drawing. Users also said they did this because they have a mental model of a direct mapping from vowel and loudness to thickness even though they know that is not how VoicePen works.

7.5.3 Fun & Novel Expressiveness in VoicePen

All of the users remarked multiple times that the VoicePen techniques were fun and enjoyable. When asked about why it was fun, some users said that part of it comes from the novelty of using non-linguistic vocalization as an input mode. Others said that using their voice like this for input was just naturally fun like singing. It appears that the participants were discovering that there is a pleasurable nature to this style of input.

The participants also thought that controlling brush parameters with non-linguistic vocalizations simultaneously with pen input was different enough from traditional tablet PC drawing that it leads to different drawings. I believe this represents expressiveness unique to VoicePen; while there may be drawings that would only be made using traditional digital pen input, there are many drawings that would only be made with VoicePen's new input techniques. This opens an interesting future research direction in how new VoicePen interaction techniques can support new and desirable expressiveness in creative tasks such as drawing, art, or graphic design.

7.5.4 VoicePen Limitations

While VoicePen's interaction techniques offer many benefits, I have identified several limitations of this approach and my prototype implementation. The primary complaint users had about VoicePen was its volume sensitivity. Although the maximum rate of change of a parameter value (e.g., brush size or rotation speed) was normalized based on each user's average volume during adaptation, some users found the mapping to be too sensitive, while others found it to be not responsive enough. Ideally, I would like to allow users to adjust the sensitivity to their preference through a control panel. Further investigation is needed to determine what an ideal mapping function should be between non-verbal vocalizations and control parameters (Malkin et al. 2005). A few participants also had initial difficulty with pronouncing the right vowel sound. This was mitigated by having the participants practice the sound and undergo the adaptation process.

A fundamental issue with VoicePen is that voice is a busy channel. When users make linguistic utterances such as in talking to themselves or someone else they are also uttering the vowels "oo" and "aw" and thus are manipulating VoicePen controls. This means that in scenarios where someone desires to talk while drawing, VoicePen cannot be used. However, some issues with the noisy channel of voice can be minimized by having the system only interpret incoming audio signals when users are actively using the pen. This allows the user to lift the pen from the tablet surface when speaking to themselves or others.

Participants found controlling multiple brush variables, such as thickness and opacity, in addition to pen position, difficult at times. One participant remarked, "it's hard to think of thickness and opacity at the same time." This suggests both that there is some learning time associated with discovering how to control multiple variables at once, and that it is just inherently difficult to

control multiple parameters. However, I observed most participants got a "feel" for it and were able to replicate target strokes.

Certainly there is a social component to the use of voice and using VoicePen in the presence of others can be somewhat embarrassing. In fact, there are numerous situations in which it is probably socially unacceptable to be making non-linguistic vocalizations. However, this is not limited to my techniques; speech input, and sometimes mice and keyboards, can suffer from the same problems. I think there are still many situations where VoicePen may be used without negative social consequences.

Non-linguistic vocalizations such as vowel sounds do not necessarily have obvious, universal, or intuitive mappings to user interface interactions. In my study, not all participants found all vowel mappings easy to remember or intuitive. Some participants mentioned that they had to spend time thinking about which vowel to use. One user said, "I don't think you'll find a good one. You just got to pick one and get used to it… you don't zoom in and out in real life, so there is no natural [mapping]." This problem can be mitigated by enabling users to quickly experiment with different vowel and loudness mappings. The participants also gave me some insights into mappings that might be more universal, such as mouth shape to control brush thickness.

7.6 Summary

In this chapter, I presented the use of non-speech vocalization as an additional modality to the digital pen through my VoicePen prototype for the tablet PC. The study suggests my four interaction techniques may be useful and viable in a variety of digital pen tasks. I was also encouraged by the participants' enthusiasm about VoicePen and believe many more interaction techniques and tablet applications could be built using simultaneous non-linguistic vocalization and pen input.

In the future, I seek to better understand human capabilities for generating and controlling non-linguistic vocalizations in conjunction with pen input through empirical studies in the lab. For example, I hope to discover how many levels of loudness a person can comfortably and reliably generate. I also hope to look at how complex of a drawing people can replicate as well as to what extent people are capable of varying two or three parameters simultaneously and the associated learning curve. I would also like to directly compare existing input techniques to VoicePen. In particular, I would like to compare non-linguistic vocalization to pen pressure for controlling

brush attributes such as thickness, color, curvature, and opacity, as well as to the use of keyboard or other button-based interfaces that allow the attribute value to be changed in a discrete or fixed increment. In the study, I did not have a direct comparison of existing techniques to VoicePen techniques. However, participants did use vocalizations to control thickness in one task and a standard tablet PC pressure scheme for controlling thickness in another task. One user commented on these two methods that using vocalizations was "pretty easy to control" and that it is "hard to control thickness of a line with pen pressure. I feel like I'm drawing the same stroke, but they come out different." Further investigations into such techniques and measures of human capability are needed to establish better foundation for designing effective future VoicePen interaction techniques.

Another future direction I would like to pursue is to create a more complete drawing application that utilizes VoicePen interactions as a test bed for experimenting with other ways of combining non-linguistic vocalization with the digital pen input. Such an application could also allow for a better understanding of VoicePen as an expressive tool for doing creative tasks. Based on the results from the feasibility study, it is clear that VoicePen is a unique experience with its own affordances and benefits. In the drawing case in particular I am interested in working with artists and graphic designers to explore how these techniques can support existing and new types of artistic expression.

The VoicePen prototype and study suggest potential for simultaneous non-linguistic voice input with digital pen input. I found that participants quickly learned the interaction techniques, enjoyed using VoicePen, and found vocalization to provide new expressiveness. Participants developed their own techniques and strategies that worked well and were pleased with how their drawings matched the trial targets. I believe the interaction techniques are feasible and open the possibilities for a variety of new digital pen based input.

Chapter 8
Voice Controller

Throughout the development of the voice-driven applications described in the previous three chapters, the original Vocal Joystick pointer control application has also continued to be improved to incorporate various lessons learned and to make it more practical as an everyday input device for the target user population. As mentioned in Section 3.4, while the original Vocal Joystick pointer control application made fluid voice-driven manipulation of the pointer a reality, it had a long way to go to become a practical tool that could be used by people on their own computers. Some of the main limitations included the fact that it was a command-line program executable only on the Linux platform, and it lacked any graphical user interface for the user to test as well as tune the program to their desired settings. Furthermore, there was no integration with any existing speech recognition software, making it difficult to be used as an augmentation to such tools, thereby limiting its practical utility. Given that the time it takes to reach high proficiency levels with the Vocal Joystick extends beyond a typical single user study session, it was also desirable to create a self-explanatory application that a user could install and use in their own environment to increase the number of experienced Vocal Joystick users who could be recruited for future user studies.

I have attempted to address many of these limitations in the development process leading up to today's version of the integrated Vocal Joystick pointer-control application, called the Voice Controller (not to be confused with the VoiceGames Controller from Chapter 6). The Voice Controller has three key areas of enhancements beyond the original pointer-control application: visual and interactive adaptation interface, real-time visual feedback, and integration with Windows Speech Recognizer (WSR). At a very high level, the Voice Controller system runs in the background on the user's computer, operating in either the *speech input mode* or *Vocal Joystick mode*. By default, the system starts out in the speech input mode, in which the standard WSR commands and dictation features are available to the user. When the user wishes to control the mouse pointer, the Vocal Joystick mode can be initiated using a specific speech command, at which point the user can start uttering the Vocal Joystick sounds to manipulate the pointer. The user can easily switch back to the speech input mode to resume issuing spoken commands or dictating, as well as open a graphical adaptation interface to tune and adjust the Vocal Joystick sounds. The following sections highlight the features from each of the three key areas of enhancements and their underlying design rationale.

8.1 Adaptation Interface

Figure 8-1 shows the screenshot of the Voice Controller adaptation window which is used to create a new user profile, record the Vocal Joystick sounds to train the system, as well as to test out the adapted profile for further tuning. The main display area shows the Vocal Joystick vowel map. The interface is designed with large buttons and clear text labels so that it can be easily accessed via point-and-click (for those who have some ability to manipulate a pointing device) as well as through voice commands (using the say-what-you-see method as described in Section 2.1.1). For example, to select the upper-left sound, the user can either click on the corresponding wedge-shaped area or utter the speech command "choose bed."

The typical sequence through the adaptation interface for a first-time user is as follows. After opening the adaptation interface by uttering the command "adaptation window" from the speech input mode, the user would create a new user profile through the "change user" dialog. Once a profile is created, all the sounds will initially have a crossed-out microphone icon to indicate that the user has not yet recorded those sounds. The user would proceed to provide a recording of their voice for all sounds by selecting each one and issuing the "record sound" command. After all sounds have been recorded, the user can test them out by issuing the "test all sounds" command,

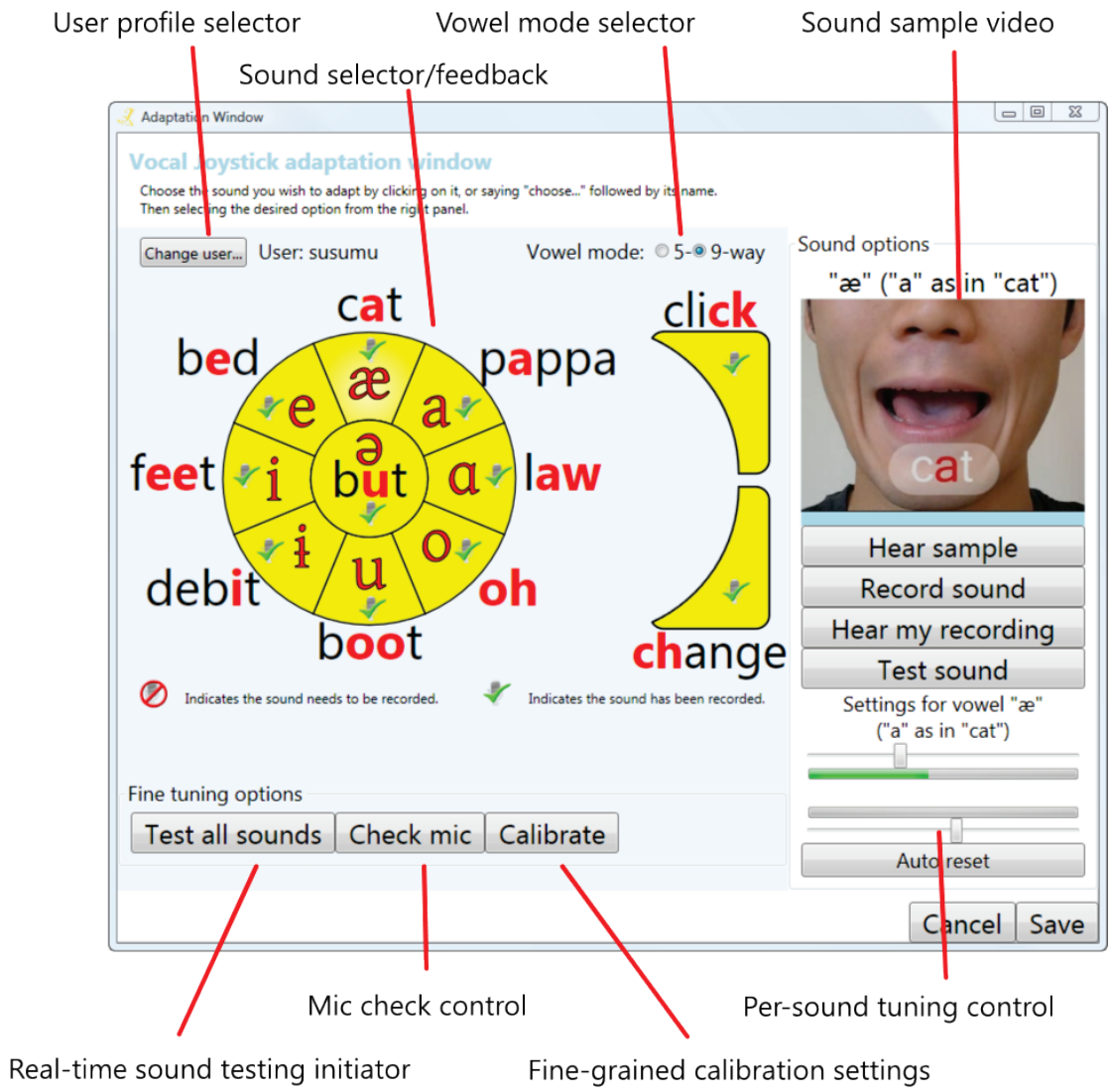


Figure 8-1: Voice Controller adaptation window. The *user profile selector* is used to change as well as create a new user profile. The *vowel mode selector* is used to switch between 5-way or 9-way vowel modes. The *sound selector/feedback* area contains the Vocal Joystick vowel map that enables the user to select a sound to record, as well as provide real-time visual feedback during testing mode (initiated using the *real-time sound testing initiator*). The *sound sample video* shows a sample video of the selected sound being vocalized. The *mic check control* can be used to quickly hear what the microphone is picking up to verify the microphone input volume level. The *per-sound tuning control* enables the quiet and loud volumes for a vowel sound to be manually adjusted. The *fine-grained calibration settings* provide access to advanced settings such as parameters for the pointer speed transfer function which are usually not exposed to users.

or make finer adjustments through the "calibrate" command. The following subsections provide further details of the feedback provided during sound production and the process for tuning the Vocal Joystick response.

8.1.1 Visual and Audio Feedback During Sound Production

One of the first challenges a user faces when starting to learn how to use the Vocal Joystick is learning how to make the vowel sounds that correspond to each direction and the discrete sounds used for controlling modes and issuing other commands. Even with an experienced Vocal Joystick researcher coaching the user on how to make the right sounds, the process can require quite a bit of time. The user may attempt to mimic the sound uttered by the coach while receiving feedback on how to alter the shape of the mouth and the lips. The coach may also offer suggestions on other words which contain the sound similar to the desired sound that the user can evoke as a reference. As it would be impractical to necessitate all potential users of Vocal Joystick systems to spend face-to-face time with a voice coach, a more scalable alternative was needed. To address this issue, I implemented several features in the Voice Controller adaptation interface to provide the user with as much visual and auditory feedback as was feasible.

The first form of visual feedback that is provided is a pre-recorded video of a person uttering a particular sound, focused around the mouth area (top right portion of Figure 8-1). The user can hear and watch the video for any of the sounds by selecting the desired sound and then uttering or clicking the “hear sample” button. The combination of visual and auditory output provided to the user attempts to mimic the experience of watching and hearing the voice coach utter the sound during a face-to-face session. Many of the study participants who have used this feature have commented on its usefulness, especially the ability to review a sound multiple times while mimicking it themselves.

In addition to being able to hear and watch an example of the sound utterance, the adaptation interface also includes the ability for the user to hear their own utterance after they have recorded it through the “hear my recording” command. This immediate feedback in combination with the feedback from the example video enables the user to quickly compare the two and experiment with subtle variations in vocal production until they feel they have gotten close to the target sound. It is also an important feature to be able to tell whether the microphone input gain is set at the appropriate level, as well as checking the level of any background noise that may be picked up by the microphone.

8.1.2 Adjusting and Tuning

While the user is instructed to vocalize each vowel sound at a "normal comfortable volume" during recording, variations in the placement of the microphone as well as differences across users and even vowel sounds can lead to a wide range of average recorded volume levels for each vowel. To enable both fine precision pointer control and quick large-scale movements, it is important that the user be able to vocalize quieter than their normal volume to move slowly and louder to move quickly. I will refer to the lowest and highest volume that the user can comfortably vocalize as *quiet volume* and *loud volume*, respectively, and the normal comfortable volume as *medium volume*. To provide a sense of consistent response across all vowels, the medium volume for each vowel needs to be mapped to the same median pointer speed, and the quiet and loud volumes need to be mapped to the same minimum and maximum pointer speeds, respectively. This allows us to calculate the *normalized volume*, calculated by scaling the raw observed volume value onto a scale ranging from quiet volume to loud volume for the corresponding vowel sound, whose value ranges consistently from 0.0 to 1.0 regardless of the vocalized vowel sound.

When the user records a vowel sound, the system automatically calculates the medium volume by averaging the volume over the two seconds of recorded audio. The quiet and loud volumes are also automatically initialized as being 0.5 and 2.0 times the medium volume, respectively, though they can be adjusted using the per-vowel tuning control in the adaptation window.

Once the user has recorded all the sounds, they can test how well the system has adapted to their voice by issuing the "test all sounds" command. This temporarily switches the adaptation interface to a testing mode, in which the vowel compass display and the discrete sound indicators both provide real-time feedback of what the system is recognizing as the user utters various sounds. To exit from the testing mode, the user simply needs to stay silent for five seconds (until the silence timer bar at the bottom of the screen fills up), at which point the system automatically switches back to the speech-based command mode. During the testing mode, speech command input is disabled to prevent any unintended invocation of spoken commands.

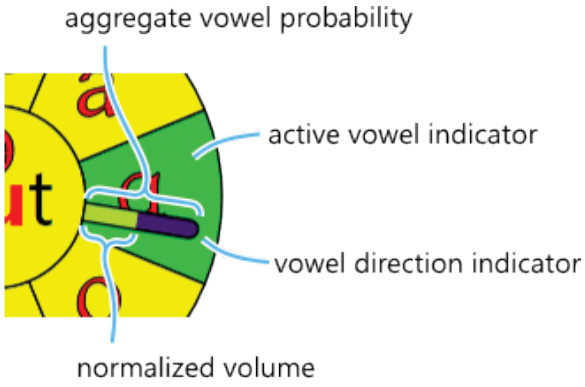
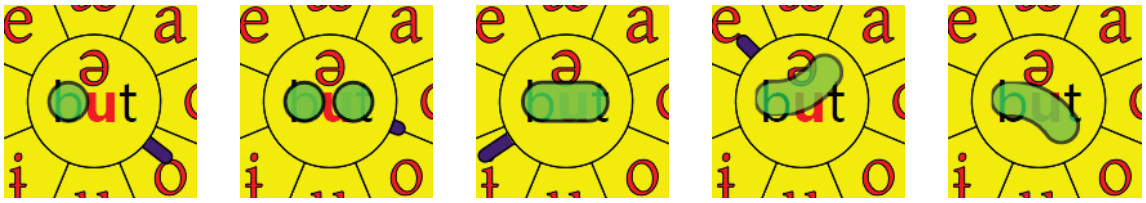


Figure 8-2: Details of the vowel sound feedback provided during the testing mode in the Voice Controller adaptation window.

Figure 8-2 shows the various types of feedback provided during the testing mode. The active-vowel indicators (along with the corresponding discrete-sound indicators) highlight the vowel sound or discrete sound that is currently recognized with the highest probability. The vowel direction indicator provides a more fine-grained indication of the overall vowel direction vector (defined in Section 3.3.2) determined by aggregating the probabilities across all directional vowel sounds. Aside from representing the angle of the vowel direction vector, the vowel direction indicator conveys two additional pieces of information. First, the overall length of the vowel direction indicator represents the magnitude of the aggregated vowel probability. Therefore, if the user vocalizes a sound that is determined by the Vocal Joystick system to have a low probability of being a vowel sound, the overall length of the indicator will be short. Second, the light green portion within the indicator itself represents the observed normalized volume. Each vowel shape also contains an indicator for each type of "voice gesture" as described in Chapter 6 that the system can recognize, such as long vowel, double vowel, and pitch sweeps (Figure 8-3). Using this feedback, the user can practice not only making each individual sound accurately but also transitioning from one to the other smoothly, identifying any particular sound that is not responding as well as the others, and testing out the voice gestures.

8.2 Integration with the Windows Speech Recognizer

To leverage the benefits of speech-driven input as described in Chapter 2, the Voice Controller augments existing speech recognition technology with non-speech vocal input methods. However, several issues need to be considered when integrating these two modalities.



(a) (b) (c) (d) (e)

Figure 8-3: Voice Controller voice gesture visual feedback details shown for the middle vowel sound. The types of vowel gestures represented are (a) single vowel, (b) double vowel, (c) long vowel, (d) rising pitch sweep, and (e) falling pitch sweep.

8.2.1 Disambiguation Challenge

The first issue that needs to be addressed is how to disambiguate between the two modalities. If the system is simultaneously monitoring for speech and non-speech vocal input, real-time determination of the intended input type can be nontrivial and in some cases is not possible. For example, if the user begins to utter the word "okay," during the utterance of the first syllable, the system would not know whether to begin moving the pointer in the lower-right direction (in the case of the Vocal Joystick pointer control) or to wait to process the rest of the utterance as a spoken word. If there is enough contextual information, such as the presence of a button labeled "ok," the system may be able to infer the user's intent to some degree. However, such a technique may not be practical when the context of the interaction is free-form dictation. It may also be possible under certain circumstances to delay the processing of non-speech vocal input until it can be determined that the input is not a speech command, but such a delay can be significantly disruptive when the task requires fluid real-time input.

To mitigate this issue of disambiguating user's intent, the Voice Controller takes the approach of separating the speech input mode and the Vocal Joystick mode as two mutually exclusive modes. Under the speech input mode, the system operates as usual under the WSR engine, recognizing command phrases and dictated phrases depending on the application in focus. Under the Vocal Joystick mode, the WSR engine is turned off, and the Vocal Joystick engine processes non-speech vocal input and manipulates the mouse pointer accordingly. While this separation of modes helps eliminate the disambiguation problem, a second problem arises, which I will refer to as the "escape-sequence dilemma."

8.2.2 The Escape-sequence Dilemma

Whenever multiple operational modes are involved in a system, there needs to be a mechanism for switching between those modes (Norman 1990). In situations where there is only one input

modality available (such may be the case with hands-free voice-driven input), the action associated with the triggering of the mode switch needs to be overloaded within the same modality used for supplying general input to the system. There are many examples of this scenario. One example is in plain-text document formats such as LaTeX, when specifying a command string in the middle of literal text. Another instance is in eye-tracking software when attempting to look at a target without having the system interpret it as the intent to act upon it (the "Midas touch" problem). The scenario also arises in digital pen input systems, when attempting to use a pen gesture to invoke a command while engaging in free-hand drawing. This issue manifests itself even within speech input modes when attempting to utter a command phrase while dictating. In cases where there is a second input modality available that is independent from the primary modality, it would be relatively trivial to use that to indicate when to switch the mode of the primary modality. In certain situations, however, such an option may be impractical or unavailable, and would require the system designer to reserve a certain subset of the single modality's input language to serve as the mode switch trigger, which inevitably decreases the conciseness and the fluidity of the primary input language. I will refer to this tradeoff as the "escape-sequence dilemma."

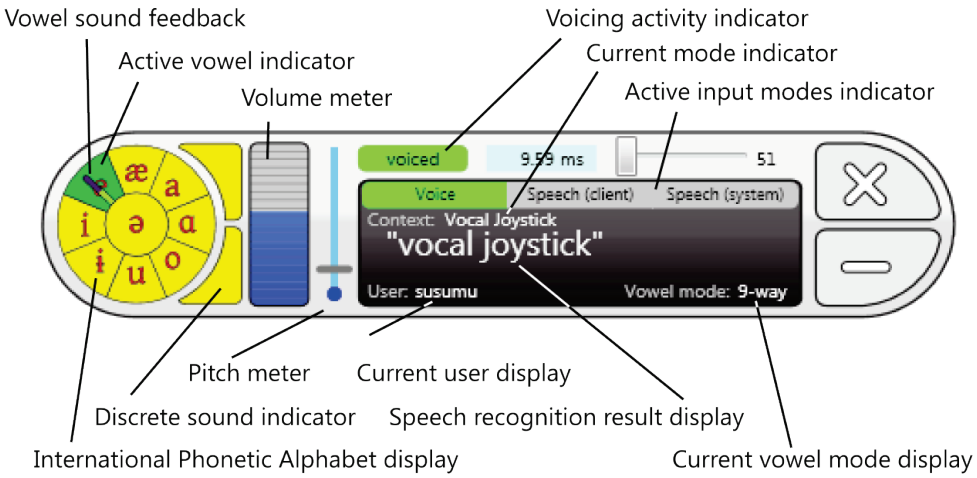
Each of the three examples mentioned in the previous paragraph has established a particular "escape sequence" that enables the user to switch to the alternate modes. In the text entry example, special "escape characters" are reserved to be used as a prefix preceding any non-literal command words, and the escape characters can be escaped themselves to be entered as a literal string. With the eye tracker, a short timeout triggered by staring at a fixed point called "dwell time" is often used as a way to execute an action such as a mouse click, with possibly an on-screen dwell-to-click button also available to toggle the active tracking of the eyes on or off. In digital stylus applications, a bezel button or a button on the stylus itself are typically used to temporarily switch the manner in which the pen gesture is interpreted. In speech recognizers, during dictation, command phrases are interpreted as commands only if the utterance is surrounded by a slight pause, otherwise being interpreted as part of the dictated text. As mentioned in Chapter 2, there have been some research prototypes that have explored ways to eliminate the need for this pause, such as by juxtaposing the pitch of the utterance with the utterance content to distinguish between command and dictation utterances.

The Voice Controller introduces two escape sequences to switch to and from the speech input mode and the Vocal Joystick mode. To switch from the speech input mode to the Vocal Joystick mode, the same mechanism for disambiguating command phrases from dictation phrases is used, namely requiring the user to pause slightly before uttering the command "Vocal Joystick." The phrase can be changed to a shorter phrase if the user desires, such as simply "mouse," as long as it does not conflict with any pre-existing WSR command phrases. To switch from the Vocal Joystick mode back to the speech input mode, I take advantage of the availability of the directionally-neutral middle vowel sound [ə]. Because the utterance of this sound does not lead to any movement of the pointer, it is an ideal candidate for overloading the mode switch functionality. To reduce the chance of accidental mode switches, the mode switch is only triggered if a "long vowel" voice gesture—a sustained vocalization of the sound for longer than half a second by default—is detected for the middle vowel. Once again, this trigger can be replaced by other voice gestures such as "double middle vowel" or "single middle vowel" depending on the user's preference. These two mode switch trigger methods seek to reduce the likelihood of unintended invocations and disruptive pauses or changes in interaction flow as much as possible, while keeping them short and easy to invoke.

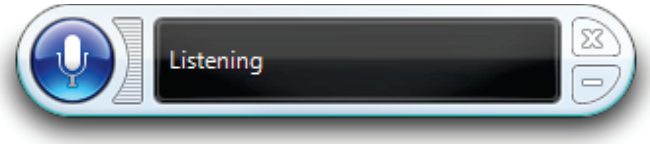
Given the modal nature of the Voice Controller system, it is especially important that the user receives clear feedback about the current mode of the system and any failed or unintended mode transitions. To address this issue, I created a floating status window called the Voice Controller Bar that augments the WSR Bar to provide the user with additional information about the mode that the system is in.

8.3 Voice Controller Bar

The Voice Controller Bar (Figure 8-4a) is an ambient feedback and status indicator akin to the Windows Speech Recognizer Bar (Figure 8-4b). The Voice Controller Bar provides real-time feedback during various modes of operation, with the two primary operating modes being the *speech mode* and the *Vocal Joystick mode*, described in further detail in the next paragraph. The *volume meter* on the Voice Controller Bar presents the same information as the volume meter on the WSR Bar, providing volume feedback when the WSR Bar is disabled. The *pitch meter* provides feedback about the currently recognized pitch level.



(a)



(b)

Figure 8-4: (a) Voice Controller Bar and (b) Windows Speech Recognizer (WSR) Bar. In the Voice Controller Bar, the *vowel sound feedback* shows the same information as the one in the adaptation window (see Section 8.1.2). The *voicing activity indicator* shows whether the current vocalization is voiced, unvoiced, or silence. The *current mode indicator* displays the current mode (speech, Vocal Joystick, or possibly some other Voice Controller client). The *active input modes indicator* shows which of the three types of input—"voice" for non-linguistic voice input, "speech (system)" for WSR's dictation and command input, and "speech (client)" for speech input specific to the current mode—are currently active.

The two main modes of operation—*speech mode* and *Vocal Joystick mode*—are shown in Figure 8-5. During the speech input mode, the user can issue any of the standard WSR commands to control the computer, including dictating text into appropriate controls. The vowel and discrete sound indicators on the Voice Controller Bar remain deactivated during the speech mode, and the recognition-result display area presents any speech commands that are recognized. Whenever the user wishes to start controlling the mouse pointer, the command "Vocal Joystick" switches the Voice Controller Bar to the Vocal Joystick mode. During the Vocal Joystick mode, the WSR Bar is disabled to prevent any unintended recognition results from being generated while the user makes non-linguistic vocalizations. Under this mode, the user can use the Vocal Joystick pointer control method introduced in Section 3.4 to fluidly manipulate the mouse pointer. The vowel and discrete sound indicators on the Voice Controller Bar provide real-time feedback similar to those

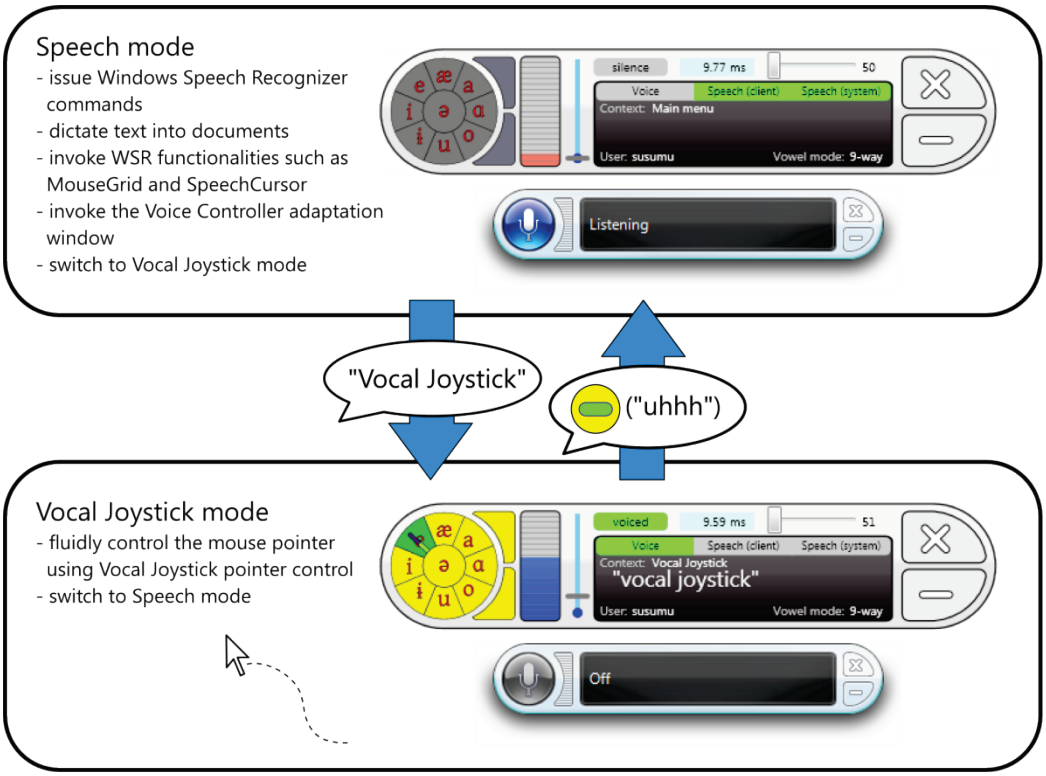


Figure 8-5: Representation of the two operating modes of the Voice Controller—speech mode and Vocal Joystick mode. In the speech mode, all regular Windows Speech Recognizer (WSR) functionalities are available, including dictation and spoken commands, as well as the ability to invoke the adaptation window (by uttering "adaptation window") and switch to the Vocal Joystick mode (by uttering "Vocal Joystick"). In the Vocal Joystick mode, WSR is turned off, and non-linguistic vocalizations are used to control the mouse pointer as described in Section 3.4. To switch back to the speech mode, the user utters a long-vowel voice gesture using the directionally-neutral middle vowel sound ([ə]).

during the testing mode in the adaptation window. If the mouse pointer moves close to the Voice Controller bar during the Vocal Joystick mode, it automatically slides over to the other side of the window to stay out of the way and avoid occluding potential targets. When the user wishes to switch back to the speech mode, a long-vowel voice gesture (see Figure 8-3) using the directionally-neutral central vowel sound [ə] is used.

To ensure that there is always a backup input mechanism available for accessing all the commands in case the user's speech commands are not being recognized, the Voice Controller bar provides a context menu that can be accessed by right clicking on the bar. The same feature can be accessed even during the Vocal Joystick mode—in which the Voice Controller bar moves away from the pointer—through a corresponding icon in the system tray that provides the same context-menu functionality. Such a redundant and multimodal access to functionality is essential

especially in the case of speech-based input, where various environmental factors may temporarily render it ineffective. In addition, each command is also accessible via a global keyboard shortcut. This feature has been especially useful for those users who have already had an alternate input modality such as a physical switch that they wished to use to perform certain actions such as toggling the Vocal Joystick on and off.

8.4 System Architecture

The Voice Controller is built around the managed Vocal Joystick engine API (see Chapter 3), and is written in C#. It also utilizes the System.Speech managed API from Microsoft to integrate with the Windows Speech Recognizer (built into the Windows operating system since Windows Vista), making it compatible with other speech recognizers that conform to the Speech API (SAPI) 5.3 standard. The Voice Controller bar and the adaptation interface are written using Windows Presentation Foundation, and thus require .NET Framework 3.5 to be installed on the system, which it is by default on Windows operating systems since Vista. The Voice Controller can be installed on the user's system using a simple installer, and can be configured to automatically launch itself upon user login alongside the Windows Speech Recognizer.

While the current focus of the Voice Controller is on the integration of the Vocal Joystick pointer control application with the Windows Speech Recognizer, the Voice Controller has been designed using an add-in architecture, enabling other developers to create additional Voice Controller clients that can be integrated into the Voice Controller to provide additional functionality, such as the VoiceGames Controller application introduced in Chapter 6.

8.5 Summary

This chapter presented the Voice Controller, a system that integrates the non-speech vocal input features of the Vocal Joystick engine with the speech-based input offered by the Windows Speech Recognizer. The Voice Controller can run in the background on a user's Windows-based personal computer, providing a quick method for switching between fluid voice-driven pointer control and conventional speech-based dictation and command functionality in any application, whenever needed. With the Voice Controller now available for public download,[1] it is possible for a greater number of people to benefit from the fluid, hands-free voice-driven computer control it offers.

[1] The Voice Controller is available for download from http://www.vocaljoystick.org/download/.

Chapter 9
Future Directions

The creation of the Vocal Joystick engine and its associated libraries has opened up a wide array of possibilities for the design and implementation of novel voice-driven applications. While the general knowledge of the capability of non-speech vocalization as an input modality has increased through the user studies and the development of voice-driven applications, there are a number of exciting next steps that can further extend the body of work presented so far.

In this chapter, I present a number of possible future directions within the context of the three phases of my research approach, which were to:

1. *Understand* the performance characteristics of non-speech vocalization as an input modality;
2. *Build* concrete applications to investigate the effectiveness of non-speech vocalization as an input modality; and
3. *Generalize* reusable building blocks for creating future voice-driven applications.

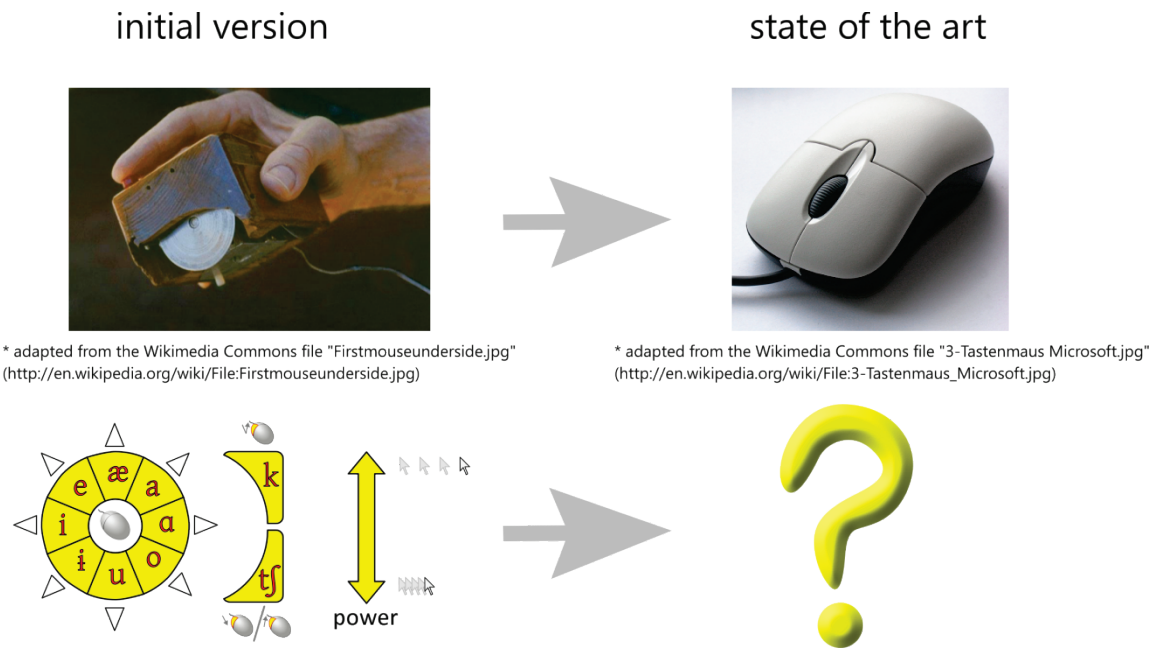


Figure 9-1: An illustration of the evolution of the Vocal Joystick in comparison to the mouse. The current version of the Vocal Joystick is analogous to the first prototype of the computer mouse (upper-left). As the research on voice-driven computer input continues, it is plausible that we will see a similar increase in the performance and capabilities of Vocal Joystick-based control as we have seen for the mouse.

9.1 Further Exploration of the Performance Characteristics of Voice Input

The user studies I have presented in this dissertation seek to serve as the groundwork for future research on understanding the characteristics and capabilities of non-linguistic vocal input. There is a myriad of ways in which future investigations could build upon this work. A number of fundamental experiments still need to be conducted, such as a formal steering task study (Accot & Zhai 1997) involving more experienced Vocal Joystick users, a longitudinal study of longer duration to refine the extrapolated learning curve from my study (Section 4.3), as well as comparative studies with a wider variety of devices including other assistive input devices such as head trackers and eye trackers. An important point to note is that since the Vocal Joystick engine itself is still a relatively new technology and is continually being improved upon, these studies need to be repeated as the Vocal Joystick engine matures. In some sense, the current situation is analogous to when the first desktop mouse was invented by Douglas Engelbart in 1963. The first prototype of the computer mouse was encased in a wooden box with rotatable disks that directly contacted the operating surface as seen in Figure 9-1. Since then, the mouse has evolved drastically, with enhancements such as pointer acceleration that make the mouse feel as

controllable and usable as it is today. The Vocal Joystick will likely follow a similar evolutionary growth as research continues in this area. The following two subsections discusses in greater detail two potential approaches for deriving a better mapping between the user's intent and the resulting effect on the interface.

9.1.1 Mapping Between Vocal Input and Interface Control

To increase the controllability of the various Vocal Joystick driven interactions, including, but also beyond, pointer control, new types of transfer functions between volume as well as pitch and the resulting user interface response need to be explored. The current mapping scheme between the utterance volume and the resulting pointer speed was based on a preliminary evaluation of a few basic transfer function types (Malkin et al. 2005).

There are many key features of a desirable transfer function that were not fully integrated in my early investigations. First, especially for pointer control, the ability to move the pointer at sub-pixel accuracy as well as to traverse the width of the screen comfortably without clutching is crucial for providing a sense of control to the user. This is not a trivial task, and to achieve this effect in common pointing devices such as the mouse and the isometric joystick, a great deal of commercial[1] as well as research effort has been invested (Barrett et al. 1995; Rutledge & Selker 1990). Due to the semantic similarity of the Vocal Joystick control to the manual isometric joystick, the body of work that has investigated the optimal transfer functions for the latter should be a good starting point (Barrett et al. 1995; Casiez & Vogel 2008; Rutledge & Selker 1990). One difference between the Vocal Joystick and the manual isometric joystick is that unlike the manual joystick, in which the user may continue to apply pressure for an extended period of time, the Vocal Joystick user will be limited by his lung capacity in how long he can sustain continuous input. This limitation, especially for users with limited respiratory capacity, will need to be considered in designing a sensible transfer function that does not require excessive "vocal clutching" (i.e., stopping and restarting vocalizations). Towards this end, an inertial response mode may be desirable, in which the pointer or the target of movement can be made to continue moving with some inertia without the user having to accompany it with constant vocalization.

Aside from pointer control, a suitable mapping for continuous input in other task contexts, such as scrolling of text, zooming in and out of documents, and rotation of objects, is needed. Once a

[1] See http://www.microsoft.com/whdc/archive/pointer-bal.mspx for the work on the transfer function used in the Windows operating system.

suitable transfer function is found for pointer control, it may be possible to apply it to these other task contexts by simply scaling the input values accordingly.

I have developed a testing framework (Figure 9-2) for exploring various transfer functions and control mappings to investigate the issues mentioned above. Explorations with the tool up to this point have not yet resulted in a pointer control transfer function that is significantly better than the current setup. One of the challenging aspects of enhancing the controllability of the pointer control method is that unlike a physical joystick, in which the user can maintain a constant pressure (in the case of isometric joysticks) or displacement (in the case of isotonic joysticks) for extended periods with relative ease, the voice-based control is more significantly limited by the finite lung capacity of the user. Based on observations so far, it appears that rather than mapping the volume of the utterance directly to the pointer speed via a transfer function, we may need a more dynamic mechanism, such as modifying the *control-display* (*CD*) *gain*[2] (Casiez et al. 2008;

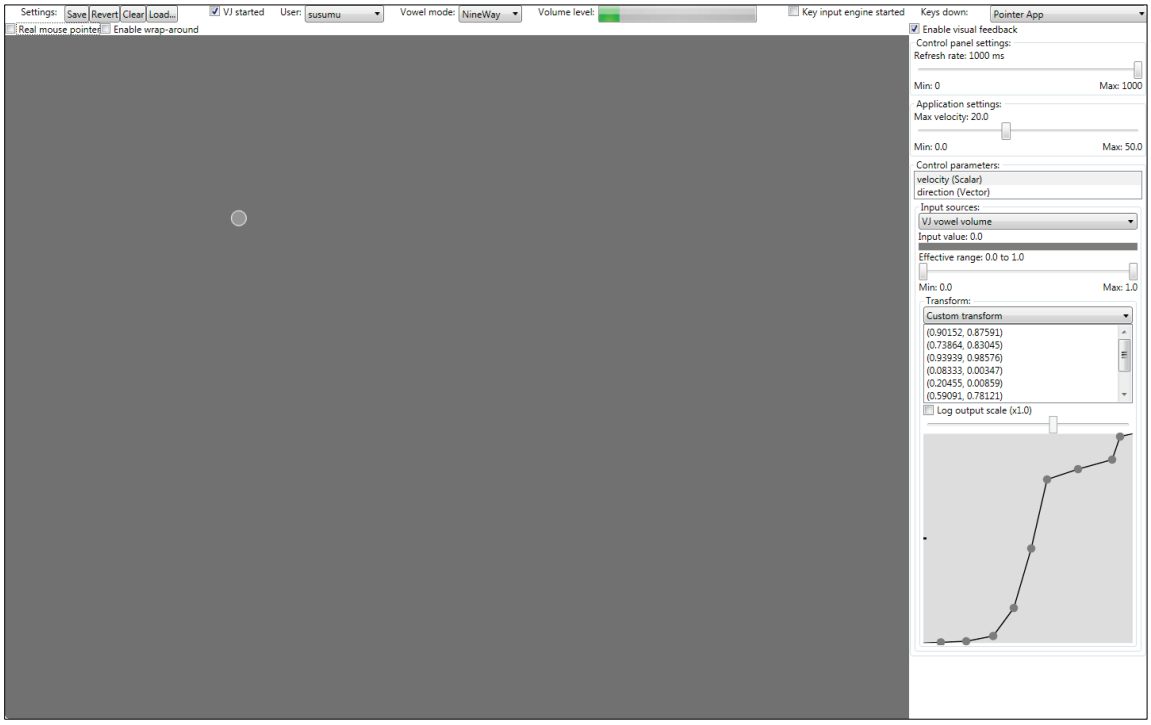


Figure 9-2: Testing framework for exploring various transfer functions and control mappings for voice-driven input.

[2] CD gain is a transformation that is applied to the control signal generated by an input device (e.g., displacement of a mouse) to produce some change in a displayed user interface component (e.g., position of the pointer).

Mackenzie & Riddersma 1994) throughout the duration of a continuous utterance. One possible implementation of this idea will be to set the CD gain to be low at the beginning of the vocalization so that short utterances are more tolerant to variations in volume, hopefully leading to a better sense of control. For longer vocalizations, as the user continues to vocalize, the CD gain can be increased, and movement speed can be adjusted with minor variations in volume.

9.1.2 Better Extraction of User's Intent

The approach described in the previous section for determining the ideal transfer function involves the user tuning the various parameters of the function until it feels "right" or some maximal performance is reached. A different approach may be to attempt to more directly elicit the user's intent in the context of continuous control by means of a controlled feedback-driven response task. In such task, a user would be presented with some visual stimulus that is dynamically changing, such as a volume meter that is slowly oscillating between high and low values. The user is then asked to vocalize the sound that they feel they should make had he been the one who was actually driving the changing feedback himself. The visual stimulus could be extended to more complex widgets such as the pointer moving in two-dimensions at varying speeds. The idea behind this type of task is to collect a series of data that can be used to directly determine the mapping between the user's intent and the resulting feedback by having the system dictate what the user's intent should be. In some sense, it could be considered to be the reverse of the Wizard of Oz study (Dahlbäck et al. 1993) where the user is asked to pretend to be the wizard and synchronize his actions to match what is happening "on the stage." I hypothesize that this approach, in combination with the approach described in the previous section, will enable the mapping between non-linguistic vocal input and interface controls to be significantly improved.

9.2 Further Exploration of Voice-Driven Applications

This section presents future directions in the context of concrete applications, both in terms of enhancements that could be made to an existing application (Voice Controller) as well as new applications (VoiceDrawBot and VoiceWrite) that I have begun to explore.

9.2.1 Enhancing the Voice Controller Application

The Voice Controller has evolved into a robust enough application that it can be deployed and installed on user's computers without the need for any elaborate hardware or software setup. Its availability on the Windows platform and integration with the Windows Speech Recognizer

enables users who already own a Windows Vista machine or higher to take advantage of the program without any additional investment, except for a microphone.

As more users express interest in the Voice Controller, a number of enhancements need to be made to ensure that it offers as intuitive and effective a solution as possible. One area of enhancement is in the visual feedback and the user interface for supporting self-diagnosis of recognition issues. Currently, the adaptation interface in the Voice Controller provides a testing mode for the user to test whether or not the system is responding properly. However, there is still not enough information conveyed to the user that would allow them to troubleshoot instances when the system fails to recognize certain inputs, especially discrete inputs such as the discrete sounds or the voice gestures. On a related topic, a method for automatically coaching the user to improve their vocalization, especially of the vowel sounds, will be beneficial. Currently, the Voice Controller provides video samples of each sound as well as real-time feedback during the testing mode, but there are no explicit suggestions to the user regarding how to change their mouth shape or other features to approach the desired sound. I have also looked into creating a game designed to train the player to make the right sounds while keeping the process engaging.

I have also observed that one of the first obstacles and sources of frustration that a user of voice and speech input modalities faces seems to be when the system generates some recognition event when the user did not intend to vocalize (false positives). This could happen either when the user forgets that the system is processing vocal input and begins speaking or making some sounds, typically when the user's attention is away from the user interface, or when the system picks up some background noise and incorrectly recognizes it as some valid vocalization. In both cases, the user may not realize that the system has processed the false positive events until some time later, e.g., when the user turns his attention back to the interface, at which point the user may become confused about what had happened, and possibly quite frustrated about not knowing what to undo if the exact series of actions that were inadvertently executed is not immediately apparent. What is needed is a way to quickly disable and enable voice input, either explicitly by the user or implicitly by the system. Current speech input methods utilize techniques such as push-to-talk (in which the user presses a physical button during intended vocalizations) and special spoken commands (in which phrases such as "stop listening" and "start listening" can be uttered to toggle the state of the recognizer). While they provide access to the needed functionality, push-to-talk may not be a viable solution for someone who requires hands-free

access, and spoken commands are only available during speech input mode and not during non-linguistic vocal input mode. Some possibilities for future work in addressing this issue include utilizing various external contexts such as the user's head posture (Morency & Darrell 2006) and gaze (Nakano & Ishii 2010) to disable voice input when the user is likely disengaged from the interface, or to utilize techniques such as the SpeechSpotter introduced by Goto and others (2004), which uses a combination of filled-pause detection and high-pitched utterance for distinguishing command utterances from others, to seamlessly switch modes using simple yet robust vocalizations.

Finally, the Voice Controller application should eventually include driver-level support to emulate devices other than the mouse such as the joystick or a digital stylus. In particular, the emulation of a digital stylus is appealing since the various degrees of freedom available in stylus input (such as pen tilt and pressure) can map nicely to the simultaneously modifiable vocal characteristics such as pitch and volume. This would also enable the user to use their voice to interact with their favorite stylus-enabled applications such as drawing and modeling programs, allowing the user to take advantage of the full range of features of these applications.

9.2.2 VoiceDrawBot

As mentioned in Section 3.5.4, the Vocal Joystick library has been used to build an application called VoiceBot that enabled voice-based control of a physical six degrees-of-freedom robotic arm. To further explore the idea of controlling physical robots with voice, I have started building a prototype robot called VoiceDrawBot (Figure 9-3) with two undergraduates from the University of Washington's Computer Science department (Mike Kung and Mike Chung). The core idea behind VoiceDrawBot was to create a physical version of VoiceDraw (Chapter 5). We were fortunate to be able to leverage a robot that one of our colleagues in the same department, Brian Ferris, had already built, called the SpydrBot. The SpydrBot was constructed using the Lego Mindstorms robot kit[3], and was designed to be suspended next to some vertical surface such as a white board by two strings, each of which was anchored to one of the corners of the white board and the other end wound up around one of the motors on the robot. The robot could be made to move along the two-dimensional vertical surface by controlling the two motors and changing the unwound length of each string. The robot also has an actuated pen holder such that it can be made to draw on the surface as it moves. The VoiceDrawBot prototype utilizes the Vocal Joystick

[3] http://mindstorms.lego.com/

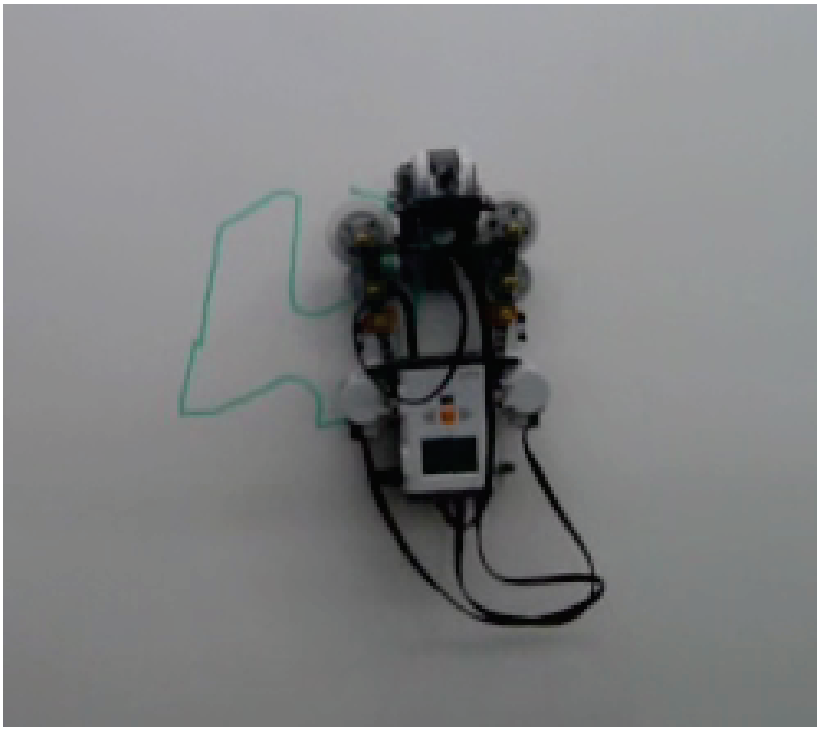


Figure 9-3: A photo of the VoiceDrawBot prototype built from the Lego Mindstorms robot kit drawing a curve with a marker on a white board in response to non-linguistic voice input from a user. The robot is suspended by two strings, whose one end is attached to the respective top corner of the white board and the other end wound around the corresponding motor near the top of the robot.

library to enable the user to make the Vocal Joystick vowel sounds to control the robot's position, just as with the pointer control application. One obvious appeal of the VoiceDrawBot is that it could enable artists such as Mr. Chavez introduced in Chapter 5 to create not just digital artwork but physical ones as well.

9.2.3 VoiceWrite

Another application domain I have started to explore is voice-driven text entry and editing, through the development of an application called VoiceWrite. Although the accuracy of speech-based dictation has been improving significantly over the past decade, there still exist a number of major hurdles that are limiting the realization of its full potential. Feng and others (2006) have found that over a third of the time during a dictation task is spent correcting recognition errors, and the majority of that time is spent on navigating through the documents to the point of correction (Feng et al. 2005). The problem lies in the fact that with current dictation systems, when the user notices an error in the dictation result, the user must first navigate to the point of

error using speech commands, which themselves could be misrecognized and potentially lead to further modification to the text that needs to be corrected as well. I hypothesize that this correction process can be enhanced through the use of non-linguistic vocal input as the means of navigating through the document at various levels of granularity, from individual characters to paragraphs and pages, giving the ability for the user to directly and fluidly control the cursor as well as the document view. The VoiceWrite system is currently being prototyped as a functional text editing environment that supports basic formatting functions such as cutting, copying, pasting, and changing text attributes such as justification and font formatting through the use of non-linguistic vidgets such as voice gestures and Voice Marking Menu for fluid access to the various functionality.

9.3 Expanding the Voice User Interface Building Blocks

Through the development and evaluation of the applications listed in the previous section, a number of common interaction methods and interface controls have emerged. By extracting these common elements and creating a framework for non-speech, voice-driven interaction, more voice-driven applications can be built with reduced effort, leveraging the body of knowledge regarding their performance and efficacy obtained through the performance evaluations presented earlier. Designers of current applications can also be informed about how to make their applications hands-free accessible by retrofitting non-speech vocal interaction on top of them, benefiting the large group of users for whom the use of hand-operated input devices is not practical. A generalized framework can also give us a glimpse into what a user interface could be like if it was to be optimized for hands-free, voice-driven interaction.

From Chapter 3, it can be seen that the Vocal Joystick can provide a rich set of control signals, in some cases, richer in terms of dimensionality than the mouse and the keyboard. However, as we saw from Chapter 4, pointing with the Vocal Joystick is still not as efficient as with the mouse. Due to this fact, when attempting to manipulate user interfaces designed for the mouse and keyboard with voice, we are forced to introduce cumbersome steps in the interaction sequence or to be restricted to the idiosyncrasies of the mouse, namely the fact that it only offers one 2-D control signal (plus an additional 1-D control signal, if the middle wheel is counted). Given that the Vocal Joystick can provide multiple simultaneous 1-D control signals that can be composed in various ways, the natural question is, how could we leverage this capacity to design user interface widgets that are optimized for such input modalities?

There are many promising opportunities in the design of the Voice User Interface and the corresponding voice widgets, or *vidgets*. The design of effective vidgets can be incorporated into new voice-driven applications, and eventually, a new paradigm for an interface designed optimally for voice interaction can emerge. In this section, I present design explorations of what some of those vidgets may be like in a voice user interface (VUI) environment optimized for voice-driven interaction, and the issues that need to be considered at the higher level to enable such vidgets to co-exist in an interface.

9.3.1 Voice Scroll List

Voice Scroll List is a vidget that supports scrolling through a discrete set of items, primarily along a one-dimensional linear representation, similar to the functionality of a standard listbox widget. The key functionality of such a vidget is to enable the user to select an item from a list of items, with support for browsing the entire list in case the user does not know *a priori* which specific item to select. Common scenarios in which such a vidget can be used include browsing through and selecting a photo in a photo album application, scrolling through a list of files in a folder, and scanning through a list of windows on the desktop or tabs in a web browser. While this is currently possible using speech input, it requires the user to either know the exact name of the item that is to be selected, or to use tedious methods for scrolling through the list by repeating commands such as "scroll down four" and "page up." Such a solution is not ideal when the items do not have any textual labels, or when the list contains a large number of items that the user wishes to browse through before deciding which to select.

The Voice Scroll List vidget facilitates browsing through a list by enabling the user to use non-speech vocalization to smoothly scroll through the entire list. When the Voice Scroll List vidget acquires voice focus, the default mode will be the *free-scrolling* mode in which vocalization results in scrolling the list in either of the two vertical (or horizontal, depending on the orientation of the list) directions. For example, in a vertical list of items, vocalizing the downward vowel-sound will reveal items further down on the list, effectively scrolling the entire content of the list upwards. The volume is mapped to the speed of the scrolling.

To reduce the need for prolonged vocalization in case of long lists, the Voice Scroll List also supports "locking" of the scrolling motion through the use of the double-vowel voice gesture. By issuing a double-vowel in a particular direction, the list will begin scrolling in the corresponding

direction at a fixed speed determined by the volume of the double-vowel utterance. With the Voice Scroll List in this *locked-scrolling* mode, the user can control the speed of the scrolling as well as the direction by uttering the relevant directional vowels. For example, once the user initiates a downward locked-scrolling mode, to make the list scroll down even faster, the user simply needs to utter the downward vowel-sound, which will have an additive effect on the velocity based on the duration and the volume of the utterance. If the user wishes to slow down the scrolling, the user simply utters the vowel-sound corresponding to the opposite direction. To switch the scroll direction, the user can either continue to utter the vowel-sound corresponding to the opposite direction until the velocity flips direction, or simply issue a double-vowel gesture in the opposite direction in which case the velocity will automatically flip direction while maintaining its original magnitude (the magnitude of the velocity will undergo a smooth transition over half a second to reduce the jarring effect of suddenly switching direction). To switch out of the locked-scrolling mode back to the free-scrolling mode, the user simply issues a single-vowel gesture with the middle vowel sound.

To facilitate the selection of items, an item at the center of the Voice Scroll List is enclosed in a *focus indicator* and is assumed to be the item in focus. Whenever the vocalization is terminated during the free-scrolling mode, the list continues scrolling until the next closest item from this center is positioned within the focus indicator. Selection and deselection of items is performed by uttering the "ch" discrete sound. In the case of single-select lists such as a list of windows on the desktop to switch among, as soon as the first selection is made, the associated default selection action for that context is performed. While the Voice Scroll List is suited for scrolling a linearly-arranged list, it can also be adapted to two-dimensionally-arranged lists such as the icon-view in file systems, where the icons are arranged in a matrix. In such a case, instead of having the entire content of the list itself scroll around, the focus indicator is manipulated, following the same control semantics as described above, except that its movement direction will directly correspond to the associated direction of the uttered vowel-sounds.

9.3.2 Voice Pan Zoom

The Voice Pan Zoom vidget extends the Voice Scroll List vidget described above into navigation of a continuous two-dimensional space. Common scenarios for such a vidget include exploring maps and navigating across a large design document with large overall structure as well as areas of fine detail. Similar to the Voice Scroll List vidget, the Voice Pan Zoom vidget will also

support *free-panning* and *locked-panning* modes, and enabling panning in any direction. The Voice Pan Zoom vidget also adds the ability to zoom in and out of the view by issuing the rising-pitch or falling-pitch voice gestures, respectively. The difference in pitch relative to the starting pitch of the utterance determines the rate of change in the zoom factor. Therefore, if the user starts an utterance at a medium pitch and transitions to a high pitch and sustains that pitch level, the view will continue to zoom in at that particular rate until the user stops vocalizing. Since the pitch can be modulated concurrently with the vowel sounds, it is possible to change the zoom level while panning. The middle vowel sound can be used to control the zoom without changing the panning position.

9.3.3 Voice Marking Menu

The Voice Marking Menu vidget was introduced in Chapter 5. As described there, the Voice Marking Menu enables navigation of a menu hierarchy using the directional vowel sounds and issuing corresponding commands upon selection of one of the items. This ability to supply an arbitrary number of 0-D control signals while in the non-speech vocalization mode is of great value in addressing the limitations of modes.

As described in Chapter 8, due to the challenge of distinguishing between speech and non-speech vocal input, the VUI Controller took the approach of separating them out into their own modes. Given the extensive use of non-speech vocal input in the vidgets introduced above, they will most likely need to operate under the non-speech vocal-input mode as well to avoid unintended processing by the speech recognition module. Therefore, the current design of the VUI environment assumes that at the highest level there are two operational modes similar to the Voice Controller: *speech mode* and *voice mode*. However, once in the voice mode, we lose access to the rich array of 0-D control signals with which to issue commands and switch modes. While the voice gestures provided an additional set of 0-D control signals, they are quickly used up by vidgets that require several 0-D control signals. In such a situation, the availability of the Voice Marking Menu replenishes the supply of 0-D control signals that can be used, as long as the voice gesture to invoke the Voice Marking Menu itself remains exclusive across the entire voice mode. This is similar to reserving the Windows-key on a PC to always open the start menu in Windows operating systems, regardless of what application currently has focus. To achieve this effect, the long-vowel gesture with the middle vowel sound is reserved as the signal to invoke the Voice Marking Menu anywhere in the voice mode. The content of the Voice Marking Menu, however,

may contain context-specific items depending on what user interface component was being interacted with.

9.4 Summary

While the new Vocal Joystick library makes it possible to quickly prototype and build a number of applications to explore the space of voice-driven interfaces, there needs to be a continued exploration of the potential and efficacy of voice as input through further user studies. While the formative user studies I have conducted provide a starting point for gauging the performance characteristics of voice input, those studies need to continue to be conducted as the Vocal Joystick engine improves. The research could also benefit from a more in-depth investigation into the fundamental cognitive and psychophysiological issues, such as the potential cognitive interference involved in using non-linguistic vocal input in conjunction with other modalities such as manual or eye-tracking input, and understanding the impact of various effects such as the Lombard effect[4] (Lane & Tranel 1971) and the integrality and separability (Jacob et al. 1994) of the tasks and control mappings involved.

In addition to continued user studies, there seems to be three main directions toward which future work in this research area are likely to head. First is in taking existing interfaces that are common across most personal computers which are designed for GUIs operated with keyboard and mouse and now increasingly with touch, and making them accessible via voice by developing adapter interfaces such as the Voice Controller that effectively translate voice input into keyboard and mouse events. The second direction is in deriving new ways to incorporate voice input with other input modalities in the context of existing or novel applications and interfaces. The third thrust is to conceptualize an entirely redesigned interface which considers speech and voice input as first class input modality. This chapter highlighted a number of possible paths that future research could take in each of these directions.

[4] The Lombard effect describes the phenomenon in which speakers involuntarily increase the volume of their voice in an attempt to increase its audibility when the environmental noise level increases.

Chapter 10
Conclusion

Non-speech voice input is a largely untapped interaction modality that holds promising potential both as an enabling technology for people with motor impairments as well as an enhancement for general users who can benefit from multimodal input for their tasks. In this dissertation, I have demonstrated my thesis that:

> *Non-speech vocal input can be used on its own and in conjunction with speech-based input to enable people—especially those with motor disabilities—to control computer interfaces effectively.*

In support of this thesis, I have conducted a number of formative evaluations of non-speech voice input to elicit its performance characteristics (Chapter 4), developed a number of applications to investigate the effectiveness of non-speech vocal input in the context of specific applications (Chapters 5-8), and re-architected the Vocal Joystick library to enable easy reuse of the functional modules in other applications (Chapter 3). The research approach I have taken through this process is as follows:

1. Understand the performance characteristics of non-speech vocalization as an input modality.
2. Build concrete applications to investigate the effectiveness of non-speech vocalization as an input modality.
3. Generalize reusable building blocks for creating future voice-driven applications.

The next sections summarize the contributions I have made through my dissertation work, followed by a set of reflections and insights that were generated during the process.

10.1 Contributions

The specific contributions of the work on this dissertation are presented below, grouped into experimental results, artifacts, and concepts and techniques.

10.1.1 Experimental Results

Throughout the course of my dissertation, a number of empirical results concerning the performance characteristics of non-linguistic vocalization as an input modality were collected. Key results from those user studies are summarized below.

- Voice-controlled pointer movement using the Vocal Joystick was shown to follow Fitts' law ($MT = 606.15 \times ID + 411.63$ milliseconds, $R^2 = 0.986$) through a user study (Section 4.1) involving four Vocal Joystick "experts" (members of the project team). Fitts' index of performance value for the Vocal Joystick pointer control application was determined to be 1.65 bits/sec for this user group, roughly 70% that of manual rate-controlled joysticks and 30% that of conventional desktop mice. These differences are expected to shrink as the Vocal Joystick engine and its pointer speed control mechanism continue to be improved.
- Novice users of the Vocal Joystick pointer control application with minimal training (five minutes) were also able to use it to perform basic target acquisition tasks. In a user study (Section 4.2) involving nine participants who received only five minutes of Vocal Joystick training each, the average target acquisition times for the Vocal Joystick pointer control method was roughly four times faster than the Speech Cursor method (introduced in Section 2.1.4), and comparable to the Mouse Grid (introduced in Section 2.1.3).
- Longitudinal user study (Section 4.3) with five motor-impaired and four non-motor-impaired participants, each spending ten sessions (totaling 11 hours) using the Vocal

Joystick, revealed that by the tenth session, users without motor impairments were able to reach the previously measured "expert" target-acquisition performance and those with motor impairments on average reached 70% of the expert level. The users were also able to fully memorize the 8-way vowel-to-direction mapping by the fifth session on average.

- Categorical input user study (Section 4.4) involving two motor-impaired and six non-motor-impaired participants revealed that in a 4-directional choice reaction task, the use of directional Vocal Joystick vowel sounds to emulate corresponding key input could lead to a total input time roughly twice as fast as that of spoken directional-word input (750 milliseconds versus 1,350 milliseconds, respectively, for the non-motor-impaired participants after accounting for the likely reduction in cognitive processing time for the Vocal Joystick modality). For the two participants with motor impairments, the total input time using the Vocal Joystick vowels was 40% to 45% faster than speech input and was either comparable to or only 25% slower than using the keyboard arrow keys.

10.1.2 Artifacts

A number of artifacts—mostly software but also a few hardware—have been produced as a result of my dissertation research, both directly by me as well as by others leveraging the reusable Vocal Joystick library I have built.

- I worked with a self-described "electronic voice painter"—a digital artist with spinal cord injury who had been creating artwork on his computer—to design and develop an application called VoiceDraw (Chapter 5), a hands-free voice-driven drawing program that enables fluid and realistic stroke creation using just one's voice and speech.
- I created VoiceGame Controller (Chapter 6), a voice-driven system that makes it possible to play a wide variety of computer games hands-free by mapping speech input as well as non-linguistic vocal input onto a set of emulated keyboard and mouse input. The system builds upon findings from the categorical input user study (Section 4.4), which showed that non-linguistic vocal input can significantly outperform speech input in choice reaction tasks. As a result, VoiceGame Controller provides enhanced playability of games over speech-based control methods, and in certain cases where fluid continuous input is required, enables games that previously were unplayable using speech-based control methods to be played by voice.

- I developed VoicePen (Chapter 7), a prototype application for enabling multimodal interaction by augmenting digital stylus input with non-linguistic vocal input. VoicePen offered a number of interaction techniques for simultaneously manipulating multiple continuous parameters using both the digital stylus and non-linguistic vocalizations, such as brush stroke path, stroke thickness, stroke opacity, object position, and object orientation.
- Throughout the course of my dissertation work, I continued to improve upon the original Vocal Joystick pointer control application by incorporating design enhancements based on observations and feedback from users and study participants, resulting in a completely revamped application called the Voice Controller (Chapter 8). The Voice Controller offers a variety of visual feedback for aiding the user in learning and tuning the Vocal Joystick vowel vocalizations, and seamlessly integrates with the Windows Speech Recognizer to provide access to dictation and spoken command functionalities in addition to the Vocal Joystick pointer control.
- To facilitate the development of voice-driven applications, I re-architected the original Vocal Joystick engine into a modularized, cross-platform, reusable library (Section 3.3). This Vocal Joystick library has been the foundation upon which all of my applications described above, as well as additional voice-driven applications created by others, have been built. I have also created a managed-code .NET wrapper library on top of the native Vocal Joystick library. This wrapper library not only makes the Vocal Joystick engine functionality easily accessible to programming languages such as C# that support rapid user interface development, but also contains a number of reusable, generalized voice-driven components, such as voice gesture detectors and Voice Marking Menu.
- There have been a number of applications built by others (Section 3.5) using the Vocal Joystick library. They include a mobile sensor data annotation tool called VoiceLabel, a game for learning the Vocal Joystick vowel direction mapping called VoiceVoiceRevolution, another game that helps one practice steering with the Vocal Joystick called VoiceRacer, and a system for controlling a physical five-degrees-of-freedom robotic arm called VoiceBot.

10.1.3 Concepts and Techniques

The development of the artifacts listed above helped to situate the study of non-linguistic vocalization as an input modality, resulting in a number of novel concepts and techniques as summarized below.

- I have demonstrated how the Vocal Joystick "vowel space" can be transformed beyond its original two-dimensional and radial mapping to accommodate a wide variety of contexts. For example, the brush stroke manipulation method in VoiceDraw can be considered to be a three-dimensional mapping, where the vowel sounds are used to move the brush along the two-dimensional canvas and the volume is used to control the stroke thickness. A pair of vowel sounds can also be used for a simple one dimensional slider-like control, as was demonstrated in the brightness slider in VoiceDraw's color picker, the continuous undo feature also in VoiceDraw, as well as the parameter manipulation method in VoicePen.
- The use of the Vocal Joystick vowel sounds as triggers for categorical input was investigated in a user study (Section 4.4) and then applied to the VoiceGame Controller application (Chapter 6) for emulating rapid and precisely-timed key press events. Such use of non-linguistic vocalization has made it possible to issue multiple categorical inputs more rapidly and at more precise timings than with spoken commands.
- The concept of *voice gestures* was also introduced (Section 6.4.1), in which various patterns of vocalization duration and pitch changes are interpreted as categorical input. When combined with the directionally-neutral middle vowel sound, these voice gestures can be used to issue a number of categorical inputs while in Vocal Joystick mode, during which speech input is most likely disabled to avoid unintentional processing of vowel sounds in spoken words (Section 8.2.2).
- The Voice Marking Menu (Section 5.4.3) provides a hierarchical menu arranged in a radial manner that can be navigated using the directional Vocal Joystick vowel sounds. This concept greatly expands the number of categorical input that can be specified while in the Vocal Joystick mode without having to switch to the speech input mode, and leverages the directional vowel mapping common to many Vocal Joystick-based applications.

10.2 Reflections and Insights

My numerous interactions with user study participants and experience with building various voice-driven applications have generated some reflections and insights, which I share in this section.

10.2.1 Challenges Encountered in Working with Participants with Motor Impairments

There were a number of challenges in conducting user studies involving participants with various motor impairments. The first challenge was the fact that the term "motor impairment" encompasses a much wider range of physical functions and abilities than other types of physical impairments, such as visual impairment or hearing impairment. Even when we seek out a specific subset of those with motor impairments who have difficulty manipulating standard pointing devices, there is a wide range of abilities, including those who cannot use the mouse but can grossly manipulate the trackball, those who can use the mouse but have sporadic spasms, etc. This observation suggests that in practical context, there will likely not be a single solution or input technique that can suitably accommodate a large majority of the users in this population. Instead, each user may benefit most by being able to customize and possibly combine multiple devices that they feel are best suited for them, such as in the case of Mr. Chavez introduced in Chapter 5 who preferred to use a combination of speech input and a trackball. As such, I have architected the Vocal Joystick library in a reusable and extensible manner that allows others to create custom extensions to it, as well as integrate its functionality with other input devices.

It also seemed that there were fewer groups and organizations representing people with motor impairments, compared to those representing people with visual or hearing impairments, and those that did exist were not as expansive and regionally dispersed as well. As a result, the number of people I could recruit locally was significantly limited. As mentioned in Chapter 8, this was the primary reason why I felt it was essential that I make the Vocal Joystick application work in real-world settings and become more widely available, so that they can be remotely deployed anywhere in the world, greatly increasing the pool of potential user study participants. Obviously, such an approach comes with its own challenges of remotely administering controlled experiments. One possible technical solution to this would be to create a highly-instrumented testbed application with a user-guiding wizard interface. A non-technical solution would be to collaborate with researchers and organizations at various localities to have them personally administer such experiments.

The wide range of participant's abilities also meant that various accommodations needed to be considered for each individual. For example, I visited certain participants at their homes to conduct some of the studies due to the difficulty they would face in travelling to the user study lab. The user study lab also needed to be adjusted to support variable-height displays and desks to accommodate those using a wheelchair. In one instance, the participant had an electronic ventilator, which she preferred to turn off during the study but which would then periodically generate a loud beep as a reminder that it is off, requiring the experimenter to temporarily attend to it before resuming with the session. As these examples suggest, the variety of specific accommodations both before and during the user study sessions made it challenging to gather highly controlled quantitative measures. As such, the value of qualitative observational data could not be underestimated, and a number of my studies focused more heavily on observational data rather than quantitative measures (for example, the study of VoiceDraw presented in Chapter 5).

10.2.2 General Challenges Observed Across All Participants

There were also a number of general challenges and insights that extended beyond the population of users with motor impairments. For instance, another difficulty in recruiting participants arose when I needed "proficient" Vocal Joystick users (e.g., in the expert user study presented in Section 4.1). Due to the fact that the Vocal Joystick input modality is unlike any existing input modality and that it requires on the order of hours to become proficient with the vowel mappings (see results from the longitudinal study presented in Section 4.3), it was difficult to recruit a large number of participants who were able or willing to come in to the user study lab for many hours at a time. As was mentioned in the previous section in the context of the difficulty of recruiting participants with motor impairments, this was one of the primary motivations behind the effort to make the Vocal Joystick application work in real-world settings and be more widely available.

Another observation related to the above issue of a completely novel input modality was that during the learning process when the users were getting acquainted with the Vocal Joystick vowel mappings, a number of them made statements about their preconceived notions or desires about the vowel sound arrangements (Section 4.3). For example, a number of people suggested the use of sounds that are similar to those contained in corresponding directional word, such as [i] for "right" and [e] for "left." There was also an interesting case of a user from Korea who told me that the Vocal Joystick vowel mapping was "incorrect." When asked about his statement, he described that in the Korean language, each cardinal direction has an associated sound based on

the underlying *jamo*, the glyph units that comprise the Korean featural alphabet system called *Hangul*,[1] and that the Vocal Joystick mapping (particularly the cardinal vowel sounds) did not correspond to those sounds. These examples point to the importance of making such novel input modalities be as non-arbitrary and as close to the concepts and experiences the users are used to as possible, while maintaining the ability for the system to effectively process the input in a meaningful way. A successful example of this approach can be seen in the case of the EdgeWrite (Wobbrock et al. 2003), a unistroke text entry method in which novel stroke gestures were designed for each letter of the English alphabet for use on PDA's and other devices. While the concept of defining a character as a sequence of corner acquisitions was entirely novel, the stroke gestures were designed to map as closely as possible to what most people naturally tended to do when given the opportunity to represent each character within this corner space on their own. I feel that future explorations of the use of non-linguistic vocal input can benefit from this type of design approach, especially for concepts such as voice gestures (Section 6.4.1).

In addition, especially with the early versions of the Vocal Joystick application that lacked much visual feedback, it seemed that the users had difficulty knowing how to effectively modify their vocalization when the system wasn't responding very well. For example, under the original Vocal Joystick pointer control application, when a user vocalized a particular vowel sound corresponding to some desired movement direction and the pointer did not move as they expected, the most common reaction by the user was to repeat the vocalization at a higher volume, even though the underlying issue may not be volume but the vowel quality. This reaction is akin to the Lombard effect (Lane & Tranel 1971), in which speakers involuntarily increase the volume of their voice when the environmental noise level increases in an attempt to increase its audibility. A similar reaction has been observed with discrete sounds as well. In both cases, there is a mismatch between the user's understanding of the cause of the problem and the actual underlying cause, a mismatch known as the *gulf of evaluation* (Norman 1990). The new Voice Controller (Chapter 8) attempts to address this issue by providing a real-time visual feedback to convey to the user appropriate information that they need to be able to make effective adjustments, such as the vowel probability and volume display. As mentioned in Chapter 9, further enhancements still remain such as providing effective feedback for discrete sounds.

[1] In the Korean alphabet system called Hangul, the underlying glyph unit corresponding to the up direction is associated with the sound [o], down with [u], left with [ʌ], and right with [a]. See http://en.wikipedia.org/wiki/Hangul for an overview of Hangul and the sounds associated with the glyphs.

10.2.3 Other Observations and Insights

In addition to the challenges observed for the various user populations, I present a few additional observations and insights below.

One of the favorable observations made was that even though the vowel sounds and the discrete sounds used in the Vocal Joystick are arbitrary from the perspective of a novice user, the fact that most Vocal Joystick-based applications provide immediate feedback and continuous fluid control seemed to help users learn the controls in an enjoyable and exploratory manner. This was most evident during the demonstration of the VoiceDraw application at a public exhibit (Section 5.5.4), when almost one hundred children of all ages were able to quickly pick up on the directional vowel control and produce a number of creative drawings. A similar observation was made during the study of VoicePen (Chapter 7), in which a number of participants commented on the enjoyability of the interaction techniques and some continued to play with them after the session was over. This points to the importance of the element of enjoyability in getting users, especially novices, engaged and willing to experiment on their own.

On a related note, throughout the study of VoiceGames (Chapter 6), it appeared that the participants were more eager to partake in the game tasks compared to participants partaking in other non-game user study tasks such as the Fitts' target acquisition task (Section 4.2). This is not surprising given the entertaining nature of games, but it suggests that such property could be leveraged to get users to play games that not only provide them with entertainment value, but also lead to the acquisition of various skills relevant to non-linguistic vocal input, such as the vowel mappings and volume control. This concept was the driving motivation behind the two third-party applications developed using the Vocal Joystick library: VoiceVoiceRevolution (Section 3.5.2) and VoiceRacer (Section 3.5.3).

Finally, one of the most rewarding moments during my dissertation work was when I saw study participants and users, especially those with motor impairments, express their excitement and exhilaration when they were able to accomplish something using the Vocal Joystick which they could not do well or at all before using their alternate input devices, such was the case with Mr. Chavez (Chapter 5).

10.3 Final Remarks

The area of non-speech voice-driven input still remains largely unexplored. However, as my investigations have shown, it holds great promise for enabling fluid hands-free computer interaction, especially for users who have limited motor abilities. Christopher Reeve, who has had spinal cord injury for almost 10 years before his passing in 2004, once stated in an online interview[2] about the value of communication technology and voice-driven computer control for people with disabilities:

> *"[The Internet is] an essential tool. And, literally, a lifeline for many disabled people. I have Dragon Dictate. And while I was in rehab, I learned to operate it by voice. And I have enjoyed corresponding with friends and strangers with that system. Many disabled people have to spend long hours alone. Voice-activated computers are a means of communication that can prevent a sense of isolation."*

The Vocal Joystick engine may be the key technology that could bring increased flexibility and expressivity to voice-based interaction to enable even greater hands-free access to a wider range of application domains. The public availability of the Voice Controller application as well as the Vocal Joystick library[3] opens up the possibility of creating many novel applications that leverage the great capacity of human voice for both users with and without disabilities.

[2] http://www.webaim.org/articles/motor/

[3] The previous version of the Voice Controller without the real-time feedback has been downloaded over 700 times as of March 19, 2010. The Voice Controller application and the Vocal Joystick library can be downloaded from the Vocal Joystick project webpage at http://www.vocaljoystick.org.

BIBLIOGRAPHY

Accot, J., & Zhai, S. (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 295-302). Atlanta, Georgia, United States: ACM. doi: 10.1145/258549.258760.

Accot, J., & Zhai, S. (2003). Refining Fitts' law models for bivariate pointing. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 193-200). Ft. Lauderdale, Florida, USA: ACM. doi: 10.1145/642611.642646.

Balakrishnan, R., & Hinckley, K. (1999). The role of kinesthetic reference frames in two-handed input performance. In *Proceedings of the 12th annual ACM symposium on user interface software and technology* (pp. 171-178). Asheville, North Carolina, United States: ACM. doi: 10.1145/320719.322599.

Baron, A., & Journey, J. W. (1989). Age differences in manual versus vocal reaction times: further evidence. *Journal of Gerontology*, *44*(5), 157-159.

Barrett, R. C., Selker, E. J., Rutledge, J. D., & Olyha, R. S. (1995). Negative inertia: a dynamic pointing function. In *Conference companion on human factors in computing systems* (pp. 316-317). Denver, Colorado, United States: ACM. doi: 10.1145/223355.223692.

Bier, E. A., Stone, M. C., Pier, K., Buxton, W., & DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on computer graphics and interactive techniques* (pp. 73-80). Anaheim, CA: ACM. doi: 10.1145/166117.166126.

Bilmes, J. A., Malkin, J., Xiao, L., Harada, S., Kilanski, K., Kirchhoff, K., et al. (2006). The Vocal Joystick. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing* (pp. 625-628). Presented at the ICASSP 2006, Toulouse, France. doi: 10.1109/ICASSP.2006.1660098.

Bohan, M., Thompson, S., Scarlett, D., & Chaparro, A. (2003). Gain and target size effects on cursor-positioning time with a mouse. In *Proceedings of the human factors and ergonomics society 47th annual meeting* (pp. 737-740).

Butler, C. G., & Amant, R. S. (2004). HabilisDraw DT: a bimanual tool-based direct manipulation drawing environment. In *CHI '04 extended abstracts on human factors in computing systems* (pp. 1301-1304). Vienna, Austria: ACM. doi: 10.1145/985921.986049.

Buxton, W. (1983). Lexical and pragmatic considerations of input structures. *ACM SIGGRAPH Computer Graphics*, *17*(1), 31-37. doi: 10.1145/988584.988586.

Buxton, W., & Myers, B. A. (1986). A study in two-handed input. *ACM SIGCHI Bulletin*, *17*(4), 321-326. doi: 10.1145/22339.22390.

Card, S. K., English, W. K., & Burr, B. J. (1987). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT. In *Human-computer interaction: a multidisciplinary approach* (pp. 386-392). Morgan Kaufmann Publishers Inc.

Card, S. K., Mackinlay, J. D., & Robertson, G. G. (1991). A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, *9*(2), 99-122. doi: 10.1145/123078.128726.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey, USA: Laurence Erlbaum Associates Inc.

Casiez, G., & Vogel, D. (2008). The effect of spring stiffness and control gain with an elastic rate control pointing device. In *Proceedings of the 26th annual SIGCHI conference on human factors in computing systems* (pp. 1709-1718). Florence, Italy: ACM. doi: 10.1145/1357054.1357321.

Casiez, G., Vogel, D., Balakrishnan, R., & Cockburn, A. (2008). The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, *23*(3), 215-250. doi: 10.1080/07370020802278163.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., et al. (1997). QuickSet: multimodal interaction for distributed applications. In *Proceedings of the 5th ACM international conference on multimedia* (pp. 31-40). Seattle, Washington, United States: ACM. doi: 10.1145/266180.266328.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience.

Crawford, C. (1984). *The Art of Computer Game Design*. Berkeley, California, USA: Osborne/McGraw-Hill.

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of Oz studies: why and how. In *Proceedings of the 1st international conference on intelligent user interfaces* (pp. 193-200). Orlando, Florida, United States: ACM. doi: 10.1145/169891.169968.

Dai, L., Goldman, R., Sears, A., & Lozier, J. (2004). Speech-based cursor control: a study of grid-based solutions. In *Proceedings of the 6th international ACM SIGACCESS conference on computers and accessibility* (pp. 94-101). Presented at the ASSETS 2004, Atlanta, Georgia, USA. doi: http://doi.acm.org/10.1145/1028630.1028648.

Davis, R. C., Colwell, B., & Landay, J. A. (2008). K-sketch: a 'kinetic' sketch pad for novice animators. In *Proceeding of the 26th annual SIGCHI conference on human factors in computing systems* (pp. 413-422). Florence, Italy: ACM. doi: 10.1145/1357054.1357122.

Davis, R. C., Saponas, T. S., Shilman, M., & Landay, J. A. (2007). SketchWizard: Wizard of Oz prototyping of pen-based user interfaces. In *Proceedings of the 20th annual ACM symposium on user interface software and technology* (pp. 119-128). Newport, Rhode Island, USA: ACM. doi: 10.1145/1294211.1294233.

Davis, S. B., & Mermelstein, P. (1990). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition* (pp. 65-74). San Francisco, California, USA: Morgan Kaufmann Publishers Inc.

Deng, L., & Huang, X. (2004). Challenges in adopting speech recognition. *Communications of the ACM*, *47*(1), 69-75. doi: 10.1145/962081.962108.

Dragicevic, P., & Fekete, J. (2004). The Input Configurator toolkit: towards high input adaptability in interactive applications. In *Proceedings of the working conference on advanced visual interfaces* (pp. 244-247). Gallipoli, Italy: ACM. doi: 10.1145/989863.989904.

Epps, B. W. (1986). Comparison of six cursor control devices based on Fitts' law models. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, *30*(4), 327-331.

Feng, J., Karat, C., & Sears, A. (2005). How productivity improves in hands-free continuous dictation tasks: lessons learned from a longitudinal study. *Interacting with Computers*, *17*(3), 265-289. doi: 10.1016/j.intcom.2004.06.013.

Feng, J., Sears, A., & Karat, C. (2006). A longitudinal evaluation of hands-free speech-based navigation during dictation. *International Journal of Human-Computer Studies*, *64*(6), 553-569.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*, 381-391.

Fitzmaurice, G. W., Ishii, H., & Buxton, W. A. S. (1995). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 442-449). Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co. doi: 10.1145/223904.223964.

Foley, J. D., Wallace, V. L., & Chan, P. (1984). The human factors of computer graphics interaction techniques. *IEEE Computer Graphics and Applications*, *4*(11), 13-48.

Fuller, R. B. (2000). *Nine Chains to the Moon*. Doubleday.

Gibbs, M., Wadley, G., & Benda, P. (2006). Proximity-based chat in a first person shooter: using a novel voice communication system for online play. In *Proceedings of the 3rd Australasian conference on interactive entertainment* (pp. 96-102). Perth, Australia: Murdoch University.

Goto, M., & Hayamizu, S. (1999). A real-time music scene description system: detecting melody and bass lines in audio signals. In *Speech Communication* (Vol. 43, pp. 31-40). doi: 10.1.1.33.1085.

Goto, M., Itou, K., & Hayamizu, S. (2002). Speech Completion: on-demand completion assistance using filled pauses for speech input interfaces. In *Proceedings of the 7th international conference on spoken language processing* (pp. 1489-1492). Presented at the ICSLP 2002, Denver, Colorado, USA.

Goto, M., Itou, K., & Hayamizu, S. (1999). A real-time filled pause detection system for spontaneous speech recognition. In *Proceedings of the European conference on speech communication and technology* (pp. 227-230). doi: 10.1.1.33.866.

Goto, M., Kitayama, K., Itou, K., & Kobayashi, T. (2004). Speech Spotter: on-demand speech recognition in human-human conversation on the telephone or in face-to-face situations. In *Proceedings of the 8th international conference on spoken language processing* (pp. 1533-1536). Presented at the INTERSPEECH 2004, Jeju Island, Korea.

Gross, M. D., & Do, E. Y. (1996). Ambiguous intentions: a paper-like interface for creative design. In *Proceedings of the 9th annual ACM symposium on user interface software and technology* (pp. 183-192). Seattle, Washington, USA: ACM. doi: 10.1145/237091.237119.

Grossman, T., & Balakrishnan, R. (2005). A probabilistic approach to modeling two-dimensional pointing. *ACM Transactions on Computer-Human Interaction*, TOCHI, *12*(3), 435-459. doi: 10.1145/1096737.1096741.

Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: the kinematic chain as a model. *Journal of Motor Behavior*, *19*(4), 486-517.

Harada, S., Landay, J. A., Malkin, J., Li, X., & Bilmes, J. A. (2006). The Vocal Joystick: evaluation of voice-based cursor control techniques. In *Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility* (pp. 197-204). Portland, Oregon, USA: ACM. doi: 10.1145/1168987.1169021.

Harada, S., Landay, J. A., Malkin, J., Li, X., & Bilmes, J. A. (2008a). The Vocal Joystick: evaluation of voice-based cursor control techniques for assistive technology. *Disability and Rehabilitation: Assistive Technology*, *3*(1), 22. doi: 10.1080/17483100701352963.

Harada, S., Lester, J., Patel, K., Saponas, T. S., Fogarty, J., Landay, J. A., et al. (2008b). VoiceLabel: using speech to label mobile sensor data. In *Proceedings of the 10th international conference on multimodal interfaces* (pp. 69-76). Chania, Crete, Greece: ACM. doi: 10.1145/1452392.1452407.

Harada, S., Saponas, T. S., & Landay, J. A. (2007a). VoicePen: augmenting pen input with simultaneous non-linguistic vocalization. In *Proceedings of the 9th international conference on multimodal interfaces* (pp. 178-185). Nagoya, Aichi, Japan: ACM. doi: 10.1145/1322192.1322225.

Harada, S., Wobbrock, J. O., & Landay, J. A. (2007b). VoiceDraw: a hands-free voice-driven drawing application for people with motor impairments. In *Proceedings of the 9th international ACM SIGACCESS conference on computers and accessibility* (pp. 27-34). Tempe, Arizona, USA: ACM. doi: 10.1145/1296843.1296850.

Harada, S., Wobbrock, J. O., Malkin, J., Bilmes, J. A., & Landay, J. A. (2009). Longitudinal study of people learning to use continuous voice-based cursor control. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 347-356). Presented at the CHI 2009, Boston, Massachusetts, USA: ACM. doi: 10.1145/1518701.1518757.

Hinckley, K., Czerwinski, M., & Sinclair, M. (1998). Interaction and modeling techniques for desktop two-handed input. In *Proceedings of the 11th annual ACM symposium on user interface software and technology* (pp. 49-58). San Francisco, California, USA: ACM. doi: 10.1145/288392.288572.

Hinckley, K., Pausch, R., Proffitt, D., Patten, J., & Kassell, N. (1997). Cooperative bimanual action. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 27-34). Atlanta, Georgia, USA: ACM. doi: 10.1145/258549.258571.

Hornof, A. J., & Cavender, A. (2005). EyeDraw: enabling children with severe motor impairments to draw with their eyes. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 161-170). New York, New York, USA: ACM Press. doi: http://doi.acm.org/10.1145/1054972.1054995.

House, B., Malkin, J., & Bilmes, J. A. (2009). The VoiceBot: a voice controlled robot arm. In *Proceedings of the 27th annual SIGCHI conference on human factors in computing systems* (pp. 183-192). Boston, Massachusetts, USA: ACM.

Igarashi, T., & Hinckley, K. (2000). Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the 13th annual ACM symposium on user interface software and technology* (pp. 139-148). San Diego, California, USA: ACM. doi: 10.1145/354401.354435.

Igarashi, T., & Hughes, J. F. (2001). Voice as sound: using non-verbal voice input for interactive control. In *Proceedings of the 14th annual ACM symposium on user interface software and technology* (pp. 155-156). Orlando, Florida, USA: ACM. doi: 10.1145/502348.502372.

Izdebski, K. (1980). Effects of prestimulus interval on phonation initiation reaction times. *Journal of Speech and Hearing Research*, *23*(3), 485-489.

Izdebski, K., & Shipp, T. (1978). Minimal reaction times for phonatory initiation. *Journal of Speech and Hearing Research*, *21*(4), 638-651.

Jacob, R. J. K. (1990). What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 11-18). New York, New York, USA: ACM Press. doi: http://doi.acm.org/10.1145/97243.97246.

Jacob, R. J. K., Sibert, L. E., McFarlane, D. C., & Mullen Jr., M. P. (1994). Integrality and separability of input devices. *ACM Transactions on Computer-Human Interaction*, TOCHI, *1*(1), 3-26. doi: 10.1145/174630.174631.

Jul, S., & Furnas, G. W. (1998). Critical zones in desert fog: aids to multiscale navigation. In *Proceedings of the 11th annual ACM symposium on user interface software and technology* (pp. 97-106). San Francisco, California, USA: ACM. doi: 10.1145/288392.288578.

Julia, L., & Faure, C. (1995). Pattern recognition and beautification for a pen based interface. In *Proceedings of the 3rd international conference on document analysis and recognition* (Vol. 1, p. 58). Washington, DC, USA: IEEE Computer Society.

Kabbash, P., Buxton, W., & Sellen, A. (1994). Two-handed input in a compound task. In *Conference companion on human factors in computing systems* (p. 230). Boston, Massachusetts, USA: ACM. doi: 10.1145/259963.260425.

Kahn, J. G., Lease, M., Charniak, E., Johnson, M., & Ostendorf, M. (2005). Effective use of prosody in parsing conversational speech. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 233-240). Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Kahn, J. G., Ostendorf, M., & Chelba, C. (2004). Parsing conversational speech using enhanced segmentation. In *Proceedings of human language technology conference / North American chapter of the Association for Computational Linguistics annual meeting* (pp. 125-128). Presented at the HLT-NAACL, Boston, Massachusetts, USA.

Kamel, H. M., & Landay, J. A. (2002). Sketching images eyes-free: a grid-based dynamic drawing tool for the blind. In *Proceedings of the 5th international ACM conference on assistive technologies* (pp. 33-40). Presented at the ASSETS 2002, Edinburgh, Scotland: ACM Press. doi: http://doi.acm.org/10.1145/638249.638258.

Karat, C., Vergo, J., & Nahamoo, D. (2003). Conversational interface technologies. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (pp. 169-186). Hillsdale, New Jersey, USA: Laurence Erlbaum Associates Inc.

Karimullah, A. S., & Sears, A. (2002). Speech-based cursor control. In *Proceedings of the 5th international ACM conference on assistive technologies* (pp. 178-185). Edinburgh, Scotland: ACM. doi: 10.1145/638249.638282.

Kencana, A. P., & Heng, J. (2007). Experiment on a novel user input for computer interface utilizing tongue input for severely disabled. In *Proceedings of the 1st international convention on rehabilitation engineering & assistive technology: in conjunction with 1st Tan Tock Seng Hospital neurorehabilitation meeting* (pp. 114-117). Singapore: ACM. doi: 10.1145/1328491.1328522.

Kent, R. D. (2000). Research on speech motor control and its disorders: A review and prospective. *Journal of Communication Disorders*, *33*(5), 391-428. doi: 10.1016/S0021-9924(00)00023-X.

Koester, H. H. (2003). Abandonment of speech recognition by new users. In *Rehabilitation Engineering and Assistive Technology Society of North America*. Presented at the RESNA 2003, Atlanta, Georgia, United States.

Kurtenbach, G., Fitzmaurice, G., Baudel, T., & Buxton, B. (1997). The design of a GUI paradigm based on tablets, two-hands, and transparency. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 35-42). Atlanta, Georgia, USA: ACM. doi: 10.1145/258549.258574.

Lane, H., & Tranel, B. (1971). The Lombard sign and the role of hearing in speech. *Journal of Speech and Hearing Research*, *14*(4), 677-709.

Leganchuk, A., Zhai, S., & Buxton, W. (1998). Manual and cognitive benefits of two-handed input: an experimental study. *ACM Transactions on Computer-Human Interaction*, TOCHI, *5*(4), 326-359. doi: 10.1145/300520.300522.

Levin, G., & Lieberman, Z. (2004). In-situ speech visualization in real-time interactive installation and performance. In *Proceedings of the 3rd international symposium on non-photorealistic animation and rendering* (pp. 7-14). New York, New York, USA: ACM Press. doi: http://doi.acm.org/10.1145/987657.987659.

Lipscomb, J. S., & Pique, M. E. (1993). Analog input device physical characteristics. *ACM SIGCHI Bulletin*, *25*(3), 40-45. doi: 10.1145/155786.155794.

Liu, Y., Shriberg, E., Stolcke, A., Hillard, D., Ostendorf, M., Peskin, B., et al. (2004). The ICSI-SRI-UW metadata extraction system. In *Proceedings of the 8th international conference on spoken language processing* (pp. 577-580). Presented at the ICSLP 2004, Jeju Island, Korea. doi: 10.1.1.64.8587.

Mace, R. L., Hardie, G. J., & Place, J. P. (1991). Accessible environments: toward universal design. In W. F. E. Preiser, J. C. Vischer, & E. T. White (Eds.), *Design Intervention: Toward a More Humane Architecture*. New York: Van Nostrand Reinhold.

MacKenzie, C. L., Marteniuk, R. G., Dugas, C., Liske, D., & Eickmeier, B. (1987). Three-dimensional movement trajectories in Fitts' task: Implications for control. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, *39*(4), 629. doi: 10.1080/14640748708401806.

MacKenzie, I. S. (1995). Input devices and interaction techniques for advanced computing. In *Virtual environments and advanced interface design* (pp. 437-470). Oxford University Press, Inc.

MacKenzie, I. S., & Buxton, W. (1992). Extending Fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 219-226). Monterey, California, USA: ACM. doi: 10.1145/142750.142794.

Mackenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, *7*, 91-139.

Mackenzie, I. S., & Riddersma, S. (1994). Effects of output display and control-display gain on human performance in interactive systems. *Behaviour & Information Technology*, *13*(5), 328. doi: 10.1080/01449299408914613.

Mackinlay, J., Card, S. K., & Robertson, G. G. (1990). A semantic analysis of the design space of input devices. *Human-Computer Interaction*, *5*(2), 145-190.

Malkewitz, R. (1998). Head pointing and speech control as a hands-free interface to desktop computing. In *Proceedings of the 3rd international ACM conference on assistive technologies* (pp. 182-188). Marina del Rey, California, USA: ACM. doi: 10.1145/274497.274531.

Malkin, J., & Bilmes, J. (2009). Multi-layer ratio semi-definite classifiers. In *IEEE international conference on acoustics, speech, and signal processing* (pp. 4465-4468). Los Alamitos, California, USA: IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/ICASSP.2009.4960621.

Malkin, J., Li, X., & Bilmes, J. A. (2005). Energy and loudness for speed control in the Vocal Joystick. In *IEEE Automatic Speech Recognition and Understanding Workshop*.

Malkin, J., Li, X., Harada, S., Landay, J., & Bilmes, J. (2010). The Vocal Joystick Engine v1.0. *Computer Speech & Language*, *in submission*.

Manaris, B., & Harkreader, A. (1998). SUITEKeys: a speech understanding interface for the motor-control challenged. In *Proceedings of the 3rd international ACM conference on assistive technologies* (pp. 108-115). Marina del Rey, California, USA: ACM. doi: 10.1145/274497.274517.

Manaris, B., McCauley, R., & MacGyvers, V. (2001). An intelligent interface for keyboard and mouse control - providing full access to PC functionality via speech. In *Proceedings of the 14th international Florida artificial intelligence research society conference* (pp. 182-188). AAAI Press.

de Mauro, C., Gori, M., Maggini, M., & Martinelli, E. (2001). *Easy access to graphical interfaces by Voice Mouse*. Technical report, available from the author at: maggini@dii.unisi.it, Università di Siena.

Mihara, Y., Shibayama, E., & Takahashi, S. (2005). The migratory cursor: accurate speech-based cursor movement by moving multiple ghost cursors using non-verbal vocalizations. In *Proceedings of the 7th international ACM SIGACCESS conference on computers and accessibility* (pp. 76-83). Baltimore, Maryland, USA: ACM. doi: 10.1145/1090785.1090801.

Miniotas, D. (2000). Application of Fitts' law to eye gaze interaction. In *CHI '00 extended abstracts on human factors in computing systems* (pp. 339-340). The Hague, The Netherlands: ACM. doi: 10.1145/633292.633496.

Morency, L., & Darrell, T. (2006). Head gesture recognition in intelligent interfaces: the role of context in improving recognition. In *Proceedings of the 11th international conference on intelligent user interfaces* (pp. 32-38). Sydney, Australia: ACM. doi: 10.1145/1111449.1111464.

Murphy, K., Corfield, D. R., Guz, A., Fink, G. R., Wise, R. J. S., Harrison, J., et al. (1997). Cerebral areas associated with motor control of speech in humans. *Journal of Applied Physiology*, *83*(5), 1438-1447.

Myers, B. A. (1990). A new model for handling input. *ACM Transactions on Information Systems*, TOIS, *8*(3), 289-320. doi: 10.1145/98188.98204.

Nakano, Y. I., & Ishii, R. (2010). Estimating user's engagement from eye-gaze behaviors in human-agent conversations. In *Proceeding of the 14th international conference on intelligent user interfaces* (pp. 139-148). Hong Kong, China: ACM. doi: 10.1145/1719970.1719990.

Nebes, R. D. (1978). Vocal versus manual response as a determinant of age difference in simple reaction time. *Journal of Gerontology*, *33*(6), 884-889.

Newman, M. W., Lin, J., Hong, J. I., & Landay, J. A. (2003). DENIM: an informal web site design tool inspired by observations of practice. *Human-Computer Interaction*, *18*(3), 259-324.

Norman, D. (1990). *The Design of Everyday Things*. Doubleday Business.

Nwe, T. L., & Li, H. (2007). Singing voice detection using perceptually-motivated features. In *Proceedings of the 15th international conference on multimedia* (pp. 309-312). Augsburg, Germany: ACM. doi: 10.1145/1291233.1291299.

Oller, D. K. (2000). *The emergence of the speech capacity*. Lawrence Erlbaum Associates, Inc.

Olwal, A., & Feiner, S. (2005). Interaction techniques using prosodic features of speech and audio localization. In *Proceedings of the 10th international conference on intelligent user interfaces* (pp. 284-286). San Diego, California, USA: ACM. doi: 10.1145/1040830.1040900.

Oviatt, S. (2003). Multimodal interfaces. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (pp. 286-304). Laurence Erlbaum Associates Inc.

Oviatt, S., Cohen, P., Wu, L., Vergo, J., Duncan, L., Suhm, B., et al. (2000). Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Human-Computer Interaction*, *15*(4), 263-322.

Owen, R., Kurtenbach, G., Fitzmaurice, G., Baudel, T., & Buxton, B. (2005). When it gets more difficult, use both hands: exploring bimanual curve manipulation. In *Proceedings of graphics interface* (pp. 17-24). Victoria, British Columbia: Canadian Human-Computer Communications Society.

Patel, S. N., & Abowd, G. D. (2007). Blui: low-cost localized blowable user interfaces. In *Proceedings of the 20th annual ACM symposium on user interface software and technology* (pp. 217-220). Newport, Rhode Island, USA: ACM. doi: 10.1145/1294211.1294250.

Penfield, W., & Rasmussen, T. (1952). *The cerebral cortex of man: a clinical study of localization of function*. Macmillan.

Radwin, R. G., Vanderheiden, G. C., & Lin, M. (1990). A method for evaluating head-controlled computer input devices using Fitts law. *Human Factors*, *32*(4), 423-438.

Raisamo, R. (1999). An alternative way of drawing. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 175-182). Pittsburgh, Pennsylvania, USA: ACM. doi: 10.1145/302979.303035.

Ramos, G., & Balakrishnan, R. (2003). Fluid interaction techniques for the control and annotation of digital video. In *Proceedings of the 16th annual ACM symposium on user interface software and technology* (pp. 105-114). Vancouver, Canada: ACM. doi: 10.1145/964696.964708.

Ramos, G., & Balakrishnan, R. (2005). Zliding: fluid zooming and sliding for high precision parameter manipulation. In *Proceedings of the 18th annual ACM symposium on user interface software and technology* (pp. 143-152). Seattle, Washington, USA: ACM. doi: 10.1145/1095034.1095059.

Ramos, G., Boulos, M., & Balakrishnan, R. (2004). Pressure widgets. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 487-494). Vienna, Austria: ACM. doi: 10.1145/985692.985754.

Ramos, G. A., & Balakrishnan, R. (2007). Pressure marks. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 1375-1384). San Jose, California, USA: ACM. doi: 10.1145/1240624.1240834.

Riemer-Reiss, M. L., & Wacker, R. R. (2000). Factors associated with assistive technology discontinuance among individuals with disabilities. *Journal of Rehabilitation*, *66*(3), 44-50.

Rudnicky, A. I., Lunati, J., & Franz, A. M. (1991). Spoken language recognition in an office management domain. In *Proceedings of the acoustics, speech, and signal processing* (pp. 829-832). Presented at the ICASSP 1991, IEEE Computer Society.

Rutledge, J. D., & Selker, E. J. (1990). Force-to-motion functions for pointing. In *Proceedings of the IFIP TC13 3rd international conference on human-computer interaction* (pp. 701-706). Amsterdam, The Netherlands: North-Holland Publishing Co.

Salem, C., & Zhai, S. (1997). An isometric tongue pointing device. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 538-539). Atlanta, Georgia, USA: ACM. doi: 10.1145/258549.259021.

Schmandt, C. (1993). From desktop audio to mobile access: opportunities for voice in computing. In H. R. Hartson & D. Hix (Eds.), *Advances in Human-Computer Interaction* (Vol. 4, pp. 251-283). Norwood, NJ: Ablex Publishing Co.

Schmandt, C., Ackerman, M. S., & Hindus, D. (1990a). Augmenting a window system with speech input. *Computer*, *23*(8), 50-56.

Schmandt, C., Hindus, D., Ackerman, M. S., & Manandhar, S. (1990b). Observations on using speech input for window navigation. In *Proceedings of the IFIP TC13 3rd international conference on human-computer interaction* (pp. 787-793). Amsterdam, The Netherlands: North-Holland Publishing Co.

Sears, A., & Young, M. (2003). Physical disabilities and computing technologies: an analysis of impairments. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (pp. 482-503). Lawrence Erlbaum Associates, Inc.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana, Illinois: Urbana University of Illinois Press.

Shipp, T., Izdebski, K., & Morrissey, P. (1984). Physiologic stages of vocal reaction time. *Journal of Speech and Hearing Research*, *27*(2), 173-178.

Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages. *Computer*, *16*(8), 57-69.

Shneiderman, B. (2000). The limits of speech recognition. *Communications of the ACM*, *43*(9), 63-65. doi: 10.1145/348941.348990.

Soukoreff, R. W., & MacKenzie, I. S. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *International Journal on Human-Computer Studies*, *61*(6), 751-789.

Sporka, A. J., Kurniawan, S. H., Mahmud, M., & Slavík, P. (2006). Non-speech input and speech recognition for real-time control of computer games. In *Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility* (pp. 213-220). Portland, Oregon, USA: ACM. doi: 10.1145/1168987.1169023.

Sporka, A. J., Kurniawan, S. H., & Slavik, P. (2004). Whistling User Interface (U3I). In *User-Centered Interaction Paradigms for Universal Access in the Information Society* (pp. 472-478).

Starkweather, C. W., Franklin, S., & Smigo, T. M. (1984). Vocal and finger reaction times in stutterers and nonstutterers: differences and correlations. *Journal of Speech and Hearing Research*, *27*(2), 193-196.

Stevens, S. S. (1957). On the psychophysical law. *Psychological Review*, *64*(3), 153-181.

Tapia, M. A., & Kurtenbach, G. (1995). Some design refinements and principles on the appearance and behavior of marking menus. In *Proceedings of the 8th annual ACM symposium on User interface and software technology* (pp. 189-195). Pittsburgh, Pennsylvania, United States: ACM. doi: 10.1145/215585.215973.

Teichner, W. H. (1954). Recent studies of simple reaction time. *Psychological Bulletin*, *51*(2), 128-149. doi: doi:10.1037/h0060900.

Till, J., Goldsmith, H., & Reich, A. (1981). Laryngeal and manual reaction times of stuttering and nonstuttering adults. *Journal of Speech and Hearing Research*, *24*(2), 192-196.

Tse, E., Greenberg, S., Shen, C., & Forlines, C. (2007). Multimodal multiplayer tabletop gaming. *Computers in Entertainment (CIE)*, *5*(2), 12. doi: 10.1145/1279540.1279552.

Venables, P. H., & O'connor, N. (1959). Reaction times to auditory and visual stimulation in schizophrenic and normal subjects. *Quarterly Journal of Experimental Psychology*, *11*(3), 175. doi: 10.1080/17470215908416307.

Wadley, G., Gibbs, M. R., & Benda, P. (2005). Towards a framework for designing speech-based player interaction in multiplayer online games. In *Proceedings of the 2nd Australasian conference on interactive entertainment* (pp. 223-226). Sydney, Australia: Creativity & Cognition Studios Press.

Warren, B. (1997). Change and necessity: creative activity, well-being and the quality of life for persons with a disability. In *Quality of Life for People with Disabilities: Models, Research and Practice* (pp. 270-291). Nelson Thornes.

Wobbrock, J. O., Myers, B. A., & Kembel, J. A. (2003). EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th annual ACM symposium on user interface software and technology* (pp. 61-70). Vancouver, Canada: ACM. doi: 10.1145/964696.964703.

Zhai, S., Kong, J., & Ren, X. (2004). Speed-accuracy tradeoff in Fitts' law tasks: on the equivalency of actual and nominal pointing precision. *International Journal of Human-Computer Studies*, *61*(6), 823-856. doi: http://dx.doi.org/10.1016/j.ijhcs.2004.09.007.

Zhu, X., Guan, C., Wu, J., Cheng, Y., & Wang, Y. (2007). Expectation-maximization method for EEG-based continuous cursor control. *EURASIP Journal on Applied Signal Processing*, *2007*(1), 26-26.

VITA

Susumu Harada received his Bachelor of Science degree in Computer Science and Human computer Interaction from Carnegie Mellon University in 2000. After working at a consulting firm in Austin, Texas for two years, he went on to receive his Master of Science degree in Computer Science from Stanford University in 2004. From 2004, he attended the University of Washington, where he was advised by James Landay and Jacob Wobbrock, and in 2010 graduated with a Doctor of Philosophy in Computer Science.