Voice Games: Investigation Into the Use of Non-speech Voice Input for Making Computer Games More Accessible

Susumu Harada[1,*], Jacob O. Wobbrock[2], and James A. Landay[3]

[1] IBM Research – Tokyo, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken, 242-8502, Japan
[2] The Information School, DUB Group, University of Washington, Seattle, Washington, 98195, USA
[3] Computer Science and Engineering, DUB Group, University of Washington, Seattle, Washington, 98195, USA
haradas@jp.ibm.com, wobbrock@u.washington.edu, landay@cs.washington.edu

Abstract. We conducted a quantitative experiment to determine the performance characteristics of non-speech vocalization for discrete input generation in comparison to existing speech and keyboard input methods. The results from the study validated our hypothesis that non-speech voice input can offer significantly faster discrete input compared to a speech-based input method by as much as 50%. Based on this and other promising results from the study, we built a prototype system called the *Voice Game Controller* that augments traditional speech-based input methods with non-speech voice input methods to make computer games originally designed for the keyboard and mouse playable using voice only. Our preliminary evaluation of the prototype indicates that the Voice Game Controller greatly expands the scope of computer games that can be played hands-free using just voice, to include games that were difficult or impractical to play using previous speech-based methods.

Keywords: Computer games, accessible games, speech recognition, non-speech vocalization.

1 Introduction

Computer games today offer much more than just entertainment. They can provide social as well as educational value, and new genres of games such as "games with a purpose" (GWAP) [1] are emerging that combine gaming with productive output. However, almost all of these games are designed to be played using some form of manual input device, typically a keyboard for generating discrete input and a pointing device such as a mouse for generating continuous input. Consequently, people with

[*] The entirety of the work presented in this paper was conducted while the first author was affiliated with the Computer Science and Engineering department, DUB Group, University of Washington, Seattle, Washington, 98195, USA.

P. Campos et al. (Eds.): INTERACT 2011, Part I, LNCS 6946, pp. 11–29, 2011.

various forms of motor impairments who find it difficult or impossible to use manual input devices are excluded from reaping the benefits of such games. Is there a way to make these games accessible to people with motor impairments?

There are a number of compelling reasons to explore how to make computer games more accessible. First, hands-free access to computer games can benefit a wide range of people, including nearly 300,000 children under the age of 18 in the U.S. alone (or 1 in 250 children) who have been diagnosed with arthritis or other rheumatologic conditions [2]. Also, for people with more severe forms of motor impairment, computer games and game-like social applications (such as SecondLife [3]) may be one of the few viable avenues for entertainment and social interaction, both of which play an important role in improving quality of life [4]. Second, research into effective hands-free game input can also spur future designs of novel multimodal games. Third, an increased understanding of how to effectively control computer games hands-free can also inform how to better design general user interfaces for interaction using non-manual input modalities.

Strengths and Limitations of Speech Input for Game Control. Speech input is one of the hands-free input modalities that holds potential for addressing the above challenges. Unlike eye trackers or head trackers, a speech recognition system does not require elaborate or expensive hardware, and recent operating systems—such as versions of Microsoft Windows since Vista—include speech recognition software for free. Furthermore, the human language enables virtually a limitless number of words and phrases to be uttered, each of which can be mapped to a discrete command to be executed on a computer. This can be beneficial for controlling a number of computer games that require numerous keystroke combinations to be executed. Also, the ability to utter natural language commands compared to the need to remember and recall arbitrary keystroke combinations can be especially beneficial for novice users and may lead to a boost in performance [5].

However, speech input also has a number of inherent limitations that are preventing it from being an effective hands-free game control modality. First, 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. Related to this is the limitation on the maximum number of utterances that can be recognized within a short period of time. When the need arises for dynamically issuing multiple input events at a rapid and varying rate, the per-utterance delay imposes a limit on how many such utterances can be recognized within the short time span.

Another major limitation of speech-based input is its inability to specify smoothly varying input. Output from a speech recognizer is discrete in nature, as the output is not generated until a word or a phrase has been recognized. 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.

The current limitations of speech-based game control methods can be summarized in the context of the space of input signals expected by computer games. As mentioned earlier, most existing computer games are designed to be played using devices that can provide some combination of discrete input signals—such as a

keyboard—and continuous input signals—such as a mouse or a joystick. However, they vary widely in the degree of temporal and spatial fidelity demanded from each type of input signals. For discrete input signals, they can range from simply requiring execution of a few non-time-critical commands, to demanding precise and rapid execution of multiple and possibly simultaneous commands. For continuous input signals, they can range from simply requiring non-time-critical selection of a point, to demanding fast and fluid steering. "Strategy games" such as card games and puzzle games typically do not require fast or precisely-timed inputs in either of these two dimensions. On the other hand, "skill-and-action" games such as most arcade-based games often require both discrete and continuous input signals to be executed rapidly and fluidly. Existing speech-based game control methods are currently limited to offering hands-free input signals within the subset of this input signal space defined by the lower ends of the discrete and continuous input signal dimensions.

Potential of Non-speech Voice Input for Game Control. Fortunately, our previous work and work by others have shown that such limitations of traditional speech-based input methods, especially their inability to provide fluid continuous input, can be addressed effectively by augmenting them with the use of non-speech voice input [6-10]. In particular, the Vocal Joystick engine [11] can continuously track non-speech vocal properties including volume, pitch, and vowel quality to enable fluid interface manipulations such as smooth mouse pointer control. A longitudinal study involving participants with and without motor impairments found that people can learn the directional vowel mapping after a few hours of training, and even approach the performance of physical joysticks for pointing tasks [12]. The Vocal Joystick engine has also been successfully integrated into applications such as a hands-free drawing program called VoiceDraw [7], demonstrating its expressivity and controllability in the context of continuous steering tasks. In a sense, these systems fill a part of the void left by current speech-based methods in the computer game input signal space mentioned above, namely the far end along the continuous input dimension.

We felt that the expressivity of the Vocal Joystick engine could also be harnessed to expand voice modality's coverage of the game input signal space along the *discrete* input signal dimension as well, by enabling faster, more responsive discrete input than speech-based methods. We hypothesized that by using the Vocal Joystick engine to generate discrete instead of continuous input events in response to directional vowel vocalizations, we could achieve significantly faster input speeds as compared to using spoken commands. Such a method would also nicely complement the continuous input mode already offered by the Vocal Joystick engine, as the same directional vowel sound mapping can be used, maintaining consistency and likely enhancing learnability. The combination of these two methods, along with traditional speech-based input methods, could greatly expand the coverage of hands-free voice-based computer game input methods.

Towards this end, we conducted a quantitative experiment to determine the performance characteristics of non-speech vocalization for discrete input generation in contrast to existing speech and keyboard input methods using an application we built called the *Vocal D-Pad*. While it was expected that non-speech voice input will be faster than a speech-based input method, the results from the study showed that non-speech voice input is significantly faster, with indications that it may further approach

the performance of keyboard input with additional training. Based on these promising results from the study, we built a prototype system called the *Voice Game Controller*, which combines the rapid discrete input capability of the Vocal D-Pad, the fluid continuous input capability of the original Vocal Joystick application, and the natural language input capability of traditional speech-based input. Our preliminary evaluation of the prototype indicates that the Voice Game Controller greatly expands the scope of computer games that can be played hands-free using just voice, to include games that were difficult or impractical to play using previous speech-based methods.

2 Related Work

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

2.1 Current Speech-Based Game Controls

Speech input in games has primarily been used for communication among multiple players [13, 14] or for command-and-control-style applications [5]. 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* [15], *Voice Buddy* [16], and *VR Commander* [17] 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 time, often mapped to obscure key combinations. Other general "speech macro" tools such as *Vocola* [18] and *WSR Macros* [19] offer similar functionality, but do not include any game-specific functionality. Most of these tools act as keystroke generators and are often used in conjunction with a physical keyboard or a mouse as an augmentative input modality. One limitation of these tools is that they cannot be used on their own to play games that require continuous input from pointing devices such as the mouse or joystick.

Aside from using speech for dictation or command-and-control, today's commercial speech recognizers also enable the execution of pointing tasks using speech. Features such as the *MouseGrid* and *SpeechCursor* [20] offer the ability to control the mouse pointer to a limited extent. Although these features do enable basic pointing and clicking tasks to be performed using speech, they take significantly longer than the mouse [20], and their discrete nature makes them unsuitable for any task that requires rapid pointing or smooth steering such as games.

2.2 Games 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* [21] and *Rock Band* [22]

track the pitch of the players' singing and generate a score based on how close they match the target pitch track. *PAH!* [23] and *shout n dodge* [24] 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* [25] 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. Igarashi [8] presents a set of simple games that demonstrate the use of pitch changes and sound detection to control simple movements of the game character. His games involve making rising or falling pitch sounds to move a character up or down, varying the duration of the vocalization to control how far a character moves, and making plosive sounds to shoot. In *Humming Tetris*, Sporka *et al.* [26] map various pitch inflection "gestures" to various controls such as move left, move right, and rotate.

While these games were designed from the onset specifically to be controlled by voice, and while we cannot expect many of the existing or future computer games to be redesigned in such a way, they provide insight into the types of non-speech vocal properties that can be used to map to existing game inputs.

2.3 The Vocal Joystick Engine

As mentioned in the introduction, one of the promising developments in the area of voice-based input has been the Vocal Joystick engine [11], which not only tracks the continuous vocal parameters such as volume and pitch, but also vowel quality. The Vocal Joystick engine samples audio input from the microphone every ten milliseconds, and for every frame, if the incoming sound is a vowel sound, reports the recognition probability for each of the nine vowel classes shown in Figure 1. These probabilities are then aggregated based on the radial arrangement of the vowel classes to yield an overall continuous directional vector, which can be used to steer a mouse pointer. The Vocal Joystick engine library [11] provides an API through which

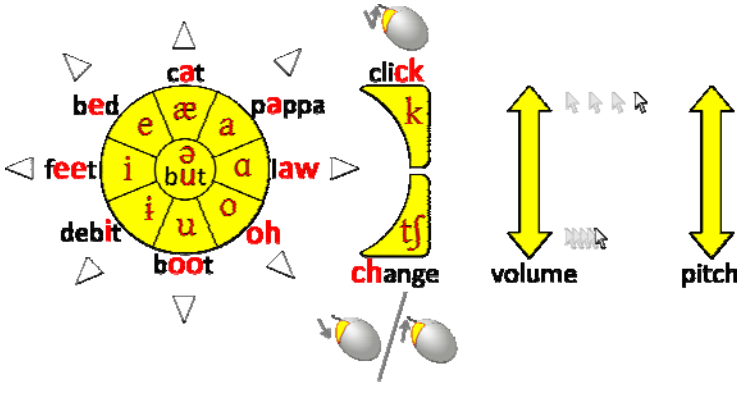

Fig. 1. The *Vocal Joystick sound map* showing the vocal features tracked by the Vocal Joystick engine. The *vowel compass* (left) shows the vowel sounds mapped to each radial direction. Red letters in each word approximate the corresponding sound. *Discrete sounds* such as "ck" and "ch" are also tracked, as well as changes in volume and pitch. The mapping to mouse pointer movements and events as used in the original Vocal Joystick application are shown.

third-party application developers can gain access to such information in real-time. The engine can also recognize non-vowel sounds, or discrete sounds, such as "ck" and "ch." The original Vocal Joystick application enabled voice-driven control of the mouse pointer using the radial mapping of vowel sounds to directions as shown in Figure 1, as well as click and toggle of the left mouse button using the discrete sounds "ck" and "ch," respectively. The volume was also mapped to the pointer speed, enabling the user to control both the pointer movement direction and speed fluidly and interactively. Studies have shown the Vocal Joystick to be effective for such basic mouse pointing and steering tasks [6, 12].

Due to the extremely short latency for vowel sound recognition compared to spoken word recognition, and the directional nature of the vowel sound mappings in the Vocal Joystick, we felt that non-speech voice-based *discrete* input can also offer significant advantages over speech-based methods. The following section describes the experiment we conducted to test this hypothesis.

3 Experimental Analysis of Non-speech Voice-Based Discrete Input

To better understand how non-speech voice input compares to keyboard and speech input as a method for generating discrete input signals, particularly under contexts requiring rapid and precisely-timed input, we conducted a reaction time (RT) experiment using an application we created called the *Vocal D-Pad*[1]. The Vocal D-Pad is built using the Vocal Joystick engine library and generates emulated directional arrow-key events in response to recognized directional vowel sound utterances. The application can also emulate the corresponding directional keys being held down for the duration of the vocalization, as well as adjacent keys being held down simultaneously (e.g., the left and up keys being held down when the vowel sound for the upper-left direction is vocalized). The system also interfaces with the Microsoft Windows Speech Recognizer to enable mapping of spoken command input to emulated keyboard events, akin to the speech macro tools mentioned in the Related Work section.

The experiment involved the presentation of directional visual stimuli and user responses generated via one of three input modalities: *key* (four arrow keys on a keyboard), *speech* (words "up," "down," "left," and "right"), and *voice* (directional vowel sounds based on the Vocal Joystick). The specific objectives and the setup of the experiment can be better understood in the context of the Model Human Processor [27], summarized next.

3.1 Experiment Background

Figure 2 shows a typical processing flow during a reaction time task in terms of the Model Human Processor. The Model Human Processor is an abstraction of the human cognitive-perceptual-motor system that can be useful in analyzing basic computer task performance. Upon the onset of a stimulus in a reaction time task, the user's

[1] The name "D-Pad" comes from "directional pad," a game input device consisting of four directional buttons.

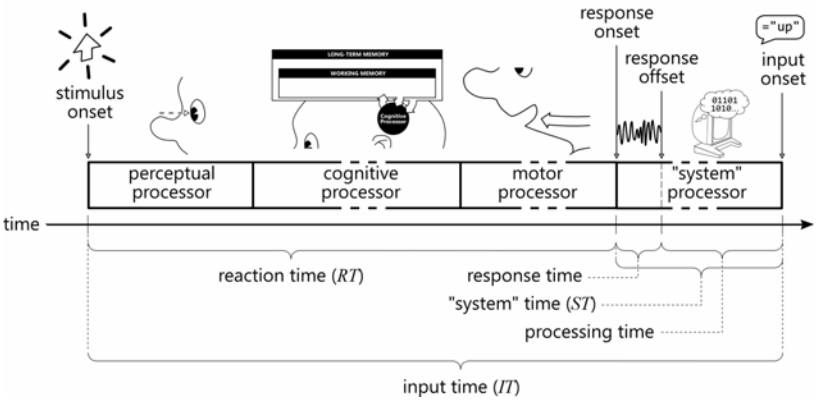

Fig. 2. Processing flow during a reaction time task based on the Model Human Processor [27] and the relevant measures. The *system processor* has been introduced to account for the time spent during response execution (*response time*) and recognition (*processing time*). At *input onset*, the recognition result is sent to the client application.

perceptual processor activates to process the relevant sensory input. The *cognitive processor* then deciphers the stimulus and determines 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. We also introduce a new module called the *system processor,* which is not in the original model since it is not part of the user. This represents any processing that the computer may need to perform on the input signal received from the user. The system processor does not add much time in cases such as button presses, but if the input is a spoken command, the system processor needs significantly more time to recognize what was uttered before it can provide its result as input to a client application. While the system processor time will be affected by the speed of the computer being used, the relative ranking of its value across different modalities should be unaffected.

Figure 2 also illustrates various measures that are of interest in a reaction time task. *Reaction time* is defined as the time between stimulus onset and *response onset*. Within the context of our experiment, response onset is defined as the earliest point when the system detects that some user response has been initiated (*e.g.*, for a button-press response, when the button switch is closed, and for a voiced response, when the microphone first detects voicing activity). *System time* is the time between response onset and *input onset*—when the system finishes processing the response signal from an input device and generates a corresponding input signal for client applications. For keyboard input, the system time is a negligible value, but for speech input using a general recognizer, the value can be significant. We refer to the total elapsed time between stimulus onset and input onset as the *input time*.

There are two types of reaction time tasks that are commonly administered: *simple reaction tasks* and *choice reaction tasks*. In a simple reaction task, there is only one type of stimulus, and the user is instructed to issue a fixed 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 response that must be correctly executed. The difference in the response times between these two tasks manifests itself in the cognitive processor time.

3.2 Hypotheses

Based on the model and representation presented above, we formulated the following hypotheses about the differences between keyboard, speech, and non-speech voice modalities with regards to their performance in reaction time tasks (summarized visually in Figure 3).

- *Hypothesis #1*: For trained users, the difference in reaction times across modalities is expected to be relatively small since the perceptual processor time and the cognitive processor time should be constant across modalities. In particular, reaction time for voice and speech modalities 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. The reaction time for the key modality may be slightly faster than that for voice or speech given that the vocal cord activation may take longer to generate an audible sound compared to the movement of a finger.
- *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 cognitive processor time and thus longer reaction time compared to the key or speech modality.
- *Hypothesis #3*: For both the simple and choice reaction tasks, the system processor time for the key modality is expected to be significantly faster than that for the speech modality, and the system processor time for the voice modality to lie somewhere in between, since non-speech vocalization should take less time to utter and be recognized compared to a speech command given its relative simplicity.

While we established the above three hypotheses in order to gain better insight into the nature of the differences among the three modalities' input times, we were also interested in the overall magnitude of those differences. We expected the key modality to be faster than the voice modality, and the voice modality to be faster than the speech modality, but the pragmatic question was, by how much, particularly in the context of computer games?

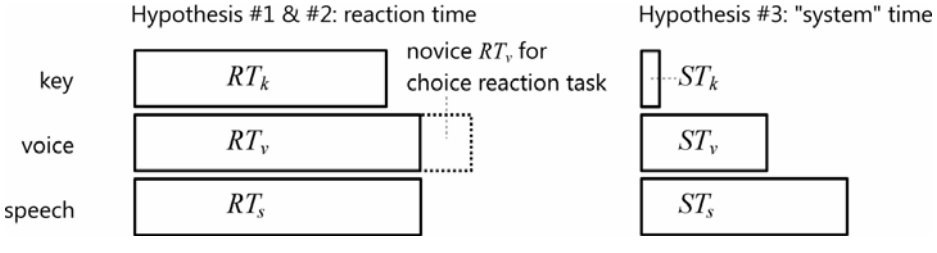

Fig. 3. Summary of the hypotheses of the experiment. Hypothesis #1 states that reaction times across modalities should be relatively similar, especially between voice and speech for trained users. Hypothesis #2 states that for a novice user, the voice modality may incur additional time to recall the unfamiliar sound-to-direction mapping. Hypothesis #3 states that the system 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.

3.3 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 group), and the rest did not (NMI group). 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 had thus been familiarized with the directional vowel mappings. One of the MI participants (MI1) 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. The other MI participant (MI2) had muscular dystrophy and was able to use the back of her fingers to press the keyboard keys.

3.4 Apparatus and Procedure

Each participant took part in a series of reaction time trials in which a test application presented them with a visual stimulus in the form of a blue arrow (100×100 pixels) on a computer screen (15" LCD monitor with resolution of 1400×900 pixels) pointing in one of the four cardinal directions. The participants were asked to respond to the stimulus in a manner dependent on a particular condition. Under all conditions, the test application waited for the correct key-down event and measured the elapsed time. The two independent factors and their levels were:

- Input modality: {key, voice, speech}
- Reaction 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 uttering one of the four cardinal vowel sounds in the Vocal Joystick vowel compass (Figure 1), which the Vocal D-Pad processed to generate an emulated key-down event. Finally, the *speech* modality involved the utterance of a direction word ("up," "down," "left," or "right"), also processed by the Vocal D-Pad in a similar fashion. The Windows Speech Recognizer's general dictation grammar was disabled and only the grammar consisting of the four directional words was activated to minimize the error rate and maximize the processing speed. For both the voice and speech modalities, each participant underwent a basic adaptation process to tune the system to their voice. Under the voice modality, the user vocalized each of the four vowel sounds for two seconds 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* reaction task, the stimulus was always an up arrow, and the participant was instructed to respond as quickly as possible using a fixed response (up key for the *key* modality, the upward vowel sound for the *voice* modality, and the word "up" for the *speech* modality). The time between the end of one trial and the appearance of the next stimulus was randomized between one and two seconds (this range of pre-stimulus interval was found in past studies to yield the fastest reaction times while reducing predictability [28, 29]). Under the *choice* reaction task, the

participant was asked to generate a response matching the arrow stimulus as quickly and as accurately as possible. In this mode, the delay between trials was one second.

For all trials, both the reaction time and system time as defined in Figure 2 were measured. In the case of the key modality, the response onset and input onset were both considered to occur simultaneously when the key-down event was generated; therefore, its system time was always zero. In the case of both the voice and speech modality, the response onset was registered when the first voiced audio frame was detected by the Vocal Joystick engine, and input onset occurred when the corresponding emulated key-down event was registered.

3.5 Results

Aggregate results from the NMI group are presented first, followed by results from the MI participants. Figure 4 summarizes the results from the NMI group. The results are grouped first by the task type, with each bar corresponding to an input modality showing the reaction time and system time portions that made up the total input time. Henceforth, results reported as significant are at $p < .05$ level.

Differences in Reaction Times. For the simple reaction task, reaction times for voice and speech modalities were comparable as expected, since they both involve the same muscle groups. Reaction time for the key modality was significantly shorter as expected [30, 31]. This is most likely due to the vocal cord activation taking longer to generate an audible sound compared to the time it takes for the finger muscle activation to result in a key being pressed. This confirms our hypothesis #1. The average difference in these reaction times (176 milliseconds) can be interpreted as the approximate difference in the motor processor times between the manual and vocal modalities

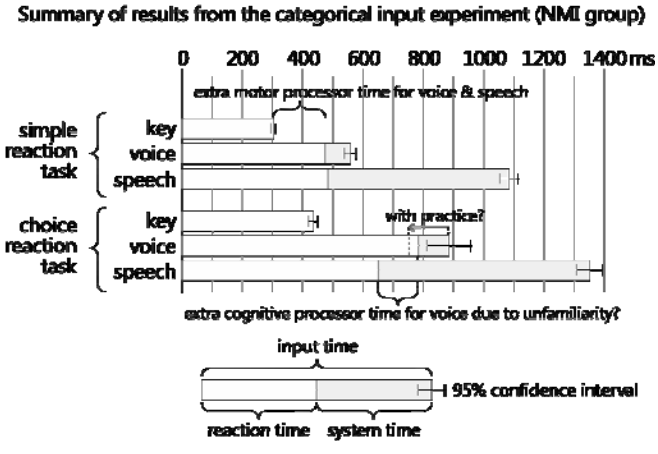

Fig. 4. Summary of the aggregate results from the categorical input experiment for the NMI group. The results are grouped by task type. Each bar shows the contribution from reaction time and system time towards the total input time, as defined in Figure 2. Error bars show 95% confidence intervals for total input times.

Under the choice reaction task, voice modality reaction time was slower than that for 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 did not manifest itself in the simple reaction task indicates that the slowdown is most likely due to the extra cognitive processing required under the choice reaction task. Our prior work [12] has shown that users can fully memorize the directional vowel mappings in under 5 hours, but it may require more practice for recall of directional vowels to become as quick and as automatic as directional words. This suggests that with practice, voice modality reaction time under the choice reaction task could approach that of the speech modality. Figure 4 depicts this potential improvement on the choice reaction task input time bar.

Differences in System Times. The comparison of system times across task types within each modality verifies that the system processor time did not depend on the task type, as expected. The significantly shorter system time for the voice modality compared to the speech modality supports our hypothesis #3, and reflects the major advantage of the use of non-speech vocal input. While it has not been precisely determined how much of this difference is due to response time or processing time, it is most likely the case that the majority of it is accounted for by the difference in processing time.

Differences in Input Times. The comparison of the total input times reveals 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 input time for the voice modality was also significantly faster than speech, by almost 50% in the simple reaction task and by 35% in the choice reaction task. The latter difference can be improved to 45% if the difference in cognitive processor time mentioned above is subtracted. While the voice modality input time still lags behind the key modality, that difference is significantly less than the improvement over the speech modality.

Results from MI Participants. The results from the two participants with motor impairments are summarized in Figure 5. For MI1, under the choice reaction task, reaction times across all three modalities were comparable (average of 1,050 ms after the learning effect adjustment described above). The fact that the key modality reaction time was comparable to voice and speech modalities is likely due to the difficulty of moving the mouth stick quickly in response to the stimulus. Also, under the choice reaction task, the voice modality input time was comparable to the key modality and almost 40% faster than the speech modality. Results for MI2 were similar as MI1, with voice modality input time under the choice reaction task (average of 780 ms after the learning effect adjustment) being only 25% slower than the key modality and almost 45% faster than the speech modality. These results offer a

promising sign for the comparative advantage of the voice modality over the key modality, especially for these participants with motor impairments.

Another interesting observation is the fact that the voice and speech input times for both MI1 and MI2 were longer than the average time for the NMI group. Our observations seem to suggest that this may be due to the effect of the MI participants' motor impairments on their lung capacity and their ability to rapidly vary vocal parameters. Further study is needed to determine the variability of such effect among individuals and how much of it can be accounted for via training.

Summary of Results. Figure 6 illustrates the results from the above study in a concrete example context of the game of Pac-Man. In the figure, the player's Pac-Man character (in yellow) is assumed to be at a standstill at a juncture, with the choice of moving up or down to move away from the chasing ghost character (in pink). The ghost character is approaching Pac-Man at a fixed speed, as indicated by the distance labeled "1 second." The multiple positions of the ghost character with corresponding input modality labels represent the closest point from Pac-Man that the ghost can be allowed to get before it is too late for the player to decide on the direction to move, execute the corresponding input, and have it be registered by the game in time

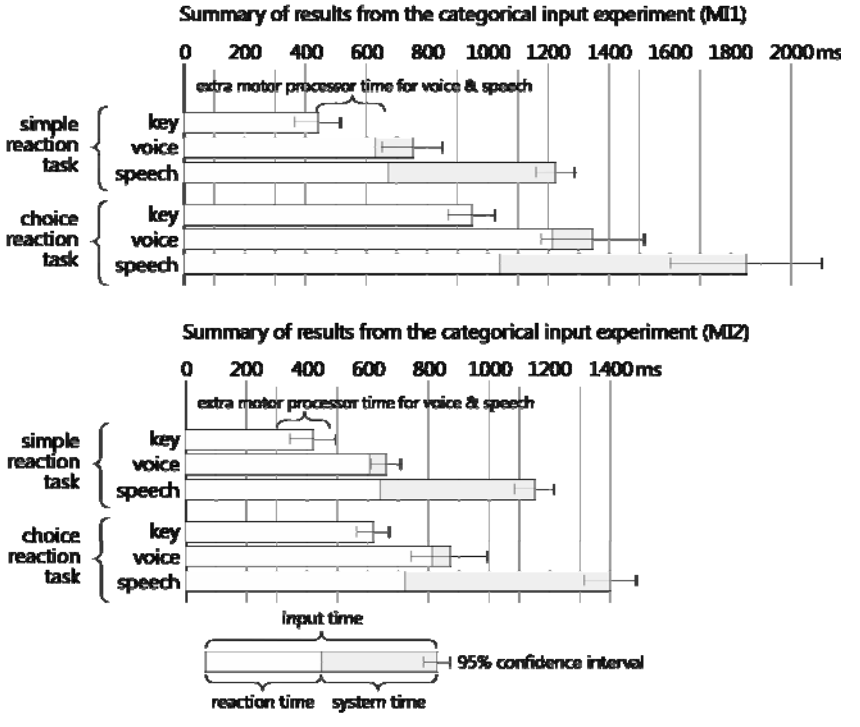

Fig. 5. Summary of the results from the categorical input experiment for the two participants in the MI group, presented in the same format as Figure 4.

to avoid contact with the ghost. In other words, these distances are the visual representations of the total input times under the choice reaction task shown in Figure 4. The position labeled "voice expert" shows the shortened distance that could be achieved if the reaction time for the voice modality is assumed to have become equal to that for the speech modality as described above. The figure highlights how significant an improvement the non-speech voice-based discrete input can offer over speech-based input, especially in the context of fast-paced time-critical computer games such as Pac-Man.

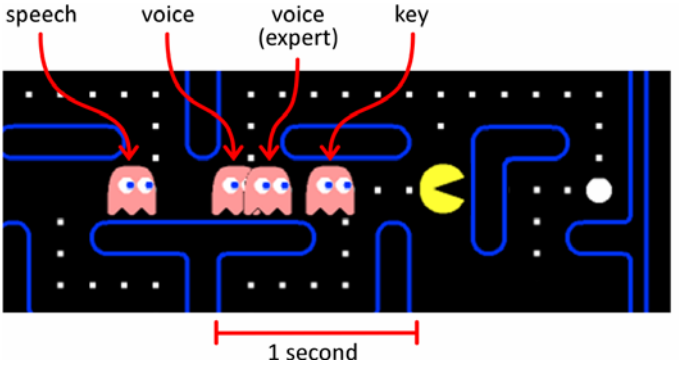

Fig. 6. An illustration of the implication of the result from the categorical input study in the context of a fast reaction time game Pac-Man. For each of the three input modalities, the figure shows the closest point that the "ghost" can be allowed to get to Pac-Man where there is just enough time for the player to decide on the direction of escape (in this case up or down), execute the corresponding response using the modality, and have the response be registered by the system.

4 The Voice Game Controller

The experimental analysis of non-speech voice-based discrete input yielded promising results over the speech-based input method. To leverage this potential and to situate it within the context of actual computer games, we built a system called the *Voice Game Controller* that extends the capability of traditional speech-based input by incorporating the rapid discrete input capability of the Vocal D-Pad and the fluid continuous input of the Vocal Joystick. The Voice Game Controller augments, rather than replaces, current speech-based input methods by retaining the option to use the dictation and command-and-control modes when non-time-critical speech input is appropriate.

The Voice Game Controller runs as a background process on the same computer as the one on which the game program is running. It translates the user's vocalizations into keyboard, mouse and joystick (KMJ) signals that game programs expect as input. Figure 7 shows a high level representation of the Voice Game Controller architecture. The Voice Game Controller utilizes the Windows Speech Recognizer to recognize spoken command input, and the Vocal Joystick engine [11] to process non-speech vocalizations uttered by the user. Since each game requires different sets of input controls, the mapping between vocal input and corresponding KMJ signals is speci-

fied separately for each game in *Voice Game mapping profiles*. The *input mapping engine* passes the recognized voice input through a Voice Game mapping profile corresponding to the currently active game program, and generates the corresponding KMJ signal via the keyboard, mouse, and joystick emulator modules. Because the emulated KMJ signals are generated at the system level, they are application-independent and thus can be used to supply voice-driven KMJ input to non-gaming applications as well. From the game program's perspective, it is as if the user is using a standard keyboard, mouse, or joystick.

The keyboard emulator module can generate key-down and key-up events for any key or set of keys, as well as repeated key-down events to emulate the user holding down a key. The mouse and joystick emulators can generate two-dimensional directional vectors (when in relative mode) or positional vectors (when in absolute mode), as well as button down and button up events corresponding to physical buttons on these devices. All these types of signals can be specified as the output of a mapping in a Voice Game mapping profile.

There are several ways in which spoken command input and non-speech vocal input can be mapped to the KMJ signals just described. First, spoken commands and non-speech discrete sound vocalization can be mapped to any of the key events for the keyboard emulator or to the button events for the mouse and joystick emulators. Second, non-speech vowel vocalization can be mapped to two-dimensional vectors based on the Vocal Joystick vowel compass mapping shown in Figure 1, with its magnitude possibly determined by the volume or pitch. This capability alone expands the realm of games playable by voice to include the numerous mouse-driven games. Third, non-speech vowel vocalization can also be mapped to key and button events similar to spoken commands and discrete sounds. This is done by determining the

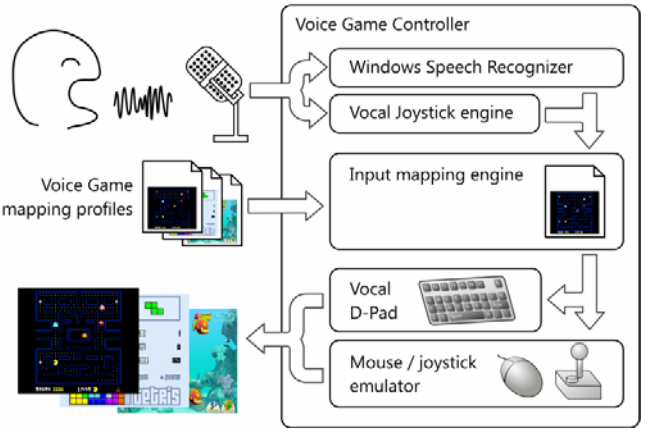

Fig. 7. Voice Game Controller architecture. User's utterances are first processed by the Windows Speech Recognizer and the Vocal Joystick engine, and the corresponding keyboard/mouse/joystick signals are emulated based on the currently active *Voice Game mapping profile*.

vowel sound with the maximum probability every frame, instead of aggregating the probabilities across all vowels, and generating a binary output of whether or not that probability exceeds some threshold. If it does, it is treated as the *vowel onset* and a corresponding key or button event can be generated. The *vowel offset* event occurs for the current vowel when the vowel probability drops below the threshold, a different vowel's probability becomes higher, or the user stops vocalizing. The directional mapping of the Vocal Joystick vowel sounds make them ideal for mapping to the arrow keys in the context of games.

In the following section, we present our initial findings from a preliminary evaluation of the Voice Game Controller we conducted in order to assess its effectiveness in the context of four different games, comparing its performance to speech-driven control as well as to the participants' preferred input devices.

4.1 Preliminary Evaluation of the Voice Game Controller

Results from the comparative analysis of categorical input above 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, we conducted a preliminary evaluation of the Voice Game Controller with the following four games to assess the viability of using the system to play real computer games.

Selected Games. We chose four games that lie at various points in the input signal space. Two Rooms [32] is a puzzle-based maze game in which the player controls square pieces using directional arrow keys. Although the time to complete each puzzle is considered, once the sequence is known, the emphasis is on how consistently and reliably the desired directional input can be generated. Pac-Man is a classic arcade game in which the player controls the Pac-Man character through a maze using directional arrow keys, attempting to eat all the white dots while avoiding four ghost characters. The game requires precise timing to avoid contact with a ghost character or to make a turn. The player's score corresponds to the number of white dots and frightened ghosts Pac-Man consumes. Tetris is a puzzle game in which the player rotates and positions falling two-dimensional tiles using arrow keys such that the tiles form solid horizontal layers, which then disappear from the playing field, yielding points for the player. It is necessary to be able to execute a rapid sequence of left or right movements to quickly move a piece into the desired position. Finally, Fish Tales [33] is a game in which the player controls a red fish in a two dimensional scene by using the mouse. The objective of the game is to eat fish that are smaller than the player's fish, while avoiding contact with larger fish. The score is based on the number of fish eaten. All of the fish move at varying speeds and directions, requiring the player to smoothly and rapidly steer the mouse pointer.

Evaluation Setup. Five of the eight participants who took part in the reaction time study, including the two participants with motor impairments (MI1 and MI2),

participated in the evaluation of the Voice Game Controller. Each participant came in for four 90-minute sessions separated by 48 hours or less. Throughout the sessions, we observed the users play each of the four games using three input modalities: non-speech voice input mode, speech-only input mode, and the user's preferred input device. The speech-only input mode represents existing speech-based control methods, namely uttering directional words to emulate corresponding arrow key events and using features such as Speech Cursor and Mouse Grid for pointer control. For the preferred input device, the three participants in the NMI group chose the mouse and keyboard, while MI1 with arthrogryposis chose a mouth stick with the MouseKeys feature for controlling the pointer, and MI2 with muscular dystrophy chose the touchpad and keyboard.

During the first session, participants were shown how to use the Voice Game Controller and the speech-only input mode, and then introduced to the four games. For the remainder of the sessions, they played each of the games for 15 minutes at a time, spending 5 minutes on each of the three input modalities per game. At the end of the final session, a timed evaluation was conducted in which the participants played all games for five minutes each using each of the three input modalities. 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 at all. For the Two Rooms game, the measure of performance was based on the average completion time across all completed stages. The other three games were measured based on the average score achieved.

Results and Observations. Figure 8 summarizes the results from the evaluation described above. The data is presented in two graphs, with the graph on the right showing how the Voice Game Controller did compared to the speech modality, and the graph on the left showing how it did compared to the user's preferred device. The horizontal axes represent how much better or worse the Voice Game Controller did with respect to the corresponding modality, with ±0% representing no difference in performance, and positive values indicating that it did better by that much percentage points.

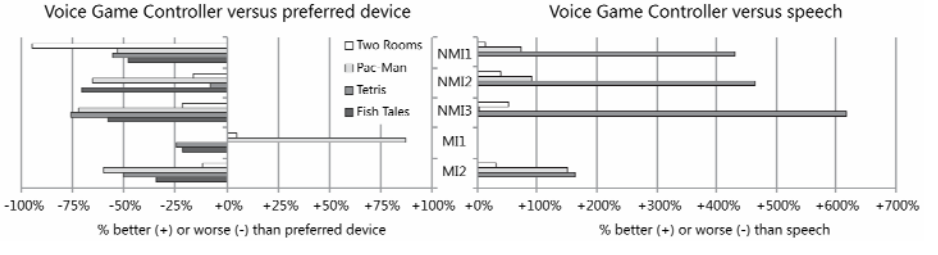

Fig. 8. Results from the Voice Game Controller user evaluation. The graph on the left shows how each participant scored in each game using the Voice Game Controller relative to their preferred device. The graph on the right shows the relative performance of the Voice Game Controller against speech-based input. ±0% indicates that the Voice Game Controller score was on par with the corresponding modality, and positive percentage indicates that the Voice Game Controller score was that much higher.

Compared to speech, the Voice Game Controller performed significantly better across all games for all participants (MI1's speech data was omitted due to incomplete tasks). For Fish Tales, the fact that the participants were able to play it at all using voice was a clear win over speech, with which the game could not be played. Within the NMI group, Voice Game Controller was 40% faster than speech in Two Rooms, and the average score was 50% higher in Pac-Man and 500% higher in Tetris. Within the MI group, Voice Game Controller was 32% faster than speech in Two Rooms, and the average score was 150% higher in Pac-Man and 160% higher in Tetris.

The better performance by the preferred input devices was expected, as was seen in previous comparative studies [20]. Within the NMI group, the Voice Game Controller performance was 40% worse than the preferred device for Two Rooms, and between 60% and 65% worse for the other three games. For the MI group, the Voice Game Controller performance was comparable to their preferred devices for Two Rooms, and between 60% and 70% for the other three games. While the preferred device performed better than the Voice Game Controller even for the MI group, both of the MI participants expressed that the fatigue induced by attempting to manipulate their physical devices was less desirable than the hands-free nature of the Voice Game Controller. None of the participants raised vocal fatigue as an issue during the study.

5 Conclusion

This paper presented results from our investigation into the viability of using non-speech voice-driven input for expanding the scope of computer games that can be controlled hands-free using voice only. Particularly, our experimental analysis into the use of non-speech vocalization for discrete input revealed a significant performance improvement over existing speech-based input method. Based on these findings, we built a prototype system called the Voice Game Controller that integrates the expressivity of traditional speech-based input with the fluid continuous input offered by the Vocal Joystick engine, as well as the non-speech voice-driven discrete input functionality of the Vocal D-Pad. The better performance obtained by the Voice Game Controller compared to standard speech-driven input in actual games 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 hands-free nature of voice-driven input may also serve as a new game input modality for the general population, expanding the realm of interactive entertainment.

Acknowledgments. We would like to thank all the study participants who volunteered their time to try out our system. This work was supported in part by the National Science Foundation under grant IIS-0326382. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was also supported by the University of Washington Royalty Research Fund.

References

1. Ahn, L.V., Dabbish, L.: Designing games with a purpose. Commun. ACM. 51, 58–67 (2008)
2. Sacks, J.J., Helmick, C.G., Luo, Y., Ilowite, N.T., Bowyer, S.: Prevalence of and annual ambulatory health care visits for pediatric arthritis and other rheumatologic conditions in the United States in 2001-2004. Arthritis Rheum. 57, 1439–1445 (2007)
3. Second Life Official Site, http://secondlife.com/
4. Riemer-Reiss, M.L., Wacker, R.R.: Factors associated with assistive technology discontinuance among individuals with disabilities. Journal of Rehabilitation 66, 44–50 (2000)
5. Tse, E., Greenberg, S., Shen, C., Forlines, C.: Multimodal multiplayer tabletop gaming. Computers in Entertainment (CIE) 5, 12 (2007)
6. Harada, S., Landay, J.A., Malkin, J., Li, X., Bilmes, J.A.: The Vocal Joystick: evaluation of voice-based cursor control techniques for assistive technology. Disabil. Rehabil.: Assistive Technology 3, 22 (2008)
7. Harada, S., Wobbrock, J.O., Landay, J.A.: 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. ACM, Tempe (2007)
8. Igarashi, T., Hughes, J.F.: 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. ACM, Orlando (2001)
9. de Mauro, C., Gori, M., Maggini, M., Martinelli, E.: Easy access to graphical interfaces by Voice Mouse. Università di Siena (2001)
10. Sporka, A.J., Kurniawan, S.H., Slavik, P.: Whistling User Interface (U3I). In: Stary, C., Stephanidis, C. (eds.) UI4ALL 2004. LNCS, vol. 3196, pp. 472–478. Springer, Heidelberg (2004)
11. Malkin, J., Li, X., Harada, S., Landay, J., Bilmes, J.: The Vocal Joystick Engine v1.0. Computer Speech & Language 25, 535–555 (2011)
12. Harada, S., Wobbrock, J.O., Malkin, J., Bilmes, J.A., Landay, J.A.: 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. ACM, Boston (2009)
13. Gibbs, M., Wadley, G., Benda, P.: 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. Murdoch University, Perth (2006)
14. Wadley, G., Gibbs, M.R., Benda, P.: 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. Creativity & Cognition Studios Press, Sydney (2005)
15. Shoot 1.6.4, http://clans.gameclubcentral.com/shoot/
16. Voice Buddy Interactive Voice Control Version 3.0, http://www.edimensional.com/index.php?cPath=23
17. VR Commander - Voice Command and Control for you Computer, http://www.vrcommander.com/
18. Vocola - A Voice Command Language, http://vocola.net/
19. Windows Speech Recognition Macros, http://code.msdn.microsoft.com/wsrmacros

20. Harada, S., Landay, J.A., Malkin, J., Li, X., Bilmes, J.A.: 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. ACM, Portland (2006)
21. Konami Digital Entertainment, Inc.: Karaoke Revolution, http://www.konami.com/kr/
22. Rock Band®, http://www.rockband.com/
23. PAH! - The first voice-controlled and activated game on the web, http://games.designer.co.il/pah/
24. shout n dodge, http://www.weebls-stuff.com/games/shout+n+dodge/
25. Racing Pitch, http://jet.ro/games/racing-pitch/
26. Sporka, A.J., Kurniawan, S.H., Mahmud, M., Slavík, P.: 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. ACM, Portland (2006)
27. Card, S.K., Newell, A., Moran, T.P.: The Psychology of Human-Computer Interaction. Laurence Erlbaum Associates Inc., Hillsdale (1983)
28. Izdebski, K.: Effects of prestimulus interval on phonation initiation reaction times. Journal of Speech and Hearing Research 23, 485–489 (1980)
29. Shipp, T., Izdebski, K., Morrissey, P.: Physiologic stages of vocal reaction time. Journal of Speech and Hearing Research 27, 173–178 (1984)
30. Nebes, R.D.: Vocal versus manual response as a determinant of age difference in simple reaction time. Journal of Gerontology 33, 884–889 (1978)
31. Venables, P.H., O'connor, N.: Reaction times to auditory and visual stimulation in schizophrenic and normal subjects. Q. J. Exp. Psychol. 11, 175 (1959)
32. Armor Games: Two Rooms, http://armorgames.com/play/3006/two-rooms
33. Fish Tales, http://flashgamesite.com/play334game.html