VoiceRelay: Voice key operation using Visual Basic

LISE ABRAMS
University of Florida, Gainesville, Florida

and

DAVID T. JENNINGS
Gainesville, Florida

Using a voice key is a popular method for recording vocal response times in a variety of language production tasks. This article describes a class module called VoiceRelay that can be easily utilized in Visual Basic programs for voice key operation. This software-based voice key offers the precision of traditional voice keys (although accuracy is system dependent), as well as the flexibility of volume and sensitivity control. However, VoiceRelay is a considerably less expensive alternative for recording vocal response times because it operates with existing PC hardware and does not require the purchase of external response boxes or additional experiment-generation software. A sample project demonstrating implementation of the VoiceRelay class module may be downloaded from the Psychonomic Society Web archive, www.psychonomic.org/archive.

Measuring vocal response times in psychological research is used in a variety of paradigms, such as word naming (e.g., Tyler, Voice, & Moss, 2000), nonword naming (e.g., Light, Kennison, Prull, & La Voie, 1996), and picture naming (e.g., Taylor & Burke, 2002). Voice keys are the most common means of measuring vocal response times, although recent research has demonstrated their susceptibility to inaccuracies. Suggested causes of these inaccuracies are biases from specific phonemes in influencing voice onsets (e.g., Kessler, Treiman, & Mullennix, 2002; Rastle & Davis, 2002) as well as inadvertent triggers of the voice key, such as unintentional vocalizations or environmental noise (e.g., Baker, Taglialatela, & Washburn, 2001). Voice keys consist of an electronic device that interfaces between a microphone and the computer. An event such as word presentation activates the voice key, which then monitors the sound level from the microphone. When the sound level exceeds a user-specified threshold, the voice key records the amount of time (in milliseconds) that has elapsed since the voice key was enabled.

The VoiceRelay program was designed to fully implement the functionality of a traditional voice key with the flexibility of the Visual Basic programming environment. Visual Basic offers a graphical interface and a simpler programming language, making it relatively easy to learn, compared with other programming languages such as $C/C++$.[1] In recent years, the development of experiment-generation programs has enabled the use of a software-based voice key for research with PC-compatible computers. However, these software programs are costly (e.g., E-Prime [Psychology Software Tools, 2000] costs $695 for a single user) and usually require the purchase of additional hardware to work in conjunction with the voice key. For example, the Psychology Software Tools (PST) deluxe serial response box is used for voice key operation in conjunction with E-Prime and costs an additional $450 for an individual box (Psychology Software Tools, 2000). If experiments are run on multiple computers, the necessary software and hardware for all of the machines can be extremely costly. Furthermore, unlike response boxes, Visual Basic offers the potential for recording the actual speech input from the microphone.

In addition to their expense, response boxes have other problems. First and foremost, the hardware always has the potential for failure; for example, the authors used two button boxes in conjunction with PsyScope (Cohen, MacWhinney, Flatt, & Provost, 1993), one in which the threshold volume control was unreliable and another that required the microphone input jack to be resoldered. Conversely, most modern computers anticipate frequent use of the hardware connections and therefore have higher structural integrity for repeated use. A second critical issue is the necessity for accurate timing. Given that vocal response times are used as indicators of underlying cognitive processes, the need for resolution in milliseconds is essential in psychological research, although lower resolutions may be acceptable for certain tasks (see Snodgrass & Yuditsky, 1996). However, the experiment-generation program Inquisit (Inquisit Millisecond Software, 2002) reports that its measurement of voice input is only accurate within ± 10 msec.

Correspondence concerning this article should be sent to L. Abrams, Department of Psychology, University of Florida, P. O. Box 112250, Gainesville, FL 32611-2250 (e-mail: abrams@ufl.edu).

One PC-compatible experiment-generation program that does not require an external response box is SuperLab for Windows (Cedrus Corp., 2000). This program uses a built-in microphone in conjunction with a software program for voice key operation. However, in December 2000, technical support reported that the software voice key does not work reliably under the Windows environment, leading to inaccurate timing, and that "users are advised to avoid using it for research" (Cedrus Corp., 2000). The Web site also reports that Cedrus is developing new voice key hardware from scratch; presently, however, the voice key has not been re-released for SuperLab for Windows, suggesting that these problems have not been easily resolved.

VoiceRelay overcomes the disadvantages of current voice key implementations. First, VoiceRelay utilizes standard PC hardware (sound card and microphone) and does not require the purchase of additional hardware, such as an external response box. Second, VoiceRelay is ideal for existing Visual Basic programmers who already possess the necessary software and programming knowledge, because VoiceRelay is a class module that can be used in any Visual Basic project. VoiceRelay also has the potential, with modifications, to work within other programming languages, such as C/C++ and Delphi. Third, with further modifications, Visual Basic can provide the capability to record actual vocal responses, in addition to capturing voice onset times. Last, although VoiceRelay does not overcome the inaccuracies of voice key timing, it has the potential to be as accurate as external response boxes, depending on system constraints, making it a useful tool for psychological research.

High-Resolution Timing

When the experimenter uses a voice key to determine a parameter such as vocal response time, a primary consideration is the precision and accuracy of the reaction time measurement. Precision is concerned with the repeatability of the time measurement; if the measurements obtained with a voice key can be repeated with identical results, the timing is said to be precise. In contrast, accuracy is the difference between the measured response time and the "true" value of voice onset. If this difference is small, the timing is said to be accurate.

Since vocal response times are used as indicators of cognitive processes (e.g., Kessler et al., 2002; Rastle & Davis, 2002), precision and accuracy are important considerations. Unfortunately, the basic architecture of the PC system can severely limit timing precision because of the inherent "slowness" of its run-time clock. The PC run-time clock operates at a frequency of 18.2 Hz, which means that the smallest interval between clock "ticks" is approximately 55 msec (Microsoft KB 81592, 1999). This limitation implies that the precision of *any* timing measurement obtained using the standard Visual Basic or the standard Windows interface (i.e., GetTickCount function) is limited to approximately ±55 msec.

However, the Windows timing problem can be overcome by accessing the high-resolution multimedia timers that may be present in the system from sound cards, MIDI sequencers, and so forth. To utilize these multimedia timers, the Windows Application Programming Interface (API) provides the QueryPerformanceCounter function, which can be utilized to perform timing operations with a precision "on the order of a microsecond" (Microsoft KB 172338, 2003; Microsoft KB 274323, 2000).

The VoiceRelay class utilizes a code module that was written to encapsulate the Windows QueryPerformance-Counter function into a user-friendly format. The benefit of using the high-precision timer is that it also improves the accuracy of response time measurements. For example, the extra time it takes to execute the API calls ("overhead") can be precisely measured and subtracted from the response time to yield a more accurate value. In the event the computer system does not support QueryPerformance-Counter, the lower precision (on the order of 10 msec) TimeGetTime function is incorporated into the timer module to preserve overall functionality (Microsoft KB 172338, 2003).

Features of VoiceRelay

The overall goal for VoiceRelay was to provide essentially "plug-and-play" functionality. As such, the VoiceRelay functions were written into a Visual Basic class module to simplify the user interface. With this implementation scheme, the experimenter can access all of the voice key functions and parameters simply by declaring a variable of the VoiceRelay type.

The VoiceRelay class module is dependent upon three other code modules, which were written to simplify the calls to the Windows API: (1) the "modMicVolCtrl.bas" module allows control over the PC microphone volume level; (2) the "modWaveIn.bas" module facilitates recording from the PC microphone; and (3) the "modTimer.bas" module provides high-resolution measurement of elapsed time. All three of these code modules must be incorporated into the main Visual Basic project in order for the VoiceRelay class module to function. This "dependent" configuration provides the flexibility of easily upgrading the individual components without compromising the overall operation of the voice key.

VoiceRelay Implementation

The core functions required for voice key operation are initialized when a variable of the type VoiceRelay is declared. This initialization process also detects whether or not the computer has a sound card that will support the required recording capabilities. If any of the initialization steps fail (which is unlikely for most modern computers), the program automatically terminates and releases any allocated memory.

The VoiceRelay class module implements voice key operation by providing the ability to perform a variety of functions directly from code. Using Visual Basic class properties, which store information about the VoiceRelay object, the experimenter can set three parameters relevant to voice key operation. The first parameter is the volume

level of the Windows microphone, expressed as a percentage. The default value for this parameter is 80%, which can either be raised or lowered depending upon the level of background noise in the testing environment. The second parameter is the threshold volume level (e.g., for detection of speech onset), which is expressed as a percentage of the microphone volume control level. The default value for this parameter is 20%. Voice key operation is directly dependent upon these first two parameters. Therefore, acceptable values for these parameters should be determined by trial and error to ensure consistent voice key operation. The third and final parameter is the maximum duration for the audio search in milliseconds. This parameter allows the experimenter to control how long the program will search for audio input before determining that there was no response. The default value has been arbitrarily set to 5,000 msec.

In addition to the properties that allow setting and retrieval of voice relay operational parameters, the class module offers three additional properties to provide the experimenter with information about the voice key. The first property indicates the real-time volume level being recorded through the PC microphone, expressed as a percentage of the microphone volume setting. This property can be used to create a "peakmeter" display, which may be useful in assessing the appropriate threshold value. The second property reports the elapsed time in milliseconds between the initiation of the audio search and the receipt of audio input that exceeds the threshold volume level. This value is adjusted for the processing overhead required by the voice relay and sound recording operations. The third and final property is the value of the code-processing overhead in milliseconds, which is useful in determining how much impact any real-time audio displays have on the voice relay operation.

In addition to class properties, VoiceRelay offers two class methods, which are specific actions or procedures that the VoiceRelay object can perform. The first method provides the ability to reset all of the voice key parameters to their default initialization state. The second method initiates recording from the PC microphone and begins voice key operation.

Obtaining Audio Input

The core method in the VoiceRelay class module implements the functions required to get audio input from the PC microphone and to report the elapsed time. This method begins by obtaining the necessary sound buffers for audio recording, followed by resetting of the "stopwatch" to ensure a precise elapsed time measurement. The code then enters a loop that terminates only if either (1) the input audio volume level exceeds the threshold volume level or (2) the elapsed time exceeds the maximum audio search duration. The method ends by retrieving the elapsed time of the audio search from the high-resolution timer and accounting for the code-processing overhead.

The audio input routines operate by using several of the "WaveIn" functions from the Windows API. To initiate recording from the PC microphone, the following steps are performed: (1) set up the audio recording parameters (i.e., wave format, sampling frequency, number of channels, etc.); (2) allocate audio recording buffers in system memory; (3) send the recording buffers to the sound card; and (4) open the WaveIn device for recording. Once recording is initiated, the incoming sound data are processed until recording is terminated. The sound processing routine is a loop that performs the following actions: (1) wait for the audio buffer to be marked as "full"; (2) copy the audio data into an array to allow manipulation; (3) normalize the data from –100% to +100% (midpoint 0%) for ease of use; and (4) provide the normalized data in a public variable. The timer is set to zero immediately before the audio processing routine is started, and the ElapsedTime measure is taken at the instant the threshold is exceeded or audio timeout is reached.

The GetAudioInput method of the VoiceRelay class module contains a loop that continually retrieves the public volume variable from the WaveIn operations. The method then compares the volume values of the received audio input with the threshold volume level to determine whether or not to continue the audio search. Whenever the threshold volume level is reached, the elapsed time in milliseconds is retrieved from the high-resolution timer, and the code-processing overhead is updated. Once the threshold is exceeded or a timeout has occurred, recording from the PC microphone is terminated to free system resources.

The fundamental code for audio processing was largely based on the examples provided in Microsoft Knowledge Base Articles 187673 and 178456 (Microsoft, 2001). The functionality for setting and retrieving the value of the Windows microphone volume was taken from Microsoft Knowledge Base Article 178456 and encapsulated into a code module to provide a simpler user interface. Likewise, the basic functionality for performing the wave-recording operations was taken from Microsoft Knowledge Base Article 187673 and encapsulated into a user-friendly code module. However, the Knowledge Base Articles originally published by Microsoft required that the sound card have a peakmeter function on the recording control. It was quickly determined that this function is not supported by most modern sound cards. As a result, the core audio recording routine was mostly rewritten to remove dependence on the Windows mixer and to save the waveform audio data directly into an array of bytes. The resulting code not only corrected compatibility limitations with the VoiceRelay class module, but also improved performance because the audio processing is done inside of a Do . . . Loop instead of within a high-frequency timer control object, such as the Common Controls Replacement Project high-performance timer objects (CCRP, 2002).

Testing of VoiceRelay's Precision and Accuracy

Precision and accuracy of the information provided by VoiceRelay depend on the core timing functions. Although these timing functions are subject to various influences, we attempted to address the three most significant ones: (1) the

placement of the timer starts and stops, (2) code overhead, or the amount of time to execute code without actually recording volume information, and (3) sound card latency, or the delay required by a sound card to get a buffer.

VoiceRelay used the Windows API QueryPerformanceCounter function instead of the CCRP stopwatch to allow more control over the placement of the timer starts and stops without the increased overhead (and memory) of additional variables. As a result, QueryPerformanceCounter allows for better measurement of the code overhead, which is later subtracted from the reported ElapsedTime to improve accuracy. Sound card latency could not be controlled but was measured with a sound card test program, such as the one available at http://www.guitar-fxbox.com/sctest.htm.

Four computers of varying processors, processor speeds, and operating systems were used to test the precision and accuracy of VoiceRelay, and the results of these tests are displayed in Table 1. The precision was measured by the Timer module during the initialization of the timers. Accuracy was evaluated through actual use of VoiceRelay as a voice key. Two "blips" were added to the code to indicate the start and end of audio recording (Microsoft KB 86281, 2003). The procedure was to (1) start an external independent tape recorder, (2) start the VoiceRelay demo program, (3) trigger the relay using a sharp sound (a handclap), and then (4) stop the external tape recorder. The audio data from the tape recorder was then imported into Audacity Version 1.0.0 (http://audacity.sourceforge. net/), where the waveform was analyzed to determine the "actual" elapsed time. All of the data were then collected and analyzed with Microsoft Excel 97 to evaluate the difference between the actual audio elapsed time (as determined by waveform) and the measured elapsed times reported by VoiceRelay.

The accuracies assessed by waveform that are presented in Table 1 represent 95% confidence intervals that reflect how close the measured time was to the actual time; note that negative values indicate an underestimation in the actual time. However, there are several potential sources of error in these data, such as: (1) the waveforms were analyzed by hand, (2) it was unknown how much of the 10-msec blip (audio start) should be removed from the difference calculations, and (3) the waveforms read by VoiceRelay are likely to be different from those read by the external tape recorder, making it impossible to pick the exact location of the audio end. Therefore, the accuracies shown in Table 1 are likely to be more conservative than the true values.

In an attempt to quantify how conservative the accuracy measurements might be, VoiceRelay was tested with the "timeout" function already built into the class module since it would remove the "human" component associated with waveform analysis. The VoiceRelay demo was started and then allowed to timeout without receiving any audio input. The data were then collected and analyzed with Microsoft Excel 97 to evaluate the difference between the theoretical timeout value and the elapsed times reported by Voice-Relay (see Table 1 for 95% confidence intervals). The results on accuracy assessed by the timeout function do, in fact, indicate a measure of conservatism in the waveform accuracy data.

The results of the preceding tests indicate that the precision of VoiceRelay was more dependent on the operating system than on the processor speed, but that overall precision was extremely high. In contrast, the accuracy of VoiceRelay was dependent on processor/speed and operating system (see Table 1). In addition, high sound card latency may cause low voice onset accuracy in VoiceRelay. The computers running Windows 98 showed greater accuracy for the Pentium II processor, which had a sound card latency of 26 msec, relative to the Pentium III processor, which had a sound card latency of 63 msec.[2]

Sample Implementation

A sample Visual Basic project was created to demonstrate the capabilities and functions of the VoiceRelay class module. A screen of the sample program is shown in Figure 1. The large text pane on the left provides instructions on how to use the VoiceRelay demonstration program. The three slider bars on the right permit changing of the three critical VoiceRelay properties: the microphone volume level, the threshold volume level, and the audio search duration. Moving the sliders back and forth dynamically changes these parameters.

The System Info and Timer Info buttons near the bottom right provide basic performance information about the PC system, if available. The system information that can be obtained is the operating system, processor model, processor speed, total RAM, and free RAM (Microsoft

Table 1
Tests of VoiceRelay's Precision and Accuracy as a Function of Computer
Processor, Processor Speed, and Operating System

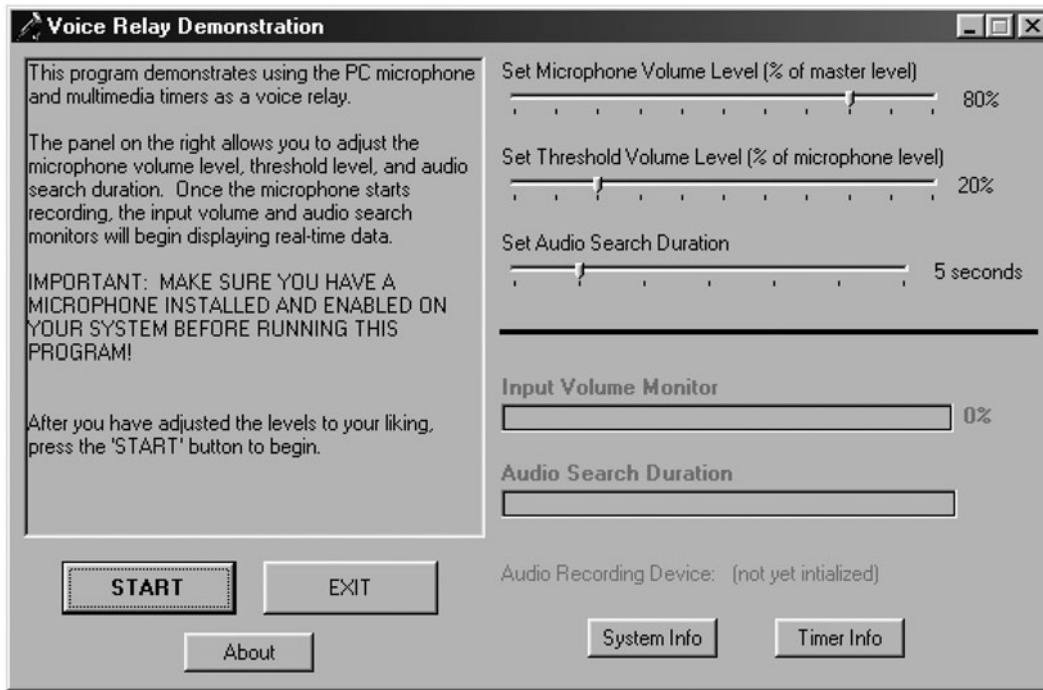| Operating System Processor/Speed	Precision (in msec)	Accuracy by Waveforms (in msec)	Accuracy by Timeout (in msec)
Windows 98			
Pentium II/233 MHz	0.0008381	−85 to −17	26 to 53
Pentium III/450 MHz	0.0008381	129 to 190	0 to 1
Windows XP			
Pentium III/600 MHz	0.0002794	2 to 59	5 to 8
Pentium IV/2.4 GHz	0.0002794	83 to 135	112 to 118

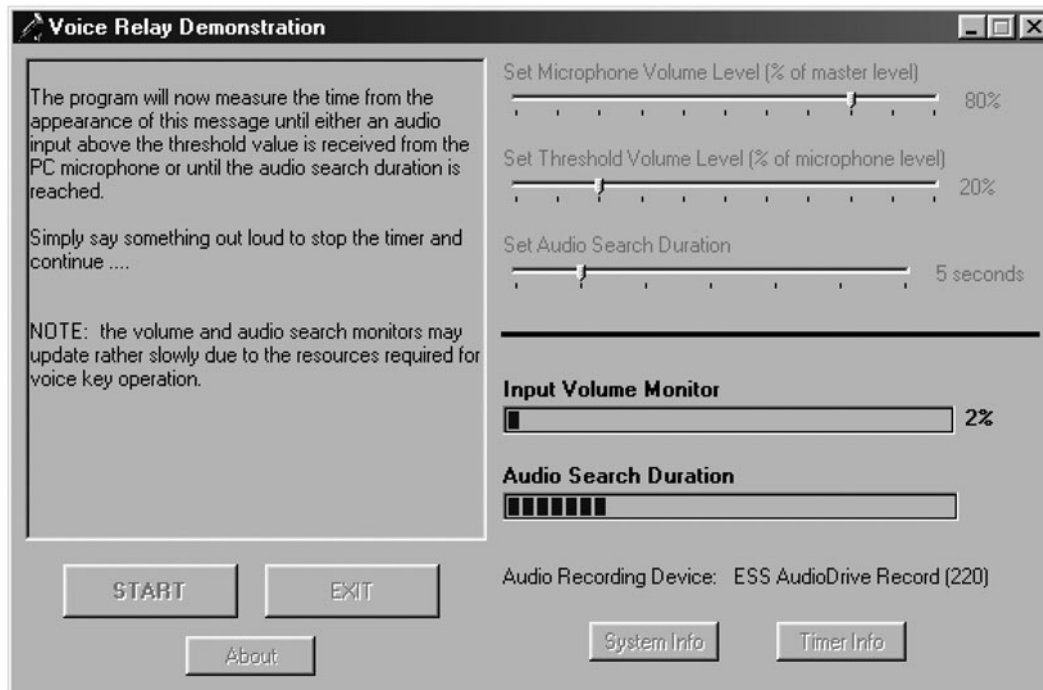**Figure 1. Startup screen of the VoiceRelay demonstration program.**

Figure 2. Screen display of the VoiceRelay demonstration during operation.

KB 145679, 2001; Microsoft KB 147886, 2003). The timer information indicates whether the system is using high-resolution or low-resolution timers. In either case, the theoretical minimum resolution of the timer is displayed. In the case of high-resolution timers, the overhead required to make the API calls is also reported.

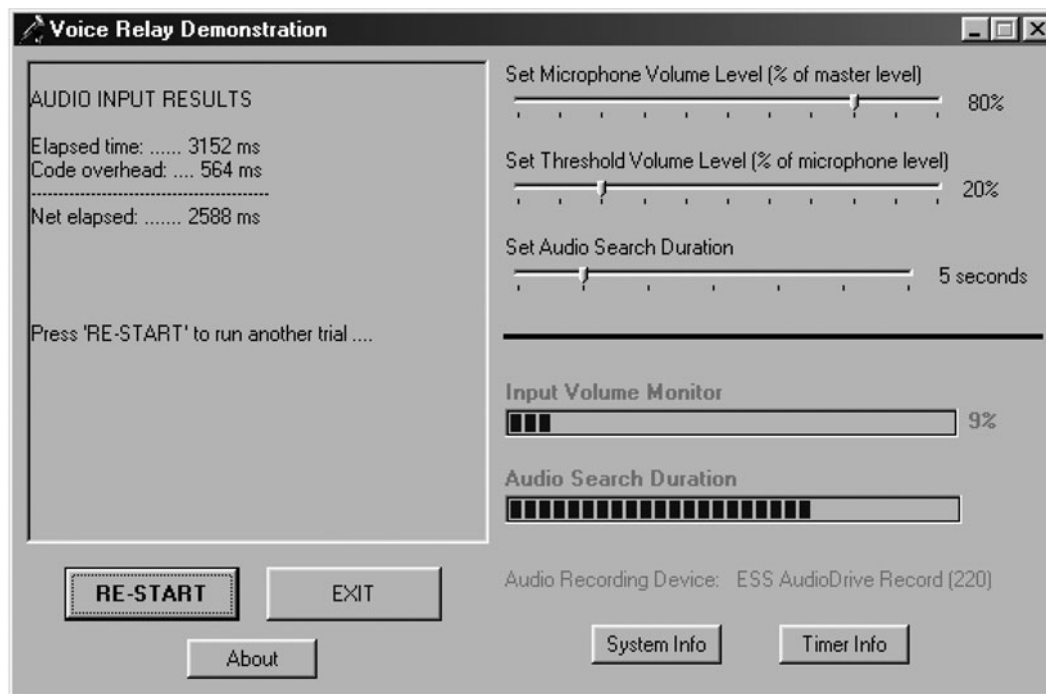The grayed-out "Input Volume Monitor" and "Audio Search Duration" progress bars in the lower right monitor

Figure 3. Screen display of the VoiceRelay demonstration timing result.

the process of voice relay operation when activated. The input volume monitor will display a dynamic reading of the microphone volume level, and the audio search duration will continuously update the elapsed time. These monitors will be activated when the START button is pressed. Finally, once audio recording has begun, the display will be updated with the name of the system audio recording device.

In the initial startup screen (Figure 1), the START button initiates voice key operation and moves the program to the second screen, where audio and timing input are collected. In the second screen (Figure 2), the buttons are grayed out while the audio search is performed. Finally, in the third screen (Figure 3), the RE-START button returns the program to the first screen for another round of voice key operation. In all three of the screens, the EXIT button terminates the program and releases any allocated memory, and the About button displays information about the authors of the program.

System Requirements

The VoiceRelay class module requires a PC with at least a Pentium II processor, Microsoft Windows 95, and Microsoft Visual Basic 5.0 Service Pack 3. Initial tests have indicated that VoiceRelay will also work with later versions of Windows, such as 98, 2000, and XP, and faster processors, such as Pentium III and IV. The PC must have a sound card with a microphone input jack.

Conclusions

In sum, the VoiceRelay class module facilitates psychological research using voice keys by providing an in-

expensive alternative to response boxes and experiment-generation software without sacrificing timing precision. Furthermore, VoiceRelay has potentially good levels of accuracy, depending on the computer's processor, operating system, and sound card. Although some researchers have suggested examining on-screen audio waveforms as an alternative to voice keys (e.g., Baker et al., 2001), examination of waveforms is undesirable because it is a time-consuming process and is prone to human error. Therefore, voice keys will continue to play a pivotal role in psychological research, and we hope that VoiceRelay will facilitate the measurement of vocal responses.

REFERENCES

Baker, L. A., Taglialatela, J. P., & Washburn, D. A. (2001, June). *On-screen audio waveform as a viable alternative to the voice key.* Poster presented at the 13th annual convention of the American Psychological Society, Toronto.

Cedrus Corp. (2000). *SuperLab Pro input options* [On line]. Available at http://www.superlab.com/pro/input-options.htm.

Cohen, J. [D.], MacWhinney, B., Flatt, M., & Provost J. (1993). PsyScope: An interactive graphic system for designing and controlling experiments in the psychology laboratory using Macintosh computers. *Behavior Research Methods, Instruments, & Computers*, **25**, 257-271.

Common Controls Replacement Project (CCRP) (2002, October 9). *CCRP high-performance timer objects for VB5* [On line]. Available at http://www.mvps.org/ccrp/controls/ccrptimer5.htm.

Inquisit Millisecond Software (2002). *Inquisit features* [On line]. Available at http://www.millisecond.com/features.sht.

Kessler, B., Treiman, R., & Mullennix, J. (2002). Phonetic biases in voice key response time measurements. *Journal of Memory & Language*, **47**, 145-171.

Light, L. L., Kennison, R., Prull, M. W., & La Voie, D. (1996). One-trial associative priming of nonwords in young and older adults. *Psychology & Aging*, **11**, 417-430.

MICROSOFT CORPORATION (1999, December 26). Microsoft Knowledge Base Article 81592. *Timer2.exe—Timers and timing in Microsoft Windows* [On-line]. Available at http://support.microsoft.com/default.aspx?scid=kb;en-us;81592.

MICROSOFT CORPORATION (2000, October 22). Microsoft Knowledge Base Article 274323. *PRB: Performance counter value may unexpectedly leap forward* [On-line]. Available at http://support.microsoft.com/default.aspx?scid=kb;en-us;274323.

MICROSOFT CORPORATION (2001, January 11). Microsoft Knowledge Base Article 145679. *HOWTO: Use the Registry API to Save and Retrieve Setting* [On-line]. Available at http://support.microsoft.com/support/kb/articles/Q145/6/79.asp.

MICROSOFT CORPORATION (2001, January 11). Microsoft Knowledge Base Article 178456. *SAMPLE: Volume.exe: Set volume control levels using Visual Basic* [On-line]. Available at http://support.microsoft.com/default.aspx?scid = kb;en-us;178456.

MICROSOFT CORPORATION (2001, January 12). Microsoft Knowledge Base Article 187673. *SAMPLE: AUDIOLVL.EXE—Monitor input and output audio* [On-line]. Available at http://support.microsoft.com/default.aspx?scid=kb;en-us;187673.

MICROSOFT CORPORATION (2003, January 8). Microsoft Knowledge Base Article 172338. *HOWTO: Use QueryPerformanceCounter to time code* [On-line]. Available at http://support.microsoft.com/default.aspx?scid=kb;en-us;172338.

MICROSOFT CORPORATION (2003, May 7). Microsoft Knowledge Base Article 147886. *HOWTO: How VB Can Get Windows Status Information via API Calls* [On-line]. Available at http://support.microsoft.com/support/kb/articles/Q147/8/86.asp.

MICROSOFT CORPORATION (2003, December 12). Microsoft Knowledge Base Article 86281. *HOWTO: Play a Waveform (.WAV) Sound File in Visual Basic* [On-line]. Available at http://support.microsoft.com/support/kb/articles/Q86/2/81.asp.

PSYCHOLOGY SOFTWARE TOOLS (2000). *PST serial response box* [On-line]. Available at http://www.pstnet.com/srbox/srb.htm.

RASTLE, K., & DAVIS, M. H. (2002). On the complexities of measuring naming. *Journal of Experimental Psychology: Human Perception & Performance*, **28**, 307-314.

SNODGRASS, J. G., & YUDITSKY, T. (1996). Naming times for the Snodgrass and Vanderwart pictures. *Behavior Research Methods, Instruments, & Computers*, **28**, 516-536.

TAYLOR, J. K., & BURKE, D. M. (2002). Asymmetric aging effects on semantic and phonological processes: Naming in the picture-word interference task. *Psychology & Aging*, **17**, 662-676.

TYLER, L. K., VOICE, J. K., & MOSS, H. E. (2000). The interaction of meaning and sound in spoken word recognition. *Psychonomic Bulletin & Review*, **7**, 320-326.

NOTES

1. The first author taught a graduate-level Visual Basic programming class in the University of Florida's Department of Psychology, and all students, most of whom had no prior programming experience, were able to use the language to program experiments within their areas of psychology.

2. This sound card test program did not work on computers operating with Windows XP.

ARCHIVED MATERIALS

The following materials and links may be accessed through the Psychonomic Society's Norms, Stimuli, and Data archive, http://www.psychonomic.org/archive/.

To access these files or links, search the archive for this article using the journal (*Behavior Research Methods, Instruments, & Computers*), the first author's name (Abrams) and the publication year (2004).

To obtain a copy of an executable version of the demonstration presented here, e-mail your request to the first author at abrams@ufl.edu. The VoiceRelay program is provided as unsupported freeware for noncommercial academic use.

FILE: Abrams-BRMIC-2004.zip.

DESCRIPTION: The compressed archive file contains ten files that comprise the sample Visual Basic Project discussed in the article. The specific files included in the compressed archive are

VRDemo.vbp & VRDemo.vbw, Visual Basic project file.
Form1.frm & Form1.frx, form object for the sample project.
microphone.ico, icon file for the sample project.
clsVoiceRelay.cls, class module that implements the VoiceRelay Functions.
modMicVolCtrl.bas, Module for control of the Windows Microphone Volume.
modWaveIn.bas, module that performs waveform-audio recording.
modTimer.bas, module that implements high-resolution timing.
modSysInfo.bas, module that retrieves information about the current PC system.

AUTHOR'S E-MAIL ADDRESS: abrams@ufl.edu.
AUTHOR'S WEB SITE: http://www.psych.ufl.edu/~abrams/.