TECHNICAL PAPER

COMPUTERIZED VOICE PRODUCTION AND RECOGNITION USING VISUAL BASIC

Francisco Cabello, Dermot Barnes-Holmes, and Ian Stewart

UNIVERSIDAD DE LA RIOJA, NATIONAL UNIVERSITY OF IRELAND - MAYNOOTH, AND NATIONAL UNIVERSITY OF IRELAND - GALWAY

Many researchers have argued that the development of personal computing has transformed the experimental analysis of human behavior over the last two decades (e.g., Cabello, Barnes-Holmes, O´Hora & Stewart, 2002; Dixon & MacLin, 2003). Certainly, the use of personal computers provides important advantages in terms of improving experimental procedures; for example, research ideas can be implemented much more rapidly, validity problems such as the presence of the experimenter are controlled easily, and computer-controlled procedures permit extremely precise application and measurement of relevant variables. Computers are also important because they provide a number of important research tools that would not otherwise be available. Consider the recent advances in neuroimaging techniques (e.g., Dickins, Singh, Roberts, Burns, Downes, Jimmieson, & Bentall, 2001; DiFore, Dube, Oross, Wilkinson, Deutsch, & McIlvane, 2000; Staunton, Barnes-Holmes, Whelan, & Barnes-Holmes, 2003), or the use of reaction times as a dependent variable (O'Hora, Roche, Barnes-Holmes & Smeets, 2002; Staunton, Barnes-Holmes, Whelan & Barnes-Holmes, 2003).

The current article focuses on one of the new research-useful capabilities provided by modern computers, which we will call 'voice technologies'. In the current context, voice technologies will be used to refer to text-to-speech capabilities (programming a computer to read text aloud), and to voice recognition capabilities (programming a computer to recognize natural human speech). Therefore, the first part of this article will deal briefly with the installation of voice technologies in any PC computer. In the second part, the use of both text-to-speech and voice recognition capabilities will be described, together with examples that show how to employ them using the Visual Basic 6 programming language. Finally, we will suggest some uses for this technology that have arisen within our own research programs.

THE MICROSOFT VOICE ENGINE

Among the different research programs that Microsoft conducts, that of voice-based technology appears to be one of the most promising. Recent advances from the Microsoft group dedicated to this technology, coupled with the increasing power of modern computers, have allowed the development of the Microsoft Speech SDK, a computerized voice system that has been made available at no cost to a wide community of users. The website http://www.microsoft.com/speech/ contains a great deal of information about the Speech SDK.

In the current article, we will focus on SDK version 5.1 for Visual Basic 6. However, there are also versions for the newest Visual Basic.NET, the use of which does not differ greatly from the principles described below. Installation of the voice system requires visiting the web address http://www.microsoft.com/speech/download/SDK51/ and downloading the SpeechSDK51.exe file to your computer (the file is about 68 MB, so it might take a while depending on your Internet connection). Once the download is finished, the user should double-click the file to run the installer program. The set-up process is relatively easy, and the user is required only to specify the location on the hard disk where the voice engine is to be placed. After the file copying process, the

Please address correspondence to: Francisco Cabello, Unidad de Ciencias Sociales del Trabajo, Universidad de La Rioja, 26004, Logroño, Spain. E-mail: F.CABELLOLUQUE@terra.es.

For further information about the use of Visual Basic for the development of experimental software, visit the web address http://www.may.ie/academic/psychology/vbindex.shtml.

voice engine is ready to be used. In case the reader finds any problem installing the Speech SDK (or writing any of the examples included below), please email the authors.

VOICE CAPABILITIES IN VISUAL BASIC

Text-to-Speech Capabilities

One feature of the voice engine, with immediate application for experimental procedures, is the capability it provides to program a computer to read text aloud. What follows is a step-by-step example that illustrates how straightforward it is to implement this capability using the Speech SDK within Visual Basic (if you are unfamiliar with the Visual Basic programming language, we recommend that you first consult a recently published article in this journal; Cabello, et al., 2002).

First, select New Project from the File Menu. When a new project with a blank form is generated, select Components from the Project Menu (pressing CTRL and T simultaneously produces the same result), which will open the components menu. Here you will find a list of different controls that are installed in your computer, and that can be managed by Visual Basic. Scroll down through the list, select the 'Microsoft Voice Text' component, and press the OK button. To build the current example, place a Text box at the top of the blank form, a Command button at the bottom, and change the text in the button to 'Talk' using the Caption property. Then, click on the TextToSpeech control (the small mouth at the bottom of your toolbar), and place a Text-to-speech control in the bottom right corner of the form, which should look like the form displayed in Figure 1.

Figure 1

Following the completion of the above sequence of steps, add the code that will control your program. To do this, double-click the Command button, and add the code indicated in Figure 2 under the Command1_Click Section.

```
If Text1.Text <>"" Then
        TextToSpeech1.Speak  Text1.Text
End If
```

Figure 2

Now, when the program is run and the 'Talk' button is pressed, the computer will read aloud the text entered in the Text box (if any). Of course, this is only a very simple demonstration of the use of the voice engine, and there are dozens of parameters that can be customized. For example, the computer voice can be changed to different types (including male and female voices) using the CurrentMode property of the TextToSpeech control, and the speed of the voice can be changed using the Speed property. Furthermore, external files can be used in XML format, that allow for the implementation of very precise and subtle pronunciations. A description of these capabilities is beyond the scope of the current article, but we recommend the documentation included with the SDK package for further reference.

Voice Recognition Capabilities

Apart from the text-to-speech capabilities described above, the Speech SDK offers a powerful human speech recognition system. Although a simple example may readily be built using Visual Basic, its use is more complex than the Text-To-Speech capability, and thus some knowledge of Visual Basic programming is required to fully understand the example. There are numerous books that would assist in this regard, but "Visual Basic for Behavioral Psychologists" by Dixon and MacLin (2003) provides a particularly good introduction to the Visual Basic language.

To generate the example, select New Project from the File Menu to create a new Visual Basic project with a blank form, and place a List box and a Command button within the form. Next, change the Caption property of the button to 'Start'. Your form should now look like Figure 3.

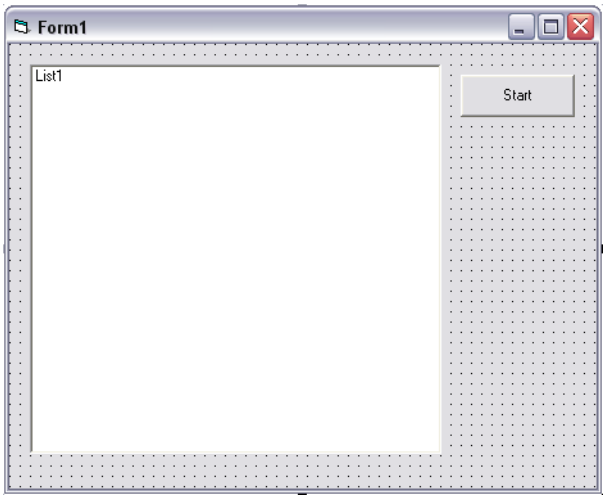

Figure 3

At this stage, there are a few steps that are necessary in order to use the recognition capabilities of the software. These steps involve adding a reference to the objects contained within the Speech SDK (in other words, the Visual Basic software is being told that the SDK will be used). To do this, select References in the Project Menu, select 'Microsoft Speech Object Library' from the list, and press OK. Once the Speech library has been activated, add the code in Figure 4 in the Declarations section of the Code Window. To do this, click the left list placed at the top of this window, and select the '(General)' section.

```
Option Explicit

Dim WithEvents Context As SpSharedRecoContext

Dim Grammar as ISpeechRecoGrammar
```

Figure 4

This code will generate two variables that are defined within the Speech library that you have just activated: Context will create a context for the recognition process, and Grammar will store those words that have been previously recognized. Next, double-click the Command button and type in the code in Figure 5 under the Command1_Click section that will start the voice system when the 'Start' button is pressed.

```
If (Context Is Nothing) Then

        Set Context = New SpSharedRecoContext

        Set Grammar = Context.CreateGrammar (1)

        Grammar.DictationLoad

End If


Grammar.DictationSetState SGDSActive
```

Figure 5

Finally, add the code in Figure 6 at the bottom of the Code Window (after the code above), in order to create a function for the recognition process itself.

```
Private Sub Context_Recognition (ByVal StreamNumber as Long, _

            ByVal StreamPosition as Variant, _

            ByVal RecognitionType as SpeechRecognitionType, _

            ByVal Result as ISpeechRecoResult)


      List1.Additem Result.PhraseInfo.GetText


End Sub
```

Figure 6

Although the foregoing code may appear somewhat daunting, especially if you have little or no experience with computer programming, spending a little time reading an introductory textbook on Visual Basic will help you understand the underlying logic.

Now, connect a microphone to your computer, run the program, start talking slowly, and see how the computer recognizes your words. Bear in mind, when running the program for the first time, the recognition will not be very accurate. This is because the Speech SDK must be trained to distinguish between vocal sounds and noise, and to distinguish particular words in the vocal stream. To train your computer, go to the Speech section of the Control Panel in Windows, and select the Train Profile button (see Figure 7). The subsequent training process, which involves reading aloud different pieces of text, does require

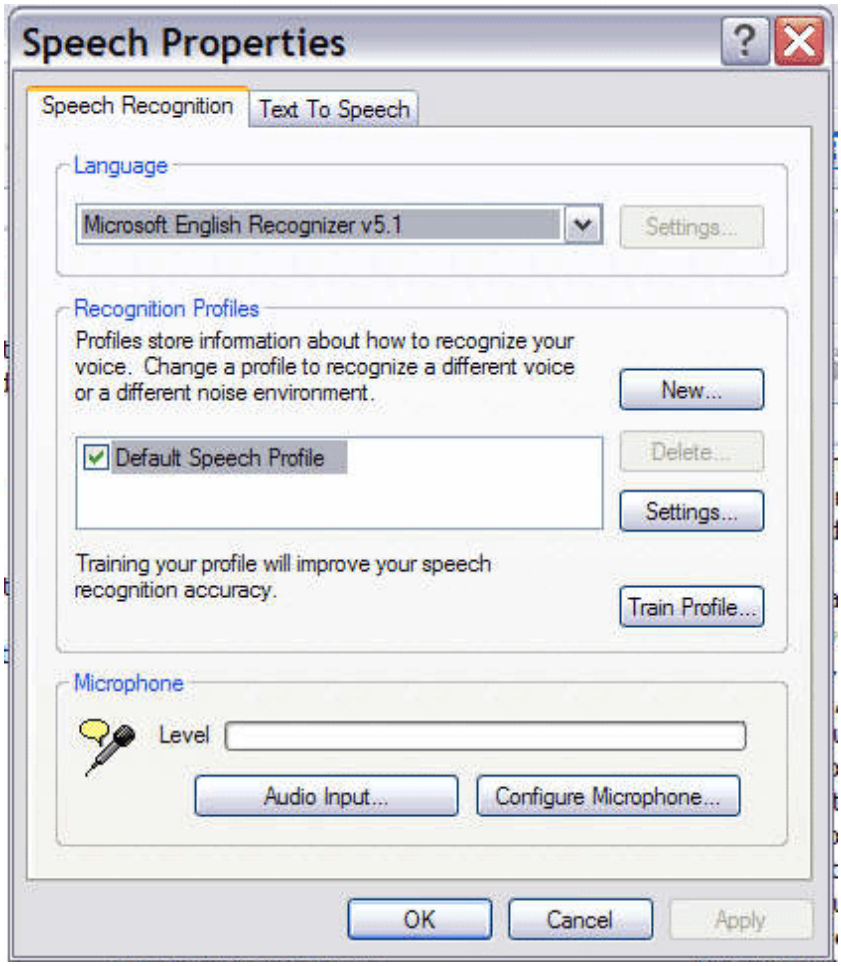

Figure 7

a little time, but doing so dramatically improves the recognition accuracy.

Finally, as was the case for the text-to-speech example, there are numerous options for customizing the voice recognition capability. Of particular interest is the option to define a task-specific grammar, and set the voice engine to recognize only the words contained within that grammar. Again, this process is extensively detailed in the help file provided with the Speech SDK package.

ADVANTAGES AND POSSIBLE USES

In the current article, we have described some examples that show the power of the Microsoft Speech SDK for Visual Basic 6. Indeed, text-to-speech and voice recognition capabilities have been implemented in just a dozen lines of code. We believe that the use of this specific SDK to implement voice production and recognition within experimental procedures has several important advantages. First, the Speech SDK is used within the context of Visual Basic, which is arguably one of the most flexible, user-friendly, and easy to learn programming languages currently available to the behavioral researcher. Second, the Speech SDK can be customized to a great extent, thus providing the ability to adapt its use to very specific research needs. Finally, the SDK can be used in a simple PC computer thus reducing the resources needed to access these powerful capacities within the behavioral laboratory.

There are numerous research areas in which the Speech SDK software might be used for the development of voice-controlled procedures. For example, we are currently developing a Visual Basic program in which the subject has to select among different comparison stimuli using the spoken words LEFT, RIGHT, or CENTER, and another program in which different relations (e.g.,

BEFORE/AFTER, SAME/DIFFERENT) are responded to using the spoken words TRUE and FALSE (pretraining the Speech SDK to recognize this limited number of words requires no more than a minute per subject). Of course, these are relatively simple uses of the SDK software, but other more ambitious uses might also be pursued. For example, numerous sets of spoken feedback and/or instructions could be presented to a subject contingent upon specific performance criteria. And perhaps, most ambitious of all, a sophisticated Visual Basic program could be written to selectively reinforce specific verbal utterances, constituting a type of on-line protocol analysis in which the protocols themselves are used to increase or decrease particular verbal operants (e.g., Cabello, Luciano, Gomez & Barnes-Holmes, in press; Wulfert, Dougher, & Greenway, 1991). In attempting this final application of the SDK to behavioral research, of course, one would first need to expose the voice recognition software to extensive training of the relevant subjects' voice patterns. Nevertheless, the benefits of doing so, in terms of increased experimental precision and reliability, would likely outweigh the time and effort required to undertake the initial training.

In any case, the Speech SDK, combined with the Visual Basic programming language, clearly provides a powerful set of laboratory tools that the behavioral researcher can readily employ at a relatively low cost. In particular, a little Visual Basic programming along with the SDK will allow an investigator to explore the effects of spoken language on behavior (in the form of instructions and feedback) and to use spoken words and phrases, uttered by subjects, as responses that can be analyzed, in vivo, and thus consequated as ongoing behavioral events. In short, a little investment in the technical skills required to master Visual Basic can open up a rich vista of research opportunities for both the beginning and seasoned behavioral researcher.

REFERENCES

Cabello, F., Barnes-Holmes, D., O'Hora, D., & Stewart, I. (2002). Using Visual Basic in the experimental analysis of human behavior: A brief introduction. *Experimental Analysis of Human Behavior Bulletin*, **20**, 17-20.

Cabello, F., Luciano, M.C., Gomez, I. & Barnes-Holmes, D. (in press). Human schedule performance, protocol analysis, and the 'silent dog' strategy. *The Psychological Record*.

Dickins, D. W., Singh, K. D., Roberts, N., Burns, P., Downes, J., Jimmieson, P., & Bentall, R, P. (2001). An fMRI study of stimulus equivalence. *Neuroreport*, **12**, 2-7.

DiFore, A., Dube, W. V., Oross III, S., Wilkinson, K., Deutsch, C. K., & Mcilvane, W. J. (2000). Studies of brain activity correlates of behavior in individuals with and without developmental disabilities. *Experimental Analysis of Human Behavior Bulletin*, **18**, 33-35.

Dixon, M. & MacLin, O. (2003). *Visual basic for behavioral psychologists*. Reno, NV: Context Press.

O'Hora, D., Roche, B. Barnes-Holmes, D. & Smeets, P. (2002). Response latencies to multiple derived stimulus relations: Testing two predictions of Relational Frame Theory. *The Psychological Record*, **52**, 51-75.

Staunton, C., Barnes-Holmes, D., Whelan, R., & Barnes-Holmes, Y. (2003). Priming and event related potentials as measures of derived relational responding. Paper presented at the 29th Annual Convention of the Association or Behavior Analysis, San Francisco, USA.

Wulfert, E., Dougher, M.J. & Greenway, D.E. (1991). Protocol analysis of the correspondence of verbal behavior and equivalence class formation. *Journal of the Experimental Analysis of Behavior*, **56**, 489-504.