SESSION XI

CONTRIBUTED PAPERS: PROGRAMMING SYSTEM DESIGN

WILLIAM SIEGEL, *University of Western Ontario, Presider*

A computer terminal with synthetic speech output*

MORTEZA AMIR RAHIMI
Department of Computer Science

and

JOHN BRYSON EULENBERG
*Department of Linguistics and Oriental and African Languages
Michigan State University, East Lansing, Michigan 48824*

The hardware configuration and software for a voice-output communications station are described. Advantages of such a capability in computer-aided instruction and as an aid for the visually handicapped are noted. The man-computer communication station incorporates a relatively inexpensive digitally controlled voice synthesizer.

In this report, we describe the man-computer communication station that we have in development, which incorporates a relatively inexpensive (less than \$3,000) digitally controlled voice synthesizer. This work has been going on for the last year and a half as part of our general research effort aimed at computer-assisted instruction systems.

We have built this "talking terminal" station with two areas of application in mind: (1) computer-assisted instruction (CAI), and (2) computer use by visually handicapped persons.

CAI

The advantages of a voice-output capability in the area of CAI are many, including the following: (1) It would allow certain subject matter of a linguistic nature to be included in CAI curricula, such as phonetics and certain aspects of foreign language learning. (2) It would add a humanizing element to the student-computer encounter, enhancing the student's enjoyment and emotional involvement. (3) It would extend the range of CAI to preliterate and illiterate populations.

Aids for the Visually Handicapped

In the area of aids for the visually handicapped, we envision a computer-based work area that can be used by

blind persons to prepare and to proofread typed-in copy, such as letters, reports, and school assignments. It would also be used to read out loud any document which can be submitted to a computer in machine-readable form.

One outgrowth that we hope will come from this work is the opening of computer-related vocational opportunities for the blind. Careers in computer programming are often suggested for the visually handicapped. In this field, however, where a misplaced comma can be disastrous, there has been no easy way for a blind programmer to proofread programs or output, whether in card or printed format. Braille output is one answer, and has been implemented at some installations, but it has its drawbacks. The talking terminal can greatly facilitate the work of blind computer users and significantly reduce their dependence on sighted persons for help in I/O situations.

DESCRIPTION OF SYSTEM

Hardware

The hardware configuration of our voice-output communications station is shown in Fig. 1. The communications station is connected to the computer by means of the phone line and an acoustic coupler, in the same way as are most remote terminals. In fact, the combination voice-and-character display station looks to the computer just like a single teleprinter terminal. In this setup, however, a program-controlled electronic switch intercepts incoming signals and relays them to either the 300-baud CRT terminal or to the voice

*We would like to express our thanks to those of our colleagues, students, and friends who have helped us in this work. Charles Yoshida, Ali Behforooz, Leonard Broski, Larry Sniderman, Richard Reid, Steve Rickelman, Dean Detrich, and Max Bishop.

Fig. 1. Hardware configuration.

synthesizer,¹ depending upon the reception of certain control characters. In our present system, the reception of a sequence of two dollar signs (“\$\$”) transfers output to the voice synthesizer, while reception of a subsequent sequence of a “4” followed by a “1” transfers output back to the CRT terminal. Thus, the station functions like an ordinary CRT or teleprinter terminal except for when the computer sends a character string starting with “\$\$” and ending with “41.” The string of characters between these two control characters drives the synthesizer to produce the speech output. The choice of control characters was arbitrary, governed only by the considerations that a double dollar sign was not likely to be needed in our printed output, and that the character string “41” was not recognized by the synthesizer as a pronounceable code.

Software

Our first programming effort was to convert the voice synthesizer into a speech synthesizer by providing the programs needed to code linguistic utterances in a form accepted by the synthesizer. The phonetic images of words and sentences are built up by concatenating the symbols of the phones² of English into strings held in a buffer memory. A stored-up string is released from memory to the output data stream for pronunciation by the synthesizer whenever the appropriate control bit configuration is encountered by the synthesizer (in this case, eight one-bits, corresponding to the ASCII character sequence “00”). Each segmental phone sent to the synthesizer is presented by a 6-bit code, to which two more bits are added to represent one of four levels

of intonation. Each combination of segmental phone and intonation, then, is represented by 8 bits. Since our computer outputs 6-bit ASCII characters, it takes two ASCII characters to represent one phonetic segment with intonation. The synthesizer uses just the four lower-order bits of each ASCII character, as shown in Fig. 2.

Our software configuration is shown in Fig. 3. At the heart of the software package is the Lexicon, which is a file whose entries contain detailed information about every word in the system’s vocabulary. This includes orthographic spelling, phonetic spelling, stress placement, part of speech, and idiosyncratic morphological information. The classical linguistic notion of a lexicon is that of a compendium of the nonregularities of a language. All the regularities of a language are represented in its system of rules. We have built a number of general phonological and morphological rules into our software configuration, but it is still far from total linguistic sophistication. Our goal is not in the realm of artificial intelligence. Our aim is simply to design software configurations which will facilitate the use of synthesized speech as a computer output mode. We recognize the fact that most computer users are not phoneticians, and we hope by means of our software package to minimize the effort involved in performing the phoneticizations necessary to the operation of a speech synthesizer.

For example, when a user wishes to add new words to the lexicon, he is assisted by a phoneticization program which, upon being given the orthographic spelling, comes up with several guesses for the corresponding pronunciation. If one of the computer’s guesses happens to satisfy the user, he gives his permission for that

orthographic representation:	answer			
segmental phones:*	AE	N	S	ER
intonation levels:*	2	2	1	1
octal code of phones:	056	015	037	072
octal code of intonation levels:	300	300	200	200
octal code of combination of phone and intonation level:	356	315	237	272
Binary representation:	11101110	11001101	10011111	10111010
High-Order Bits:	1110	1100	1001	1011
Corresponding ASCII character:	N	L	I	K
Binary code of ASCII character:	001110	001100	001001	001011
Low-Order Bits:	1110	1101	1111	1010
Corresponding ASCII character:	N	M	O	J
Binary code of ASCII character:	001110	001101	001111	001010
Code sent to synthesizer: (Low-Order, then High-Order)	NN	ML	OI	JK

*Phonetic symbols are those used by the manufacturer of the synthesizer Federal Screw Works, and are not identical to those commonly used by linguists and phoneticians.

Fig. 2. Encoding of the phonetic image of a word for transmission to the synthesizer.

Software Configuration

Fig. 3. Software configuration.

version to be entered in the lexicon. The user may, however, reject all of the computer's suggestions and try out his own phoneticization, editing his trial versions until he finds the right pronunciation to be sent to the lexicon.

The system provides a number of readout modes. Given an input text spelled out in phonetic transcription, it will pronounce the text out loud according to the transcription. If the input text is spelled in normal English orthography, it will be analyzed into its component parts, each word looked up in the lexicon, appropriate morphological analyses performed, and a phonetic image thereby obtained for submission to the synthesizer. At present, only the more rudimentary morphological operations are performed, such as checking for the right pronunciation of the plural or first-person singular suffix, orthographic *//s//*, which is pronounced [s] after nonvoiced stops such as the /t/ in "cats," but [z] after voice consonants and vowels, such as after the /g/ in dogs"; and, when written *//es//* after sibilants, it is pronounced [uhz].

If the reading program encounters a word not in the dictionary, it prints a message to that effect on the CRT and then either tries to pronounce the new word or else pronounce the spelling, depending upon the option taken by the user. Thus, consider the following hypothetical excerpt from a recent newspaper: "Today, President Nixon nominated a new Vice President. He is

Gerald Ford of Michigan." In this example, suppose the words "Gerald" and "Ford" were not in the lexicon. They would then be spelled out by the reading program, which would inform the user, "Today, President Nixon nominated a new Vice President. He is GEE-EE-AHR-AY-ELL-DEE SPACE EFF-OH-AHR-DEE of Michigan."

IMPLEMENTATION IN WRITING PROGRAMS WITH VOICE OUTPUT

One advantage of our communications station is that a computer user can write and execute programs with voice output, using the usual programming languages with which he is familiar, such as FORTRAN.

Direct Coding

If no spoken output statements in his program include variables, the programmer can code the phonetic output directly into his program, as illustrated below:

```

      .
      .
      WRITE(6,34)
34  FORMAT(26H$KMMNVNVNRORKJKRNTNVN0041)
      .
      .
  
```

Statement 34 here formats directly the coded phonetic

input to the synthesizer. The communication station is indexed as Output Device #6. The particular phonetic code given in the example is for the question "who are you?"

Precompile Mode

In actual practice, however, we would not wish to burden the programmer with the job of phoneticizing his messages and then coding the phonetic transcription. We have therefore provided for a phonetic precompiler which accepts as input a FORTRAN program, containing pseudo-output statements written in English orthography, and generates a program identical in every respect to the input, except that, for each pseudo-output statement, it would have the properly phoneticized and coded true FORTRAN output statements. Thus, in place of the two statements given above, the programmer need merely write

SAY("WHO ARE YOU?")

and the precompiler will generate the WRITE and FORMAT statements automatically.

FORTTRAN Subroutine for Voicing Variables

The above method of preparing FORTRAN programs with voice output, which we may call the precompile mode, will not, however, be sufficient to cover those cases where values of variables are to be spoken out. And, indeed, it is one of the virtues of the digital voice synthesizer that it enables us to output linguistic strings which we have not stored in advance. For this purpose, we need to construct our phonetic coding in the course of the execution of a program. To accomplish this, we can add a subroutine called VOICE, which is compiled with the program and which can access, during the course of the execution of the program, the file containing the lexicon. For many applications, of course, only a small subset of the system's dictionary need be accessed, namely, the part necessary for reading

numerical variables. Rather than being placed on a separate file, this minidictionary can be compiled, along with a number-parsing routine, together with the main program.

An example of a FORTRAN program using the subroutine VOICE is shown below. The example is from part of a hypothetical CAI program.³

```

      .
      .
      WRITE(10,24) I,J
      CALL VOICE('WHAT IS THE SUM OF THE INTEGERS SHOWN?')
99  READ(5,36) K
      IF ((I+J).NE.K) GO TO 100
      CALL VOICE('GOOD! THE CORRECT ANSWER IS 'K'.')
      .
      .
100  CALL VOICE('NO. THE CORRECT ANSWER IS NOT 'K'. TRY AGAIN.')
      GO TO 99
      .
      .

```

Conclusion

The day of the relatively inexpensive talking terminal has arrived. Although our present system does not command all the subtlety inherent in human speech, it is nevertheless a dramatic advance in human-computer communication. A somewhat sophisticated linguistic interface is needed to achieve intelligible output, but a working system is now in operation which can easily be incorporated into most existing on-line configurations. We feel that many systems like this one will be springing up at computer installations around the world in the next few years.

NOTES

1. The synthesizer is a VOTRAX-IVTM voice synthesizer, provided by the manufacturer, Federal Screw Works, Detroit, Michigan. The CRT terminal, a CDC 713, also is provided by the manufacturer, Control Data Corporation, Minneapolis, Minnesota. We wish to thank both companies for their assistance in providing this equipment.

2. Phones are minimal units of linguistically significant sound, not to be confused with phonemes, which are classes of phones.

3. Actually, most CAI courses being developed at MSU and other centers would not be written in FORTRAN, but rather in PLANIT or other languages specially designed for CAI applications. Such languages can also make use of the voice output capability of our terminal.