THE ENHANCEMENT OF CATIB WITH GRAPHICS, SPEECH AND SOUND EFFECTS

by

Mohammad M. Mandurah, Sabri A. Mahmoud

College of Computer and Information Sciences
King Saud University
P.O.Box 51178, Riyadh 11543

and

Fouad A. Dehlawi

Electrical Engineering Department
College of Engineering
King Abdul-Aziz University, Jeddah

Abstract:

The enhancement of CATIB is discussed in this paper with special emphasis on the implementations of the advanced features of the language. These include features such as support of high resolution graphics, support of speech synthesis and sound/music subsystems, and commands for text presentation and highlighting. Files that contain speech data or sound/music data can be accessed by the user, and the data can be sent to the appropriate device. Graphics commands are also implemented in CATIB to allow the user to fully utilize the powerful graphics capablities of the hardware. Using CATIB in the development of CAI courseware, graphics, speech messages and sound effects can be combined and intermixed with the text to enhance the effectiveness of the educational material.

1. Introduction:

CATIB is the first Arabic programming language that was especially developed to facilitate the production of CAI courseware. The language is so designed to have a very simple syntax with highly conversational nature. This makes it easy for persons with no prior experience in programming, such as teachers, to learn the language and effectively use it in writing and editing CAI courseware.

CATIB was first introduced at the 7th Saudi National Computer Conference that was held in Riyadh in 1984, and a paper that defines the language appeared in the conference proceeding [1]. Since then, the language has been subjected to several improvements and revisions that can be summarized in the following:

i- the general instruction format was redefined for better performance and ease of use.
ii- implementation of instructions to process speech and music data.
iii- implementation of instructions to facilitate graphics creation and manipulation.

The development of CATIB is part of a large project supported by SANCST to develop computer-assisted instructional systems for the Arab societies [2]. This paper reviews the current status of CATIB, with emphasis on the implementation of the advanced features in the language.

2. Definition of the Language:

The general instruction format of CATIB was redefined to enhance the language performance and further simplify its use. Under the new definition, CATIB keywords can be used as full words or as two characters abbreviations. CATIB instructions now follow the following format:

<keyword> b [<cond>] : [<message>]

where

<keyword>	is CATIB full keyword or two characters abbreviation for the keyword.
b	blank.
[<cond>]	an optional conditional which could be yes, y, no, n.
[<message>]	an optional message which could be: a. a text to be operated on b. an instruction modifier c. empty (not used).

The new definition of the language allows the same keyword to be used to define different forms of an instruction. This feature has several advantages. First, each instruction has the power of several instructions. Secondly, the user does not have to remember many keywords. The different forms of the instructions were kept consistent. To clarify the above points, the following exapmle shows how a CATIB keyword is used to form different instructions.

The basic "Uktob" keyword can have the following forms:

```
Uktob   : [<message>]
Uktob Y : [<message>]
Uktob N : [<message>]
Uktob   :=
Uktob Y :=
Uktob N :=

Uktobh  : [<message>]
Uktobh Y: [<message>]
Uktobh N: [<message>]
Uktobh  :=
Uktobh Y:=
Uktobh N:=
```

where

[<message>] could have the following forms:

(1) and empty or blank string.
(2) a string of characters to be typed.
(3) a string of characters to be typed with embedded string or integer variables.
(4) a string or integer variable or both whose content are to be typed.

"=" indicates that the content of the "accept" buffer should be typed.

"h" adding "h" to the keyword "uktob" specifies that no carriage return/line feed is needed.

The general CATIB user program has the form given below. A user program is very simple with few required statements to simplify the use of the language by persons who have no programming background. A program consists of a main part and subroutines part. The main part is formed from any number of CATIB statements terminated by the END statement. A subroutine must start with a label and end with and END statement. The depth of the nested subroutines is limited to 5, which we think is adequate for CATIB applications.

CATIB User Program Form

<CATIB program>	<main program> [{subroutine,..}]
<main program>	{<statement>,.....} <end>
<statement>	{<label> ! <statement>}
<label>	'* character string'
<statement>	<CATIB defined statement>
<subroutine>	<label> {<statement>,....} <end>

where

< >	at least one entry is required
[ ]	optional entry
{ }	entry repeated any number of times
' '	character string is enclosed
!	or, either entry is possible

3. Enhanced Features of CATIB:

Several instruction were added to CATIB to cover certain needs of CAI applications and to make the language more powerful and easier to use. These are the following:

(1) Jump [<Y!N>] : >P

This instruction can be used to jump to the next problem. A possible use for the instruction is to skip more explanations in a lesson if his answers were satisfactory.

(2) Remarkw [<Y!N>] : [<message>]

This instruction can be used to write remarks to a file to help tracking the student activities for the purpose of evaluating his performance.

(3)
NV :	clear all previous variables
NV : $	clear all previous string variables
NV : @	clear all previous integer variables

These instruction which clear the variables assist the efficient utilization of memory space.

(4) PR :

This is a problem indicator. It can be used with the (Jump : >P) instruction described above.

(5) CN [<cond>] : [<message>]
CNh [<cond>] : [<message>]

These instructions center the message on the screen and skip to the next line (CN) or hang (CNh).

(6) CP [<cond>] : [<message>]
CP [<cond>] :=
CPh [<cond>] : [<message>]
CPh [<cond>] :=

These instructions are similar to the "uktob" instruction with its different variations. However, these instructions send the text to the printer, not to the screen.

(7) Foot : [<text>]

Type the text at the bottom of the screen and execute an "accept" command. This instruction can be used to put the display of screens under user control.

(8) JF : <LM>,<RM>
CJ :

These instructions control the flag for margin justification. JF sets the left margin at LM and the right margin at RM. CJ cancels margin justification.

4. Implementation of Speech Instructions:

Speech provides a natural and efficient channel of communication between the computer and the user. Studies have shown that the students learm more by increasing the number of channels that communicate the educational material to them (the ear in addition to the eye) [3,4]. Realizing the importance of speach, especially for CAI systems, a speech synthesis subsystem has been developed in this project [5]. The speech output unit is based on the Texas Instruments' TMS-5220 Voice Synthesis Processor (VSP). This chip uses linear predictive coding (LPC) to generate high quality speech at low data rate.

In this section, the structure of the commands that allow the user of CATIB to exploit the capability of the speech subsystem is discussed. This starts with a discussion to the types of speech data and then to the algorithms used in implementing the speech commands.

4.1 Speech Data:

The nature of the speech messages to be used in CATIB should be analyzed before deciding on the suitable structure and algorithms to be used in implementing the speech statements. The speech messages in CATIB can be divided into two general types:

(1) course independent speech messges (global messages).
(2) course dependent speech messages.

The course independent speech messages are of repetitive nature (i.e. the same message is used again and again in almost every course). In this type, some messages can be interchanged with others without any change to the required meaning. Examples of this type of messages are the following:

(a) messages to respond to a correct answer from the student
(b) messages to respond to a wrong answer from the student
(c) operational messages ...etc

The usage of specific words is normally not necessary or required. In fact, the use of different messages to a student response, say a correct answer, is better than using the same message again and again; it is less boring to the student.

The course dependent messages, on the other hand, rarely repeat and the exact words of the teacher should be produced. This message type is part of the course material and hence does not repeat. It is normally used once.

As shown above, the two speech messages types are of different nature. Hence, for better performance of CATIB speech messages, the two is to be implemented differently.

4.2 Structure of Speech Statements:

In designing the structure and algorithms to implement CATIB speech statements, we have to examine the requirements of CAI applications. These can be summarized in the following:

(a) Simple structure and speech statements to make it easy for the teacher to use.

(b) The use of reasonable size of storage. Speech data require much more storage than text data; nearly 10 KByte for one minute of speech.

(c) Whenever possible, different speech messages should be used in response to the student answers to avoid boring him.

To fulfill the above requirements two types of speech instructions had to be implemented in CATIB: lesson independent and lesson dependent instructions.

4.2.1 Lesson Independent Instructions:

The lesson independent instructions use global speech data and they have the following forms:

(i) SG [<cond>] : <file-name> [<,message-number>]

This form means: say the message specified by message-number in the file named file-name. If the message number is zero or absent then say a random message from that file.

(ii) SG [<cond>] : <message>

Here <message> is a string of characters. This instruction means say the message given by the character string <message>.

The lesson independent (global) speech messages are kept in different files for ease of reference, retrieving speech data, and simplification of processing of CATIB statements. Each group of related messages are kept in one file with a special name. For example, all speech messages used to respond to a student correct answer are kept in a seperate file, and so on. These speech data files are produced once and are used by all courses.

For ease of use of the above messages, all file names and their contents of speech messages are printed on seperate cards for quick reference by the teachers using CATIB. To use any message, the teacher need only specify the file name and the message number from that file.

If a teacher finds it difficult to use the above technique, he can use the second form described above. Using this form, the teacher has to write the message to be said in <message>. A programmer at a later stage will scan the text of CATIB lessons and replace all global lesson independent speech messages by their corresponding message-numbers and file-names. If any of these messages do not exist in current files, they can be created and incorporated in the current speech files or in new speech files.

4.2.2 Lesson Dependent Instructions:

The lesson dependent instructions use speech data that are dependent on the lesson under run. These type of instructions may any of the following forms:

(i) SY [<cond>] : <message>

Say the text of the message

(ii) TS [<cond>] : <message>

Type the message on the screen and say it.

(iii) TSH [<cond>] : <message>

Same as (ii) but the cursor on the screen stays on the same line (i.e. no CRLF).

The <message> in the above instructions are caracter strings that represent the actual text need to be said to the student. These speech messages must be converted into speech data that can be sent to the speech sybsystem. For this purpose, a speech preprocessor program performs the conversion task in the following steps:

(1) A preprocessor program scans the CATIB lesson for any <message> within a lesson dependent speech instruction. A file will be created that contains all messages that need to be converted into speech data. These messages will be arranged sequentially according to their occurance in the lesson. This process can be done manually also.

(2) The speech data for the messages will be then generated on the speech development system and stored in another file under the same name of the lesson and with extension (.sph). The speech data will be arranged in the same sequence of the messages.

(3) Another program then scans the CATIB lesson again and replaces all <messages> with their respective file-names and message-numbers. The final executable version of the lesson should not have any speech <messages>, and all speech instructions should refer to file-names and message-numbers.

5. Implementation of Music and Sound Effects:

Music and sound effects can be effective tools to attract the attention of the students and increase their interest and enthusiasm in learning. Advancement in electronics technology has resulted in the production of many sophisticated single-chip music and sound generators. In this project, a music-sound subsystem is also developed that is based on the General Instruments AY-3-8910 programmable sound generator. Upon receiving the appropriate commands and data such sound generators can generate a variety of tunes, melodies and sound effects.

The commands in CATIB that allows the manipulation of music and sound effects have only one form:

PY [<cond>] : <file-name> [,<piece-number>]

This command means: play a music or sound effect piece given by piece-number from a data file given by file-name. If piece-number is absent or equal to zero, then CATIB will play a random piece from that file.

CATIB processes the data for music and sound effects in a similar way as it does to speech data. The data for music and sound effects pieces must be prepared separately and put in data files with appropriate names. File cards can also be prepared that show the conents of each music and sound file.

6. Implementation of Graphics Instructions:

Graphics is an essential element in CAI systems. Figures, shapes, pictures and charts are frequently used to explain concepts and ideas, represent data and results, simulate equations and processes, and to carry knowledge and information that are difficult to describe in words [6-8]. Therefore, CAI systems (hardware and software) must be provided with all the tools to benefit from the many advantages and uses of graphics.

Fortunately, the terminal developed in this project is equipped with an advances graphics display controller (GDC) that is based on the NEC uPD-7220 chip [2,9]. This device is an intelligent microprocessor peripheral with a powerful instruction set that allow it to manage the display memory and generate high resolution graphics.

In the original definition of CATIB, several graphics and cursor and text control commands have been proposed [2]. In the following, the opcodes and formats for some of these commands are defined. Also, a description to each command and how to use it is given.

a) ENTER GRAPHICS MODE:

This command signals the terminal to enter into the graphics mode and interpret all coming information as graphics data and commands. All of the graphics commands and data are built of seven-bit binary words (parity bit not used). A command or data byte that is equal to zero is not permitted because some operating systems filter out any zero-valued bytes.

Command Form : ESC 0

b) MOVE TO (X,Y):

The virtual pointer is assigned the absolute coordinates (X,Y). Although the terminal has a resolution of 640 dots x 408 lines, X and Y have maximum values of 1024 to allow for future enhancements.

Command Form:

opcode	P	0	1	1	0	1	X1	X0
1st operand	P	1	X7	X6	X5	X4	X3	X2
2nd operand	P	1	Y3	Y2	Y1	Y0	X9	X8

3rd operand	P	1	Y9	Y8	Y7	Y6	Y5	Y4

In designing the opcode and operand bytes, the parity bit is not used and bit 6 in the opernds bytes are set to 1 so that a zero-valued byte is avoided.

c) POINT AT (X,Y):

The virtual pointer is assigned the absolute coordinates (X,Y).

Command Form:

operand	P	0	0	1	1	0	X1	X0

The 1st, 2nd and 3rd operands are the same as in MOVE TO command.

d) LINE TO (X,Y):

A line is drawn from, but not including, the virtual pointer's currently assigned absolute coordinate to the absolute coordinate (X,Y). The line drawn is subject to the current line style. At the completion of this command, the virtual pointer is assigned the absolute coordinate (X,Y).

Command Form:

opcode	P	0	1	1	0	0	X1	X0

The 1st, 2nd and 3rd operands are same as in MOVE TO command.

e) RECTANGLE TO (X,Y):

A rectangle is drawn with the virtual pointer's currently assigned absolute address as one coordinate and the absolute coordinate (X,Y) as the diagonally opposite vertice. At the completin of this command, the virtual pointer is assigned the absolute coordinate (X,Y). The rectangle's sides drawn are subjected to the current line style.

Command Form:

operand	P	0	1	0	1	0	X1	X0

The 1st, 2nd and 3rd operands are same as in MOVE TO command.

f) AREA FILL TO (X,Y):

This command is similar to RECTANGLE-TO command; however, the area inside the rectangle is filled. Starting at, but not including, the virtual pointer's currently assigned absolute coordinate, a horizontal line is drawn to the opposite side of the rectangle. When possible, a second line starting at the original side of the rectangle is drawn adjacent to the first

line. This procedure is repeated until the area is filled. The line drawn is subject to the line style attribute.

Command Form:

opcode	P	0	1	0	1	1	X1	X0

The 1st, 2nd and 3rd operands are the same as MOVE TO command.

g) CIRCLE (R):

A circle with the radius R is drawn. The coordinates of the center is the virtual pointer's currently assigned absolute coordinate. The radius of the circle has a maximum value of 255 dots.

Command Form:

opcode	P	0	0	1	1	1	R1	R0
1st operand	P	1	R7	R6	R5	R4	R3	R2

h) LINE STYLE:

This command permits solid, dashed or dotted lines to be generated. The style of the line is determined by the pattern byte (Z). The eight bit line style pattern is repetitively traced to the screen when drawing a line.

Command Form:

opcode	P	0	1	1	1	0	Z1	Z0
1st operand	P	1	Z7	Z6	Z5	Z4	Z3	Z2

Examples of line styles are the following:

line style	pattern byte
solid	1 1 1 1 1 1 1 1
dotted	1 0 1 0 1 0 1 0
dashed (long)	1 1 1 1 0 0 0 0
dashed (short)	1 1 0 0 1 1 0 0

i) SET COLOR (C):

This sets the default color for the following graphics. One color out of 16 colors can be chosen although the terminal currently supports 8 colors only.

Command Form:

opcode	P	1	0	0	C3	C2	C1	C0

j) ERASE SCREEN:

This command clears the screen

opcode	P	0	0	0	0	1	*	*

Here (*) means don't care.

k) EXIT GRAPHICS MODE:

Exit graphics and return to alphanumeric mode.

opcode	P	1	1	1	1	1	*	*

In this section, only some of the basic graphics commands are defined and implemented. More commands are under consideration. These include pattern drawing, Turtle Graphics (similar to LOGO), and graphics editing using light pens or digitizers.

7. Summary:

The incorporation of speech, music, sound effencts and graphics in CATIB was discussed in this paper. The commands that facilitate the use of these advanced features by CATIB users in the development of CAI courseware were defined, and the implementation of these commands was discussed.

REFERENCES:

1- Mandurah, M.M. and El-Azhary, I., "CATIB: An Arabic Authoring Language for CAI", Proc. of the 7th Saudi NCC, Riyadh, 1984, pp. 271-278.

2- Mandurah, M.M. and Dehlawi, F.A., The Development of Computer Assisted Instructional Systems in Arabic, SANCST project # AR-5-100, Saudi National Center for Science and Technology, Riyadh, 1984.

3- Chapanis, A., Parrish, R.N., Ochman, R.B. and Weeks, G.D. "The Effect of Four Communication Modes on the Linguistic Performance of Teams During Cooperative Problem-Solving", Human Factors, 19, 1977, pp. 101-126.

4- Neels, F., Lienard, J.S. and Mariani, J.J., "An Experiment of Vocal Communication Applied to Computer-Aided Learning", Computers in Education, R. Lewis and D. Tagg (editors), North Holland, 1981, pp. 337-341.

5- Mandurah, M.M., Kanawati, A.G., Kanawati, A.N. and El-Azhary, I., "The Development of a Speech Subsystem for Computer-Assisted

Instruction", Accepted for Presentation in the 8th Saudi NCC to be held in Oct. 1985 in Al-Khobar, Saudi Arabia.

6- Paivio, A., Imagery and Verbal Processes, Holt Rinchart and Winston, New York, 1971.

7- Lodding, K.N., "Iconic Interfacing", IEEE Computer Graphics, Vol. 3, No. 2, March/April 1983, pp. 11-20.

8- Kent, E.W., The Brains of Men and Machines, McGraw Hill, New York, 1981.

9- Mandurah, M.M. and Al-Mousa, A.O., "The Use of Graphics to Generate High Quality Arabic Characters", Accepted for Presentation at the 8th Saudi NCC, to be held in Oct. 1985 in Al-Khobar, Saudi Arabia.